Model Predictive Control - Notes

Supernotes

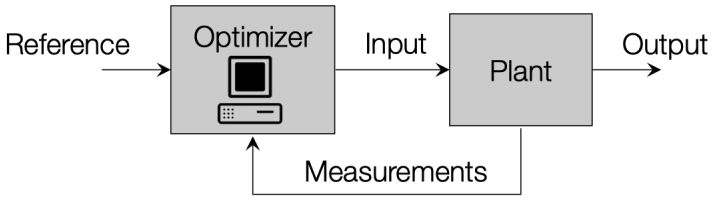
Contents

Chapter 1

Introduction

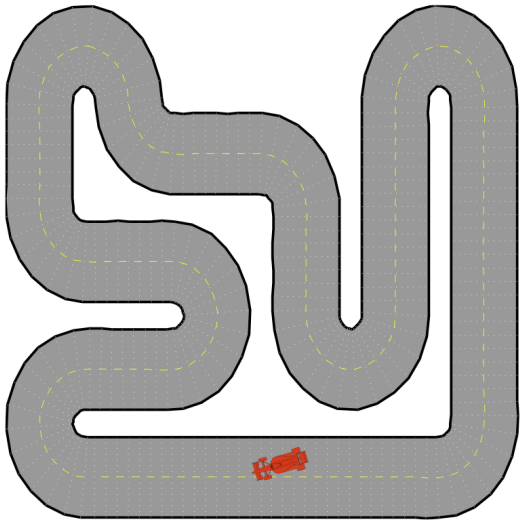
1.1 Optimization in the Loop

In a classical control loop we have the reference signal and a measurements feedback going into our controller. Out controller will then compute the input that it has to give to the plant to generate the desired output. In model predictive control, the classical controller is replaced by an optimization algorithm, meaning that the reference signal and the feedback measurements will be fed into an optimizer, that will then output to the plant. **The optimization uses predictions based on a model of the plant.**

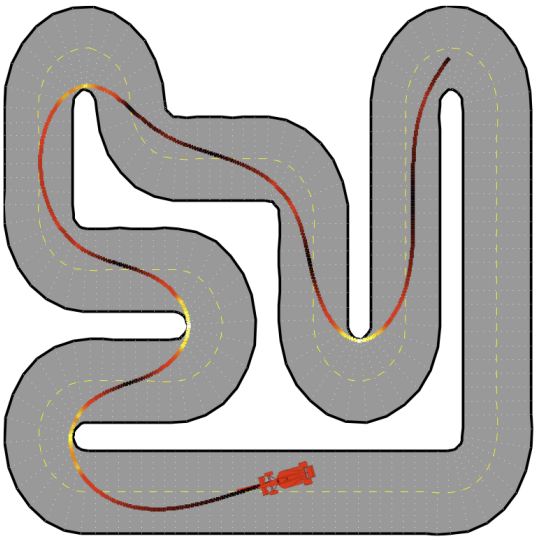


Example - Lap Time Optimization

Say we want to make an autonomous car go around a given track as fast as possible. The objective here is to minimize lap time, however we also have quite a few constraints in our problem. We want to avoid other cars, we want to stay on the road, not skid and stay within the limits of acceleration that our car can give.



The intuitive approach here would be to look forward and plan a path based on the road condition, upcoming corners, abilities of the car, etc. However, we can solve the optimization problem we listed above to compute the minimum-time path.



Let's say we solve the optimization problem to compute the minimum-time path. What should we do if something unexpected happens? Say we didn't see a car around a corner! This motivates us to introduce feedback in our optimization problem.

We will make a controller that follows the steps below :

- Solve the optimization problem to comput minimum-time path
- Obtain a series of planned control actions
- Apply **first** control action
- Repeat the planning procedure

We can visualize this method of control in figure 1.1.

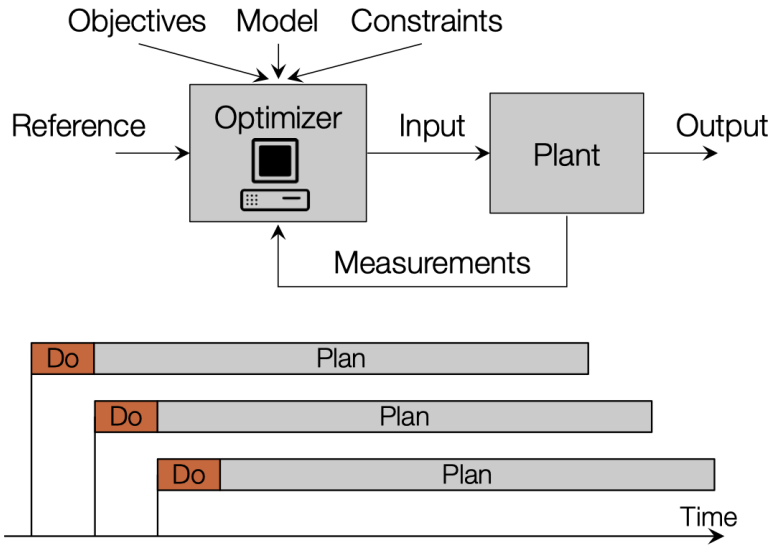
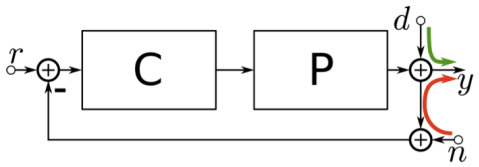


Figure 1.1

1.2 Concept of MPC

Classical Controller Design
We want to design the controller C

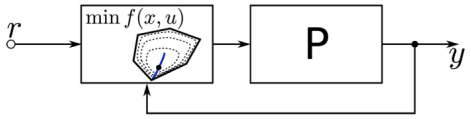


Dominant issues addressed :

- Disturbance rejection ($d \to y$)
- Noise insensitivity ($n \to y$)
- Model uncertainty

(usually in **frequency domain**)

MPC
Real-time, repeated optimization to choose $u(t)$ - often in supervisory mode



Dominant issues addressed :

- Control constraints (limits)
- Process constraints (safety)

(usually in **time domain**)

1.2.1 Constraints in Control

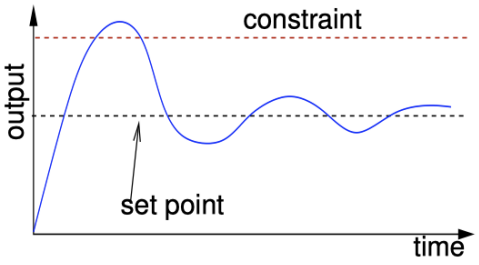
All physical systems have constraints :

- Physical constraints, e.g. actuator limits
- Performance constraints, e.g. overshoot
- Safety constraints, e.g. temperature/pressure limits

Optimal operating points are often near constraints.

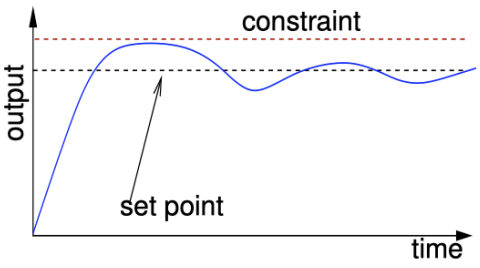
With classical control methods we have :

- Ad hoc constraint management
- Set point sufficiently far from constraints
- Suboptimal plant operation



With model predictive control we have :

- Constraints included in the design
- Set point optimal
- Optimal plant operation



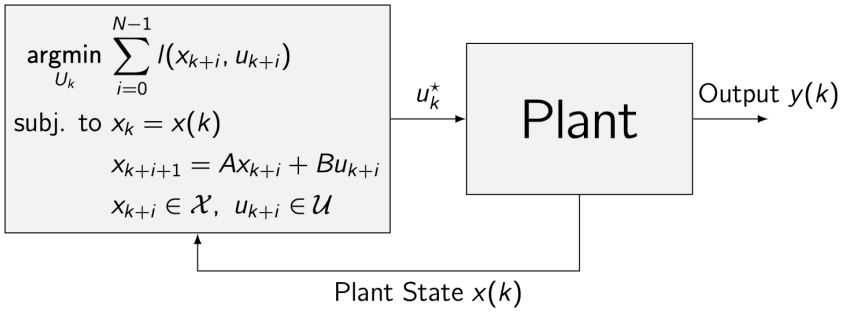
1.3 MPC - Mathematical Formulation

$$
\begin{aligned}
U_k^*\left(x\left(k\right)\right) \coloneqq \arg\min_{U_k} & \sum_{i=0}^{N-1} l\left(x_{k+i}, u_{k+i}\right) & \\
\text{subject to} \quad & x_k = x(k) & \text{measurement} \\
& x_{k+i+1} = Ax_{k+i} + Bu_{k+i} & \text{state model} \\
& x_{k+i} \in \mathcal{X} & \text{state constraints} \\
& u_{k+i} \in \mathcal{U} & \text{input constraints} \\
& U_k = \{u_k, u_{k+1}, \ldots, u_{k+N-1}\} & \text{optimization variables}
\end{aligned}
$$

The problem is defined by :

- **Objective** that is minimized
- Internal **system model** to predict system behavior
- **Constraints** that have to be satisfied

We can view this controller in the full control loop :



At sample time :

- Measure / estimate current state $x(k)$
- Find the optimal input sequence for the entire planning window N :
  $U_k^* = \left\{u_k^*, u_{k+1}^*, \ldots, u_{k+N-1}^*\right\}$
- Implement only the **first** control action u_k^*

Chapter 2

System Theory Basics

To refresh our memory, here are the requirements for MPC :

- A model of the system
- A state estimator
- Define the optimal control problem
- Set up the optimization problem
- Get the optimal control sequence (solve the optimization problem)
- Verify that the closed-loop system performs as desired

We will repeat some of the basics on modelling and linear systems in this chapter.

2.1 Models of Dynamic Systems

The goal here is to introduce mathematical models to be used in Model Predictive Control for describing the behaviour of dynamic systems. We can classify our systems as state space/ transfer functions, linear / non-linear, time varying / time invariant, continuous-time / discrete-time, deterministic / stochastic models. If not stated differently, we will be using deterministic models.

Models of physical systems derived from first principles are mainly : non-linear, time-invariant, continuous-time, state space models ($*$). Target models for standard MPC are mainly : linear, time-invariant, discrete-time, state-space models ($\dagger$). In this section we will focus on how to transform $*$ models to $\dagger$ models.

We will use the following abbreviations to simplify notation :

LTI – Linear, time-invariant
DT – Discrete time
CT – Continuous time

Initial Remarks

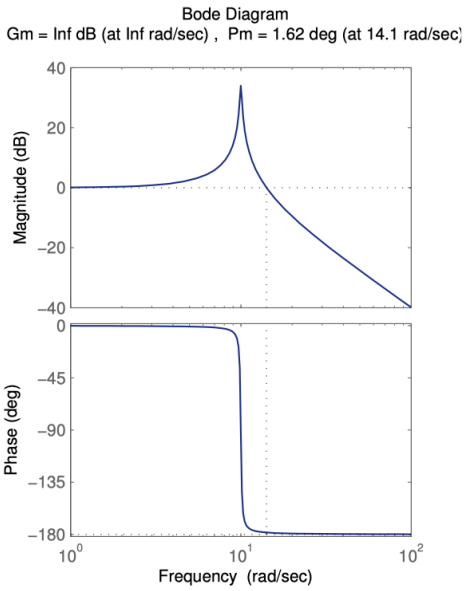
In classical controller design we use **transfer functions**. We can represent the input-output behaviour of continuous time LTI systems in the frequency domain :

$$Y(s) = G(s)U(s)$$

And we know the theory and have the tools for the analysis of closed-loop behaviour, e.g. Bode plots etc.

The problem is that these tools usually don't extend to non-linear systems (recall : constrained systems have non-linear dynamics).

We will focus on the state-space representation of systems from now on.



2.1.1 Nonlinear Time-Invariant Continuous-Time State Space Models

We represent our system as such :

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

with

$x \in \mathbb{R}^n$	State vector
$u \in \mathbb{R}^m$	Input vector
$y \in \mathbb{R}^p$	Output vector
$g(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$	System dynamics
$h(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$	Output function

This representation can apply to a very general class of models. Higher order ODEs can be easily brought to this form (as we will see later). Analysis and control synthesis are generally hard ($\to$ linearization to bring it linear, time invariant, continuous time, state-space form).

Equivalence of one n^{th} order ODE and n 1^{st} order ODEs

Say we have the system dynamics under the form :

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

In order to simplify this, we define a different state for each of our derivatives :

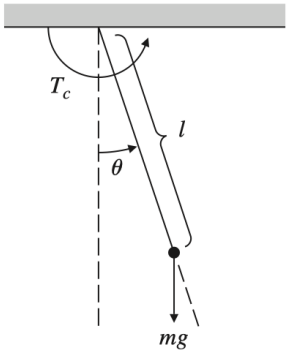
$$x_{j+1} = x^{(j)} \qquad j = 0, \ldots, n-1$$

Our transformed system is then

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

Example - Pendulum

Moment of inertia :	ml^2
Torque caused by external force :	T_C
Torque caused by gravity :	$mgl\sin(\theta)$



Our system's equation of movement is then :

$$ml^2\ddot{\theta} = T_C - mgl\sin(\theta)$$

Using $x_1 \coloneqq \theta$, $x_2 \coloneqq \dot{\theta} = \dot{x}_1$ and $u \coloneqq T_C/ml^2$, the system can be brought to standard form :

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

The output equation depends on the measurement configuration, for example, if θ is measured then $y = h(x,u) = x_1$.

2.1.2 LTI Continuous Time State Space Models

Our continuous time system is represented in state-space form :

$$\dot{x} = A^c x + B^c u$$
$$y = Cx + Du$$

with

$x \in \mathbb{R}^n$	State vector
$u \in \mathbb{R}^m$	Input vector
$y \in \mathbb{R}^p$	Output vector
$A^c \in \mathbb{R}^{n\times n}$	System matrix (continuous time)
$B^c \in \mathbb{R}^{n\times m}$	Input matrix (continuous time)
$C \in \mathbb{R}^{p\times n}$	Output matrix
$D \in \mathbb{R}^{p\times m}$	Throughput matrix

Vast theory exists for the analysis and control synthesis of linear systems.

Solution to linear ODEs

If we consider the ODE (written with explicit time dependance)

$$\dot{x}(t) = A^c x(t) + B^c u(t)$$

with initial condition $x_0 \coloneqq x(t_0)$, then its solution is given by :

$$x(t) = e^{A^c(t-t_0)}x_0 + \int_{t_0}^{t} e^{A^c(t-\tau)}Bu(\tau)\,d\tau \quad \text{with} \quad e^{A^c t} \coloneqq \sum_{n=0}^{\infty}\frac{(A^c t)^n}{n!}$$

The issue with LTI systems is that most physical systems are non-linear, however linear systems are much better understood. Non-linear systems can be well approximated by a linear system in a "small" neighborhood around a point in state space. The idea is to have the controller keep the system around some operating point. If we replace the non-linear system by a linearized

system around this operating point, we can have good control. The first order Taylor expansion of the function f will give us the linearized system around the operating point $\bar{x}$:

$$f(x) \simeq f(\bar{x}) + \left.\frac{\partial f}{\partial x^T}\right|_{x=\bar{x}} \cdot (x - \bar{x}) \qquad \text{with} \qquad \frac{\partial f}{\partial x^T} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Linearization

If we have a stationary operating point x_s, u_s : $\dot{x}_s = g(x_s, u_s) = 0$, $y_s = h(x_s, u_s)$

$$\dot{x} = \underbrace{g(x_s, u_s)}_{=0} + \underbrace{\left.\frac{\partial g}{\partial x^T}\right|_{\substack{x=x_s \\ u=u_s}}}_{=A^c} \cdot \underbrace{(x - x_s)}_{=\Delta x} + \underbrace{\left.\frac{\partial g}{\partial u^T}\right|_{\substack{x=x_s \\ u=u_s}}}_{=B^c} \cdot \underbrace{(u - u_s)}_{=\Delta u}$$

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

$$y = \underbrace{h(x_s, u_s)}_{y_s} + \underbrace{\left.\frac{\partial h}{\partial x^T}\right|_{\substack{x=x_s \\ u=u_s}}}_{=C} \cdot (x - x_s) + \underbrace{\left.\frac{\partial h}{\partial u^T}\right|_{\substack{x=x_s \\ u=u_s}}}_{=D} \cdot (u - u_s)$$

$$\hookrightarrow \quad \underbrace{\Delta y}_{y - y_s} = C\Delta x + D\Delta u$$

Remark : Subsequently, instead of Δx, Δu and Δy, x, u and y are used for brevity.

Example - Linearization of Pendulum Equations

In state-space representation, our system has the form :

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

Case 1: We want to keep the pendulum around $x_s = [\pi/4, 0]^T \quad \rightarrow \quad u_s = \frac{g}{l}\sin(\pi/4)$

$$A^c = \left.\frac{\partial g}{\partial x^T}\right|_{\substack{x=x_s \\ u=u_s}} = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l}\cos(\pi/4) & 0 \end{pmatrix} \qquad B^c = \left.\frac{\partial g}{\partial u^T}\right|_{\substack{x=x_s \\ u=u_s}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$C = \left.\frac{\partial h}{\partial x^T}\right|_{\substack{x=x_s \\ u=u_s}} = \begin{pmatrix} 1 & 0 \end{pmatrix} \qquad D = \left.\frac{\partial h}{\partial u^T}\right|_{\substack{x=x_s \\ u=u_s}} = 0$$

Case 2: We want to keep the pendulum around $x_s = [0\ 0]^T$.
For feedback $u = -x_2$ linearization results in

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & -1 \end{pmatrix} x$$

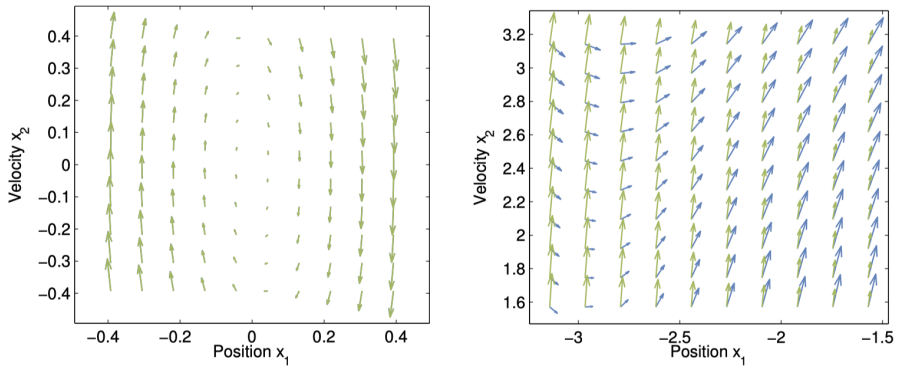


Figure 2.1: Gradient of non-linear system : blue, gradient of linearized system : green

Linearization provides good approximation for small angles and velocities.

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

Discrete time systems are described by difference equations :

$$x(k+1) = g(x(k), u(k))$$
$$y(k) = h(x(k), u(k))$$

with

$x \in \mathbb{R}^n$	State vector
$u \in \mathbb{R}^m$	Input vector
$y \in \mathbb{R}^p$	Output vector
$g(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$	System dynamics
$h(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$	Output function

The inputs and outputs of a discrete time system are defined only at discrete time points, i.e. its inputs and outputs are sequences defined for $k \in \mathbb{Z}^+$. In general, discrete time systems describe either :

- Inherently discrete systems, e.g. bank savings account balance at the k^{th} month

$$x(k+1) = (1+\alpha) \cdot x(k) + u(k)$$

- "Transformed" continuous time systems

The truth is that the vast majority of controlled systems are not inherently discrete time systems, even though modern controllers are almost always implemented using microprocessors. This means that the finite computation time must be considered in the control system design. We will discretize the system.

Discretization is the procedure of obtaining an "equivalent" discrete time system only at particular instances t_k, $k \in \mathbb{Z}^+$ in time, where $t_{k+1} = t_k + T_s$, and T_s is called the sampling time. Usually $u(t) = u(t_k)\, \forall t \in [t_k, t_{k+1}[$ is assumed (and implemented).

Given the continuous time model :

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

We approximate the derivative $\dot{x}^c(t)$:

$$\dot{x}^c(t) \simeq \frac{x^c(t+T_s) - x^c(t)}{T_s}$$

In order to simplify notation, we write :

$$x(k) := x^c(t_0 + kT_s)$$

Example - Euler Discretisation of the Non-Linear Pendulum Equations

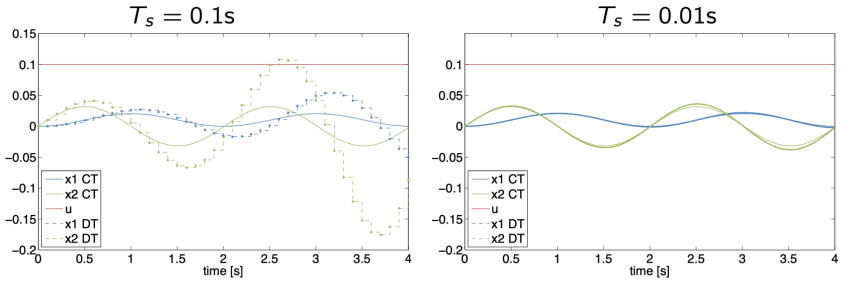
We set $g/l = 10\ \left[s^{-1}\right]$. The non-linear equations are given by :

$$\dot{x} = \begin{pmatrix} x_2 \\ -10 \cdot \sin(x_1) + u \end{pmatrix}$$

Discretising the continuous time system using Euler we get the following discrete time system :

$$x(k+1) = x(k) + T_s \begin{pmatrix} x_2(k) \\ -10 \cdot \sin(x_1(k)) + u(k) \end{pmatrix}$$

Below we can see the different results for two different sampling times :



Therefore we can conclude that with the given continuous-time model :

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

The discrete-time model obtained with Euler discretisation is

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

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

Remark : There are a variety of discretisation approaches (in Matlab we can use `c2d`).

If we recall the solution of the ODE :

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

We choose $t_0 = t_k$ (hence $x_0 = x(t_k)$), $t = t_{k+1}$ and use $t_{k+1} - t_k = T_s$ and $u(t) = u(t_k)$ $\forall t \in [t_k, t_{k+1}[$:

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

We have found the **exact** discrete-time model predicting the state of the continuous-time system at the time t_{k+1} given $x(t_k)$, $k \in \mathbb{Z}_+$ under the assumption of a constant $u(t)$ during a sampling interval.

Remark : $B = (A^c)^{-1}(A - I)B^c$ if A^c is invertible.

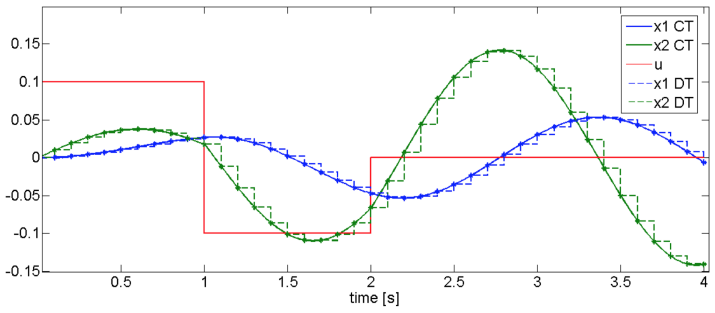
Example - Discretisation of the Linearised Pendulum Equations

Using $g/l = 10\ [s^{-1}]$ the pendulum equations linearised about $x_s = \begin{pmatrix}\pi/4 & 0\end{pmatrix}^T$ are given by :

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -10/\sqrt{2} & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) \tag{2.1}$$

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

$$x(k+1) = \begin{pmatrix} 0.965 & 0.099 \\ -0.699 & 0.965 \end{pmatrix} x(k) + \begin{pmatrix} 0.005 \\ 0.1 \end{pmatrix} u(k)$$



Given :

- The discrete-time linear system $x(k+1) = Ax(k) + Bu(k)$
- An initial state $x(k)$ at discrete-time k
- An input sequence $\{u(k), \ldots, u(k+N-1)\}$

Find the solution of the discrete-time system at time $k+N$:

$$\begin{aligned}
x(k+1) &= Ax(k) + Bu(k) \\
x(k+2) &= Ax(k+1) + Bu(k+1) = A\left(Ax(k) + bu(k)\right) + Bu(k+1) \\
&= A^2x(k) + ABu(k) + Bu(k+1) \\
x(k+3) &= Ax(k+2) + Bu(k+2) \\
&= A^3x(k) + A^2Bu(k) + ABu(k+1) + Bu(k+2) \\
&\vdots \\
x(k+N) &= A^Nx(k) + A^{N-1}Bu(k) + \cdots + ABu(k+N-2) + Bu(k+N-1)
\end{aligned}$$

And therefore :

$$x(k+N) = A^Nx(k) + \sum_{i=0}^{N-1} A^iBu(k+N-1-i)$$

Remark : We have a linear function of the initial state and the input sequence.

In summary, we will use discrete-time models. We will use :

- Non-linear discrete-time models

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

- or linear time-invariant discrete-time models

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

Definition 2.1.1. Discretisation

We call **discretisation** the procedure of obtaining an "equivalent" discrete-time model from a continuous-time model.

2.2 Analysis of LTI Discrete-Time Systems

2.2.1 Coordinate Transformations

If we consider again the system :

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

Remark : Our state-space model is not unique.

The input-output behaviour i.e. the sequence $\{y(k)\}_{k=0,1,2\ldots}$ is entirely defined by $x(0)$ and $\{u(k)\}_{k=0,1,2\ldots}$. There are an infinite number of choices of the state that yield the same input-output behaviour, however certain choices facilitate system analysis.

If we consider the linear transformation $\tilde{x}(k) = Tx(k)$ with $\det(T) \neq 0$ (invertible).

$$\begin{aligned}
T^{-1}\tilde{x}(k+1) &= AT^{-1}\tilde{x}(k) + Bu(k) \\
y(k) &= CT\text{`}{-}1\tilde{x}(k) + Du(k)
\end{aligned}$$

or

$$\tilde{x}(k+1) = \underbrace{TAT^{-1}}_{\tilde{A}}\tilde{x}(k) + \underbrace{TB}_{\tilde{B}}u(k)$$
$$y(k) = \underbrace{CT^{-1}}_{\tilde{C}}\tilde{x}(k) + \underbrace{D}_{\tilde{D}}u(k)$$

Remark. $u(k)$ and $y(k)$ are unchanged.

2.2.2 Stability

Theorem 2.2.1. ***Global Asymptotic Stability***

The LTI system

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

is globally asymptotically stable

$$\lim_{k\to\infty} x(k) = 0 \quad \forall x(0) \in \mathbb{R}^n$$

if and only if $|\lambda_j| < 1$, $\forall j = 1, \ldots, n$ *where* λ_j *are the eigenvalues of* A.

2.2.3 Observability & Controllability

Controllability

Definition 2.2.1. Controllability

A system $x(k+1) = Ax(k) + Bu(k)$ is **controllable** (a.k.a. reachable for discrete systems) if for any pair of states $x(0)$, x^* there exists a finite time N and a control sequence $\{u(0), \ldots, u(N-1)\}$ such that $x(N) = x^*$, i.e.

$$x^* = x(N) = A^N x(0) + \begin{pmatrix} B & AB & \cdots & A^{N-1}B \end{pmatrix} \begin{pmatrix} u(N-1) \\ u(N-2) \\ \vdots \\ u(0) \end{pmatrix}$$

It follows from the **Cayley-Hamilton** theorem that A^k can be expressed as linear combinations of A^j, with $j \in [0, 1, \ldots, n-1]$ for $k \geq n$. Hence for all $N \geq n$:

$$\text{range}\begin{pmatrix} B & AB & \cdots & A^{N-1}B \end{pmatrix} = \text{range}\begin{pmatrix} B & AB & \cdots & A^{n-1}B \end{pmatrix}$$

If the system cannot be controlled to x^* in n steps, then it cannot in an arbitrary number of steps.

We define the controllability matrix C as :

$$C = \begin{pmatrix} B & AB & \cdots & A^{n-1}B \end{pmatrix}$$

The system is controllable if

$$C \begin{pmatrix} u(n-1) \\ u(n-2) \\ \vdots \\ u(0) \end{pmatrix} = x^* - A^n x(0)$$

has a solution for all right-hand sides (RHS). From linear algebra : solution exists for all RHS if and only if n columns of C are linearly independent.

Remarks :

- The necessary and sufficient condition for controllability is $\text{rank}\,(C) = n$.
- Another related concept is stabilisability
- A system is called **stabilisable** if there exists an input sequence that returns the state to the origin asymptotically, starting from an arbitrary initial state.
- A system is stabilisable if and only if all of its uncontrollable modes are stable.
- Stabilisability can be checked using the following condition :

$$\text{if} \quad \text{rank}\,([\lambda_j I - A|B]) = n \quad \forall \lambda_j \in \Lambda_A^+ \quad \rightarrow \quad (A, B) \text{ is stabilisable}$$

  where Λ_A^+ is the set of all eigenvalues of A lying on or outside the unit circle.
- Controllability implies stabilisability

Observability

If we consider the following system with zero input :

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

Definition 2.2.2. Observability

A system is said to be **observable** if there exists a finite N such that for every $x(0)$ the measurements $y(0)$, $y(1)$, $\ldots$, $y(N-1)$ uniquely distinguish the initial state $x(0)$.

The question of uniqueness of the linear equations comes down to :

$$\begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{pmatrix} = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{pmatrix} x(0)$$

As previously we can replace N by n *w log* (Cayley-Hamilton).

We define $\mathcal{O} = \begin{pmatrix} C^T & (CA)^T & \cdots & \left(CA^{n-1}\right)^T \end{pmatrix}^T$.

From linear algebra, the solution is unique if and only if the n columns of $\mathcal{O}$ are linearly independent.

Remarks :

- The necessary and sufficient condition for observability of the system (A, C) is $\text{rank}\,(\mathcal{O}) = n$.
- Another related concept is detectability
- A system is called **detectable** if it possible to construct from the measurement sequence to a sequence of state estimates that converges to the true state asymptotically, starting from an arbitrary initial estimate.
- A system is detectable if and only if all of its unobservable modes are stable

- Detectability can be checked using the following condition

$$\text{if} \quad \text{rank}\left(\left[A^T - \lambda_j I\right] C^T\right) = n \quad \forall \lambda_j \in \Lambda_A^+ \quad \rightarrow \quad (A, C) \quad \text{is detectable}$$

  where Λ_A^+ is the set of all eigenvalues of A lying on or outside the unit circle.

- Observability implies detectability

2.3 Analysis of Non-Linear Discrete-Time Systems

2.3.1 Stability of Non-Linear Systems

Let us consider first the stability of a non-linear, time-invariant, discrete-time system

$$x(k+1) = g\left(x(k)\right) \tag{2.2}$$

with an *equilibrium point* at $\bar{x}$, i.e. $g(\bar{x}) = \bar{x}$.

We know that for non-linear systems there are many definitions of stability. Informally, we define a system to be stable in the sense of Lyapunov, if it stays in any arbitrarily small neighbourhood of the equilibrium when it is disturbed slightly.

In the following we always mean "stability" in the sense of Lyapunov. Note that system 2.2 en compasses any open or closed loop autonomous system. We will then derive simpler stability conditions for the specific case of LTI systems. Note that the stability is a property of an equilibrium point of the system.

Definition 2.3.1. Lyapunov Stability

Formally, the equilibrium point $\bar{x}$ of a system 2.2 is

- **Lyapunov stable** if for every $\epsilon > 0$ there exists a $\delta(\epsilon)$ such that

$$\|x(0) - \bar{x}\| < \delta(\epsilon) \quad \rightarrow \quad \|x(k) - \bar{x}\| < \epsilon, \, \forall k \geq 0$$

- **unstable** otherwise

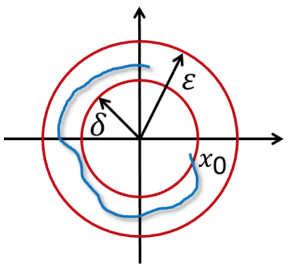


Figure 2.2: If we start within the circle δ, then we stay within the circle ϵ

Definition 2.3.2. Global Asymptotic Stability

An equilibrium point $\bar{x} \in \Omega$ of system 2.2 is

- Asymptotically stable in $\Omega \subseteq \mathbb{R}^n$ if it is Lyapunov stable and **attractive**, i.e.

$$\lim_{k \to \infty} \|x(k) - \bar{x}\| = 0 \quad \forall x(0) \in \Omega$$

- **globally asymptotically stable** if it is asymptotically stable and $\Omega = \mathbb{R}^n$

We can show stability by constructing a **Lyapunov function**. The idea is that a mechanical system is asymptotically stable, if the total mechanical energy is decreasing over time (friction losses). A Lyapunov function is a system theoretic generalisation of energy.

Definition 2.3.3. Lyapunov Function

Consider the equilibrium point $\bar{x} = 0$ of system 2.2. Let $\Omega \subset \mathbb{R}^n$ be a closed and bounded set containing the origin. A function $V : \mathbb{R}^n \to \mathbb{R}$, continuous at the origin, finite for every $x \in \Omega$, such that

$$V(0) = 0 \text{ and } V(x) > 0, \forall x \in \Omega n \{0\}$$
$$V(g(x)) - V(x) \leq -\alpha(x) \quad \forall x \in \Omega$$

where $\alpha : \mathbb{R}^n \to \mathbb{R}$ is continuous positive definite, is called a **Lyapunov function**.

Theorem 2.3.1. ***Lyapunov Stability (Asymptotic Stability)***

If a system 2.2 admits a Lyapunov function $V(x)$, then $x = 0$ is ***asymptotically stable*** *in Ω.*

Note : If $V(x)$ satisfies the condition $V(g(x)) - V(x) \leq -\alpha(x)\ \forall x \in \Omega$ only with $\alpha(x)$ positive semidefinite, then $x = 0$ is **Lyapunov stable** in Ω.

Theorem 2.3.2. ***Lyapunov Stability (Global Asymptotic Stability)***

If a system 2.2 admits a Lyapunov function $V(x)$ for $\Omega = \mathbb{R}^n$, which additionally satisfies :

$$\|x\| \to \infty \quad \Rightarrow \quad V(x) \to \infty$$

then $x = 0$ is ***globally asymptotically stable.***

Theorem 2.3.3. ***Lyapunov Indirect Method***

Let $A = \frac{\partial g}{\partial x^T}\Big|_{x=0}$ be the linearization matrix for system 2.2 around $\bar{x} = 0$. Then ***the origin is locally stable if*** *$|\lambda_i| < 1$* ***for all the eigenvalues of A****. Instead, if there exists at least one eigenvalue such that $|\lambda_i| < 1$, then the origin is unstable.*

- If there exists at least one eigenvalue $|\lambda_i| = 1$, then we cannot say anything about stability. We will need to construct a Lyapunov function.
- The theorem is valid also for continuous-time systems. The eigenvalue conditions will change accordingly :

$$|\lambda_i| < 1 \quad \to \quad \mathrm{Re}(\lambda_i) < 0 \qquad \text{and} \qquad |\lambda_i| > 1 \quad \to \quad \mathrm{Re}(\lambda_i) > 0$$

Remarks :

- Note that the Lyapunov theorems only provide sufficient conditions
- Lyapunov theory is a powerful concept for proving stability of a control system, but for general nonlinear systems it is usually difficult to find a Lyapunov function
- Lyapunov functions can sometimes be derived from physical considerations
- One common approach
  - Decide on **form** of Lyapunov function (e.g. quadratic)
  - Search for parameter values, e.g. via optimization so that the required properties hold
- Later on we'll see that MPC allows for a stability guarantee for constrained systems: Lyapunov function is provided in the form of the optimal cost function (under certain assumptions on the problem setup)
- For linear systems there exist constructive theoretical results on the existence of a quadratic Lyapunov function

2.3.2 Lyapunov Stability of LTI Discrete-Time Systems

Let's consider the system :

$$x\left(k+1\right) = Ax\left(k\right) \tag{2.3}$$

If we take $V(x) = x^t Px$ with $P > 0$ (positive definite) as a candidate Lyapunov function. It satisfies $V(0) = 0$, $V(x) > 0$ and $\|x\| \to \infty \Rightarrow V(x) \to \infty$.

We can also check the "energy decrease" condition :

$$\begin{aligned} V\left(Ax(k)\right) - V\left(x(k)\right) &= x^T(k)A^T PAx(k) - x^T(k)Px(k) \\ &= x^T(k)\left(A^T PA - P\right)x(k) \leq -\alpha\left(x(k)\right) \end{aligned}$$

We can choose $\alpha\left(x(k)\right) = x^T(k)Qx(k)$, $Q > 0$. Hence, the condition can be satisfied if a $P > 0$ can be found that solves the **discrete-time Lyapunov equation** :

$$A^T PA - P = -Q \quad Q > 0 \tag{2.4}$$

Theorem 2.3.4. ***Existence of Solution to the DT Lyapunov Equation***
The discrete-time Lyapunov equation 2.4 has a unique solution $P > 0$ if and only if A has all eigenvalues inside the unit circle, i.e. if and only if the system $x(k+1) = Ax(k)$ is stable.

- Therefore, for LTI systems global asymptotic Lyapunov stability is not only sufficient but also necessary, and it agrees with the notion of stability based on eigenvalue location
- Note that stability is always "global" for linear systems

Property of P

The matrix P can also be used to determine the infinite horizon cost-to-go for an asymptotically stable autonomous system $x(k+1) = Ax(k)$ with a quadratic cost function determined by Q. More precisely, defining $\Psi\left(x(0)\right)$ as

$$\Psi\left(x(0)\right) = \sum_{k=0}^{\infty} x(k)^T Qx(k) = \sum_{k=0}^{\infty} x(0)^T \left(A^k\right)^T QA^k x(0)$$

we have that

$$\Psi\left(x(0)\right) = x(0)^T Px(0)$$

2.3.3 Example - Lyapunov Stability for Linear Systems

Consider the discrete-time system :

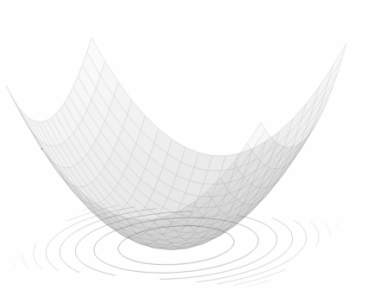
$$x(k+1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u(k)$$

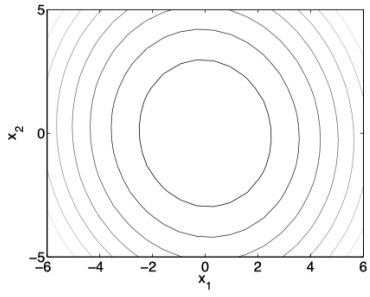
with $u(k) = Kx(k) = -\begin{pmatrix} 0.1160 & 0.5385 \end{pmatrix} x(k)$.

We choose $Q = I$ and solve the discrete-time Lyapunov equation :

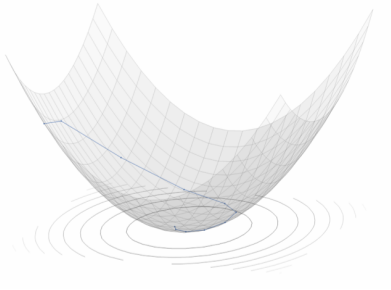
$$\left(A + BK\right)^T P\left(A + BK\right) - P = -Q \quad \Rightarrow \quad P = \begin{pmatrix} 0.3135 & 0.0129 \\ 0.0129 & 0.2257 \end{pmatrix}$$

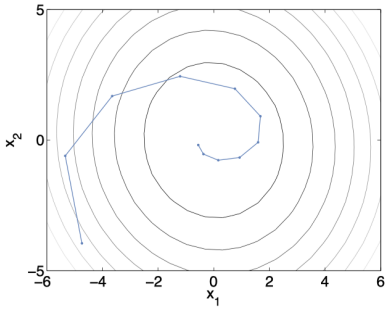
Since $V(x) = x^T Px$ is a quadratic function, all level sets are ellipsoids with shape matrix P :





After close-loop simulating the system we see that the system converges (asymptotically) to the origin.





Summary - System Theory Basics

- Lyapunov stability provides framework to analyze stability of general nonlinear systems
- Asymptotic stability captures convergence in the limit and is commonly used in MPC
- For LTI systems, asymptotic stability can be shown by eigenvalues of the dynamic matrix
- Lyapunov functions provide means to prove stability for nonlinear systems
- The key is how to find a Lyapunov function
- For linear systems there exists a quadratic Lyapunov function if and only if the system is stable
- MPC allows to prove stability by providing a Lyapunov function for the closed-loop system
- Controllability/Observability are necessary requirements for design of a controller/estimator (can be relaxed into stabilizability and detectability)

Chapter 3

Unconstrained Linear Quadratic Optimal Control

3.1 Introduction

Discrete-time **optimal control** in concerned with choosing an optimal input sequence $U \coloneqq \left[u_0^T, u_1^T, \dots\right]$ (as measured by some objective function), over a finite or infinite time horizon, in order to apply it to a system with a given initial state $x(0)$.

The objective, or cost, function is often defined as a sum of **stage costs** $l\left(x_i, u_i\right)$ and, when the horizon has a finite length N, a **terminal cost** $l_f\left(x_N\right)$:

$$J\left(x_0, U\right) \coloneqq l_f\left(x_N\right) + \sum_{i=0}^{N-1} l\left(x_i, u_i\right) \tag{3.1}$$

The states $\{x_i\}_{i=0}^N$ must satisfy the system dynamics :

$$\begin{aligned} x_{i+1} &= g\left(x_i, u_i\right) \quad i = 0, \dots, N-1 \\ x_0 &= x(0) \end{aligned}$$

and there may be state and/or input constraints :

$$h\left(x_i, u_i\right) \leq 0 \quad i = 0, \dots, N-1$$

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

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

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

$$\begin{aligned} J_k^*\left(x\left(0\right)\right) \coloneqq & \min_U J\left(x(0), U\right) \\ \text{subject to} \quad & x_{i+1} = g\left(x_i, u_i\right) \quad i = 0, \dots, N-1 \\ & h(x_i, u_i) \leq 0 \quad i = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned}$$

which is similar to our original MPC problem.

3.1.1 Linear Quadratic Optimal Control

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

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

and **quadratic** cost functions

$$J(x_0, U) \coloneqq x_n^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i\right) \tag{3.2}$$

are considered, and we consider only the problem of regulating the state to the origin, **without state or input constraints**. The two most common approaches that will be described later on are :

1. **Batch Approach**, which yields a series of **numerical values** for the input
2. **Recursive Approach**, which uses dynamic programming to compute control policies or laws, i.e. function that describe how the control decisions depend on the system states.

Our goal is to find a sequence of inputs $U \coloneqq \left[u_0^T, \ldots, u_{N-1}^T\right]^T$ that minimizes the objective function :

$$\begin{aligned} J^*\left(x\left(0\right)\right) \coloneqq & \min_U x_N^T P x_N + \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \quad i = 0, \ldots, N-1 \\ & x_0 = x(0) \end{aligned}$$

In order to simplify the problem for resolution, we have made the following assumptions :

- $P \geq 0$, with $P = P^T$, is the **terminal** weight
- $Q \geq 0$, with $Q = Q^T$, is the **state** weight
- $R > 0$, with $R = R^T$, is the **input** weight (all inputs have a cost)
- N is the horizon cost

Remark Note that $x(0)$ is the current state, whereas x_0, ..., x_N and u_0, ..., u_{N-1} are **optimization variables** that are constrained to obey the system dynamics and the initial condition.

3.1.2 Batch Approach

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

$$\underbrace{\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{pmatrix}}_{S^x} x(0) + \underbrace{\begin{pmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{pmatrix}}_{S^u} \underbrace{\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{N-1} \end{pmatrix}}_{U}$$

The equation above can be represented as

$$X \coloneqq S^x x(0) + S^u U \tag{3.3}$$

We define $\bar{Q}$ and $\bar{R}$ as :

$$\bar{Q} \coloneqq \text{block diag}\,(Q, \ldots, Q, P) \qquad \bar{R} \coloneqq \text{block diag}\,(R, \ldots, R)$$

The finite horizon cost function 3.2 can be written as :

$$J\,(x(0), U) = X^T \bar{Q} X + U^T \bar{R} U \tag{3.4}$$

If we eliminate X by substituting from 3.3, equation 3.4 can be expressed as :

$$\begin{aligned} J\,(x(0), U) &= \left(S^x x(0) + S^u U\right)^T \bar{Q}\left(S^x x(0) + S^u U\right) + U^T \bar{R} U \\ &= U^T H U + 2x(0)^T F U + x(0)^T {S^x}^T \bar{Q} S^x x(0) \end{aligned}$$

where $H \coloneqq \left(S^u\right)^T \bar{Q} S^u + \bar{R}$ and $F \coloneqq \left(S^x\right)^T \bar{Q} S^u$.

Remark : Note that $H > 0$, since $R > 0$ and $\left(S^u\right)^T \bar{Q} S^u \geq 0$.

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

$$\begin{aligned} &\nabla_U J\,(x(0), U) = 2HU + 2F^T x(0) = 0 \\ \hookrightarrow \quad &U^*\,(x(0)) = -H^{-1} F^T x(0) \\ &\qquad = -\left(\left(S^u\right)^T \bar{Q} S^u + \bar{R}\right)^{-1} \left(S^u\right)^T \bar{Q} S^x x(0) \end{aligned}$$

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

Remark : Note that H^{-1} always exists, since $H > 0$ and therefore has full rank.

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

$$\begin{aligned} J^*\,(x(0)) &= -x(0)^T F H^{-1} F^T x(0) + x(0)^T \left(S^x\right)^T \bar{Q} S^x x(0) \\ &= x(0)^T \left[\left(S^x\right)^T \bar{Q} S^x - \left(S^x\right)^T \bar{Q} S^u \left(\left(S^u\right)^T \bar{Q} S^u + \bar{R}\right)^{-1} \left(S^u\right)^T \bar{Q} S^x\right] x(0) \end{aligned}$$

In summary, the batch approach expresses the cost function in terms of the initial state $x(0)$ and input sequence U by eliminating the states x_i. Because the cost $J\,(x(0), U)$ is a positive definite quadratic function of U, its minimizer U^* is unique and can be found by setting $\nabla_u J\,(x(0), U) = 0$. This gives the optimal input sequence U^* as a linear function of the initial state $x(0)$:

$$U^*\,(x(0)) = -\left(\left(S^u\right)^T \bar{Q} S^u + \bar{R}\right)^{-1} \left(S^u\right)^T \bar{Q} S^x x(0)$$

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

$$J^*\,(x(0)) = x(0)^T \left[\left(S^x\right)^T \bar{Q} S^x - \left(S^x\right)^T \bar{Q} S^u \left(\left(S^u\right)^T \bar{Q} S^u + \bar{R}\right)^{-1} \left(S^u\right)^T \bar{Q} S^x\right] x(0)$$

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

3.1.3 Recursive Approach

Alternatively, we can use dynamic programming to solve the same problem in a recursive manner. We define the "j-step optimal cost-to-go" as the **optimal** cost attainable for the step j problem :

$$\begin{aligned} J^*\left(x\left(j\right)\right) &\coloneqq \min_{U_{j\to N}} x_N^T P x_N + \sum_{i=j}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \quad i = j, \ldots, N-1 \\ & x_j = x(j) \end{aligned}$$

This is the minimum cost attainable for the remainder of the horizon after step j.

If we consider the 1-step problem (solved at time $N-1$) :

$$J_{N-1}^*\left(x_{N-1}\right) \coloneqq \min_{u_{N-1}} x_{N-1}^T P x_{N-1} + x_{N-1}^T Q x_{N-1} + u_N^T R u_N \tag{3.5}$$

$$\begin{aligned} \text{subject to} \quad & x_N = Ax_{N-1} + Bu_{N-1} \\ & P_N = P \end{aligned} \tag{3.6}$$

where we introduce the notation P_j to express the optimal cost-to-go $x_j^T P_j x_j$. In particular, $P_N = P$. Substituting equation 3.6 into equation 3.5, we have :

$$J_{N-1}^*\left(x_{N-1}\right) = \min_{u_{N-1}} \left\{x_{N-1}^T \left(A^T P_N A + Q\right) x_{N-1} + u_{N-1}^T \left(B^T P_N B + R\right) u_{N-1} + 2x_{N-1}^T A^T P_N B u_{N-1}\right\}$$

Solving again by setting the gradient to zero leads to the following optimality condition for u_{N-1} :

$$2\left(B^T P_N B + R\right) u_{N-1} + 2B^T P_N A x_{N-1} = 0$$

We have then the **optimal 1-step input** :

$$\begin{aligned} u_{N-1}^* &= -\left(B^T P_N B + R\right)^{-1} B^T P_N A x_{N-1} \\ &\coloneqq F_{N-1} x_{N-1} \end{aligned}$$

and the **1-step cost-to-go** :

$$J_{N-1}^*\left(x_{N-1}\right) = x_{N-1}^T P_{N-1} x_{N-1}$$

where

$$P_{N-1} = A^T P_N A + Q - A^T P_N B \left(B^T P_N B + R\right)^{-1} B^T P_N A$$

Now let's consider the 2-step problem, posed at time $N-2$:

$$\begin{aligned} J_{N-2}^*\left(x_{N-2}\right) &\coloneqq \min_{u_{N-1}, u_{N-2}} \sum_{i=N-2}^{N-1} \left(x_i^T Q x_i + x_i^T R x_i\right) + x_N^T P x_N \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = N-2, N-1 \end{aligned}$$

From the principle of optimality, the cost function is equivalent to :

$$\begin{aligned} J_{N-2}^* &= \min_{u_{N-2}} x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2} + J_{N-1}^*\left(x_{N-1}\right) \\ &= \min_{u_{N-2}} x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2} + + x_{N-1}^T P_{N-1} x_{N-1} \end{aligned}$$

$$\begin{array}{ccccccccc} P_N & \longrightarrow & P_{N-1} & \longrightarrow & P_{N-2} & \longrightarrow & P_{N-3} & \cdots \\ & \searrow & F_{N-1} & \searrow & F_{N-2} & \searrow & F_{N-3} & \cdots \\ & & \downarrow & & \downarrow & & \downarrow & \\ & & u^*_{N-1}(x_{N-1}) & & u^*_{N-2}(x_{N-2}) & & u^*_{N-3}(x_{N-3}) & \cdots \end{array}$$

As with the 1-step solution, we solve the problem by setting the gradient with respect to u_{N-2} to zero. The **optimal 2-step input** is then :

$$u^*_{N-2} = -\left(B^T P_{N-1} B + R\right)^{-1} B^T P_{N-1} A x_{N-2} \coloneqq F_{N-2} x_{N-2}$$

The **2-step cost-to-go** is then :

$$J^*_{N-2}(x_{N-2}) = x^T_{N-2} P_{N-2} x_{N-2}$$

where

$$P_{N-2} = A^T P_{N-1} A + Q - A^T P_{N-1} B \left(B^T P_{N-1} B + R\right)^{-1} B^T P_{N-1} A$$

We now recognize the recursion for P_j and u^*_j, with $j = N-1, \ldots, 0$.

We can obtain the solution for any given time step i in the horizon

$$\begin{aligned} u^*_i &= -\left(B^T P_{i+1} B + R\right)^{-1} B^T P_{i+1} A x(i) \\ &\coloneqq F_i x_i \qquad \text{for} \quad i = 1, \ldots, N \end{aligned}$$

where we can find any P_i by recursive evaluation from $P_N = P$, using the equation :

$$P_i = A^T P_{i+1} A + Q - A^T P_{i+1} B \left(B^T P_{i+1} B + R\right)^{-1} B^T P_{i+1} A \tag{3.7}$$

This is called the **Discrete Time Riccati Equation** or **Riccati Difference Equation (RDE)**. Evaluating down to P_0, we obtain the N-step cost-to-go :

$$J^*(x(0)) = J^*_0(x(0)) = x(0)^T P_0 x(0)$$

The recursive solution method used from here relies on Bellman's **principle of optimality**. For any solution for steps 0 to N to be optimal, any solution for steps j to N with $j \geq 0$, taken from the 0 to N solution, must be itself optimal for the j-to-N problem. Therefore we have, for any $j = 0, \ldots, N$:

$$\begin{aligned} J^*_j(x_j) &\coloneqq \min_{u_j} l(x_i, u_i) + J^*_{j+1}(x_{j+1}) \\ \text{subject to} \qquad & x_{j+1} = A x_j + B u_j \end{aligned}$$

Principle of Optimality - Interpretation

> Suppose that the fastest route from Los Angeles to Boston passes through Chicago. Then the principle of optimality formalizes the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

In summary, from the principle of optimality, the optimal control policy for any step j is then given by :

$$u^*_i = -\left(B^T P_{i+1} B + R\right)^{-1} B^T P_{i+1} A x_i = F - i x_i$$

and the optimal cost-to-go is

$$J^*_i(x_i) = x^T_i P_i x_i$$

Each P_i is related to P_{i+1} by the Riccati Difference equation :

$$P_i = A^T P_{i+1} A + Q - A^T P_{i+1} B \left(B^T P_{i+1} B + R\right)^{-1} B^T P_{i+1} A$$

which can be initialized with $P_N = P$, the given terminal weight.

3.1.4 Comparison of Batch and Recursive Approaches

The fundamental difference between the two approaches is that the batch optimization returns a sequence $U^*(x(0))$ of **numeric values** depending only on the initial state $x(0)$, while dynamic programming yields **feedback policies** $u_i^* = F_i x_i$, $i = 0, \ldots, N-1$ depending on each x_i.

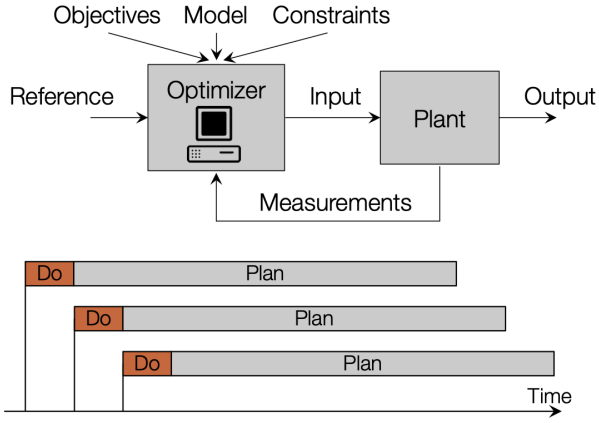
If the state evolves exactly as modelled, then the sequences of control actions obtained from the two approaches are identical. However, **the recursive solution should be more robust to disturbances and model errors**, because if the future states later deviate from their predicted values, the exact optimal input can still be computed.

The recursive approach is computationally more attractive because it breaks the problem down into single-step problems. For a large horizon length, the Hessian H in the batch approach, which must be inverted becomes very large.

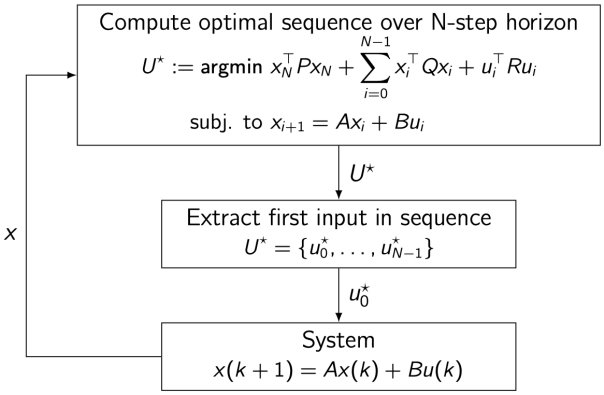
Without any modification, both solution methods will break down when inequality constraints on x_i or u_i are added. The batch approach is far easier to adapt than the recursive approach when constraints are present. We just need to perform a constrained minimization for the current state.

3.2 Receding Horizon

The receding horizon strategy introduces feedback into our control loop.



For unconstrained systems, this is a **constant linear controller**. However, we can extend this concept to much more complex systems (such as MPC).



3.2.1 Example - Impact of Horizon Length

Consider the lightly damped, stable system :

$$G(s) := \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}$$

where $\omega = 1$, $\zeta = 0.01$. We sample at 10 Hz and set $Q = I$ and $R = 1$.

The discrete-time state space model is :

$$x(k+1) = \begin{pmatrix} 1.988 & -0.988 \\ 1 & 0 \end{pmatrix} x(k) + \begin{pmatrix} 0.125 \\ 0 \end{pmatrix} u(k)$$

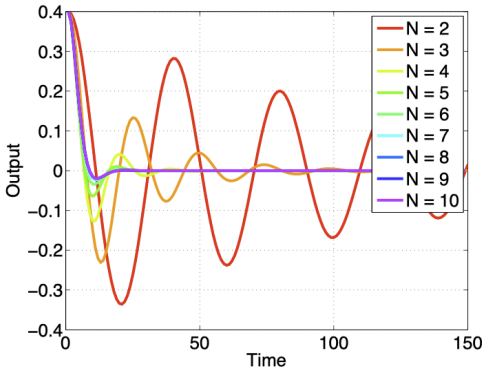


Figure 3.1: The closed-loop response of the system

The shorter the horizon (the smaller N is) we don't predict as far into the future, which is why we oscillate so much for small values of N.

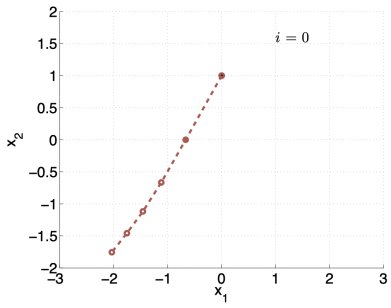


Figure 3.2: $i = 0$

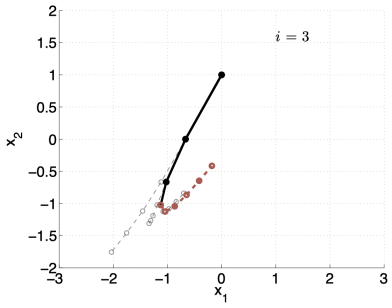


Figure 3.3: $i = 3$

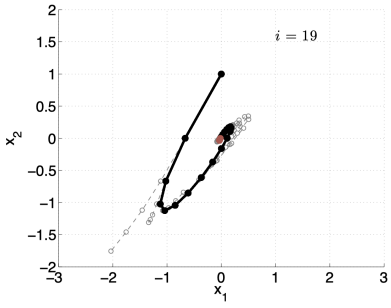


Figure 3.4: $i = 19$

With 20 steps, we are looking ahead enough. Now there is very little change in our predictions :

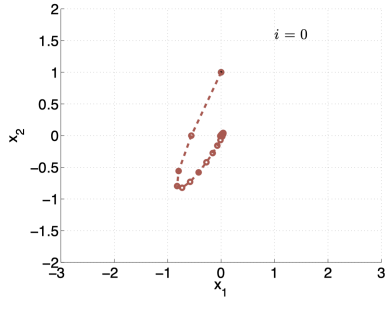


Figure 3.5: $i = 0$

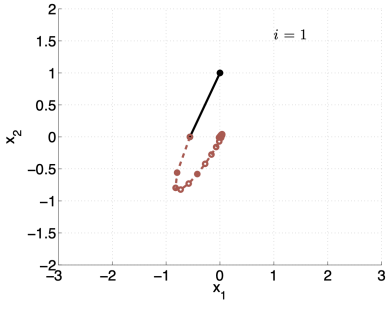


Figure 3.6: $i = 1$

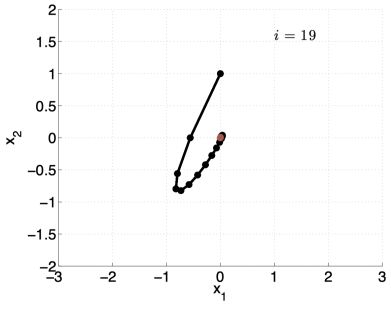


Figure 3.7: $i = 19$

3.2.2 Stability of Finite-Horizon Optimal Control Laws

If we consider the system :

$$G(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}$$

where $\omega = 0.1$ and $\zeta = -1$, which has been discretized at 1 $^{\text{rad}}/_{s}$ (note that the system is unstable). Is the system :

$$x(k+1) = (A + BK_{R,N})\, x(k)$$

stable?

Where $K_{R,N}$ is the finite horizon LQR controller with horizon N and weight R (Q taken to be the identity).

We plot the regions where the system is stable and where it s unstable in Figure 3.8 . We can see that just increasing the horizon length does not necessarily guarantee stability.

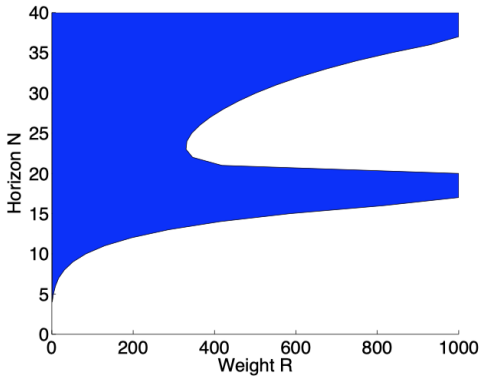


Figure 3.8: Blue : Stable, White : Unstable

3.3 Infinite Horizon LQR

3.3.1 Optimal Solution

In some cases we may want to solve the same problem with an infinite horizon :

$$\begin{aligned} J^*(x(0)) &\coloneqq \min_{u(\cdot)} \sum_{i=0}^{\infty} \left(x_i^T Q x_i + u_i^T R u_i\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, \infty \\ & x_0 = x(0) \end{aligned}$$

As with the dynamic programming approach, the optimal input is of the form

$$u^*(k) = -\left(B^T P_\infty B + R\right)^{-1} B^T P_\infty A x(k) \coloneqq F_\infty x(k)$$

and the infinite-horizon cost-to-go is

$$J_\infty(x(k)) = x(k)^T P_\infty x(k)$$

The matrix P_∞ comes from an infinite recursion of the RDE, from a notional point infinitely far into the future. Assuming the RDE does converge to some constant matrix P_∞, it must satisfy the following (from equation 3.7, with $P_i = P_{i+1} = P_\infty$)

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B \left(B^T P_\infty B + R\right)^{-1} B^T P_\infty A$$

which is called the **Algebraic Riccati Equation (ARE)**.

The constant feedback matrix F_∞ is referred to as the asymptotic form of the **Linear Quadratic Regulator (LQR)**. In fact, if (A, B) is stabilizable and $(Q^{1/2}, A)$ is detectable, then the RDE (initialized with Q at $i = \infty$ and solved for $i \to 0$) converges to the unique positive definite solution P_∞ of the ARE.

3.3.2 Stability of the Infinite Horizon LQR

In addition, the closed-loop system with $u(k) = F_\infty x(k)$ is guaranteed to be asymptotically stable, under the stabilizibility and detectability assumptions. The latter statement can be proven by substituting the control law $u(k) = F_\infty x(k)$ into $x(k+1) = Ax(k) + Bu(k)$, and then examining the properties of the system

$$x(k+1) = (A + BF_\infty)\, x(k)$$

The asymptotic stability of equation 3.3.2 can be proven by showing that the infinite horizon cost $J^*(x(k)) = x(k)^T P_\infty x(k)$ is actually a Lyapunov function for the system, i.e.

1. $J^*(x) > 0 \qquad \forall x \neq 0,\ J^*(0) = 0$
2. $J^*(x) \to \infty$ for $\|x\| \to \infty$
3. $J^*(x(k+1)) < J^*(x(k))$ for any $x(k) \neq 0$

This implies that

$$\lim_{k\to\infty} x(k) = 0$$

Lemma

If the system is stabilizable and detectable, the optimal value function $J^*(x) = x^T P_\infty x$ is a Lyapunov function for the system $x^+ = (A + BF_\infty)\, x$ where $F_\infty = -\left(R + B^T P_\infty B\right)^{-1} B^T P_\infty A$ and P_∞ solves the Algebraic Riccati equation for some $Q \geq 0,\ R > 0$.

$P_\infty > 0$ gives the first two requirements :

$$J^*(x(k)) = x^T(k) P_\infty x(k) = \sum_{i=k}^{\infty} x^T(i) \left(Q + F_\infty^T R F_\infty\right) x(i)$$

Consider the value of $J^*(x(k+1))$:

$$\begin{aligned} J^*(x(k+1)) &= J^*((A + BF_\infty)x(k)) \\ &= \sum_{i=k+1}^{\infty} x^T(i) \left(Q + F_\infty^T R F_\infty\right) x(i) \\ &= J^*(x(k)) - x^T(k) \left(Q + F_\infty^T R F_\infty\right) x(k) < J^*(x(k)) \end{aligned}$$

3.3.3 Choices of Terminal Weight P in Finite Horizon Control

1. The terminal cost P of the infinite horizon problem can in fact trivially be chosen so that its solution matches the infinite horizon solution.
   - To do this, we make P equal to the optimal cost from N to ∞ (i.e. the cost with the optimal controller choice). This can be computed from the ARE :

$$P = A^T P A + Q - A^T P B \left(B^T P B + R\right)^{-1} B^T P A$$

2. Choose P assuming no control action after the end of the horizon, so that

$$x(k+1) = Ax(k) \qquad k = N, \ldots, \infty$$

   - This P can be determined from solving the Lyapunov equation :

$$A^T P A + Q = P$$

   - This approach only makes sense if the system is asymptotically stable (or no positive definite solution P will exist).

3. Assume we want the state and input both to be zero after the end of the finite horizon. In this case no P but an extra constraint is needed.

$$x_{i+N} = 0$$

Summary - Unconstrained Linear Quadratic Optimal Control

Goal : Control law to minimize the relative "energy" of input and state/output.

Why?

- Easy to describe objective / tune controller
- Simple to compute and implement
- Proven and effective

Why infinite-horizon?

- Stable
- Optimal solution (doesn't usually matter)

In MPC we normally cannot have an infinite horizon because it results in an infinite number of optimization variables. Instead, we use tricks to simulate a quasi-infinite horizon.

Chapter 4

Convex Optimisation

4.1 Optimization in MPC

As we know, in MPC we want to do the following three steps at each sample time :

- Measure / estimate the current state $x(k)$
- Find the optimal input sequence for the entire planning window N :

$$U_k^* = \left\{u_k^*, u_{k+1}^*, \ldots, u_{k+N-1}^*\right\}$$

- Implement only the first control action u_k^*

In this chapter we will focus on the second step in this process.

4.1.1 Optimization Problems Arising in MPC

There are several issues that can arise when trying to optimise the inputs for our system. We have categorised them based on what kind of system we are dealing with.

Linear Systems

- Linear system dynamics
- Continuous set of states and inputs, e.g.

$$x \in [x_{\min}, x_{\max}]\,,\; u \in [u_{\min}, u_{\max}]$$

- Example : Chemical processes

Non-linear Systems

- Non-linear system dynamics
- Continuous set of states and inputs, e.g.

$$x \in [x_{\min}, x_{\max}]\,,\; u \in [u_{\min}, u_{\max}]$$

- Example : Kites

Hybrid Knowledge

- Mixed dynamics that are both continuous and discrete, e.g.

$$\begin{cases} x_{k+1} = -c_1 & x_k \geq x_{\max} \\ x_{k+1} = c_2 - c_1 & x_k < x_{\max} \end{cases}$$

- Continuous set of states and inputs
- Example : Walking robot

Discrete Decision Variables

- Inputs and / or states can only take discrete values, e.g.

$$u \in \{1, 2, 3, 4, 5\}$$

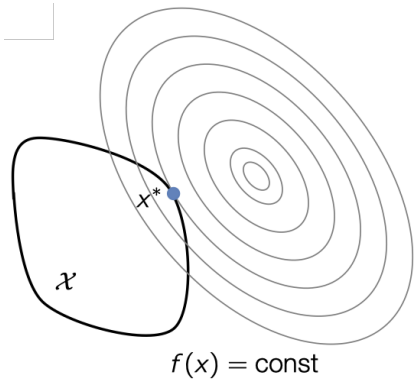
- Example : Internet

4.2 Main Concepts

4.2.1 Mathematical Optimization Problem

A mathematical optimization problem is generally formulated as :

$$\begin{aligned} & \min_{x \in \mathrm{dom}(f)} f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & h_i = 0 \qquad i = 1, \ldots, p \end{aligned}$$



Our optimization variables are $x \coloneqq [x_1, x_2, \ldots, x_n]$, which include our states and our inputs. Our optimization problem also includes :

- Objective function $f : \mathrm{dom}(f) \to \mathbb{R}$
- Domain $\mathrm{dom}(f) \subseteq \mathbb{R}^n$ of the objective function
- Optional inequality constraint functions $g_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, m$
- Optional equality constraint functions $h_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, p$
- $\mathcal{X} \coloneqq \{x \in \mathrm{dom}(f) | g_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p\}$: set of feasible decisions, or feasible set.

4.2.2 Terminology

We define the following terminology that we will be using.

Feasible Point
$x \in \mathrm{dom}(f)$ satisfying the inequality and equality constraints, i.e. $g_i(x) \leq 0$, for $i = 1, \ldots, m$, $h_i(x) = 0$ for $i = 1, \ldots, p$.

Strictly Feasible Point
Feasible $x \in \mathrm{dom}(f)$ satisfying the inequality constraints strictly, i.e. $g_i(x) < 0$ for $i = 1, \ldots, m$.

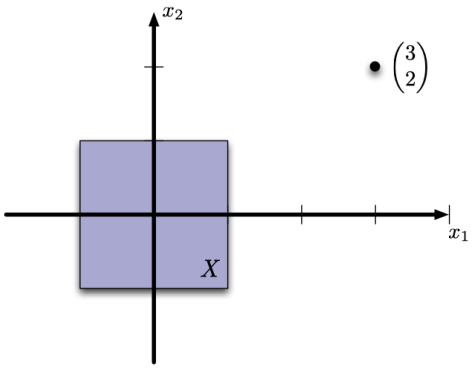
Optimal Value
Lowest possible cost value $p^* = f(x^*) \triangleq \min_{x \in \mathcal{X}} f(x)$ also denoted by f^* or J^*.

Optimizer
Any feasible x^* that achieves smallest cost p^*, i.e., $x^* \in \mathcal{X}$ with $f(x^*) \leq f(x)$ for all feasible $x \in \mathcal{X}$.
It is worth noting that the optimizer is not always unique. The set of solutions is :

$$\arg\min_{x \in \mathcal{X}} f(x) \coloneqq \{x \in \mathcal{X} | f(x) = p^*\}$$

4.2.3 Example - Simple Optimization Problem

In $\mathbb{R}^2$, find the point in the unit box X closest to the point $(x_1, x_2) = (3, 2)$.



We can write the same problem in the standard format :

$$\begin{aligned} & \min_{(x_1,x_2)\in\mathbb{R}^2} (x_1-3)^2 + (x_2-2)^2 \\ \text{subject to} \quad & x_1 \leq 1 \\ & -x_1 \leq 1 \\ & x_2 \leq 1 \\ & -x_2 \leq 1 \end{aligned}$$

Obviously the optimal solution in this problem is $x^* = (1, 1)$, which we determined graphically.

4.2.4 Active, Inactive & Redundant Constraints

If we consider the standard problem :

$$\begin{aligned} & \min_{x\in\mathrm{dom}(f)} f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & h_i = 0 \qquad i = 1, \ldots, p \end{aligned}$$

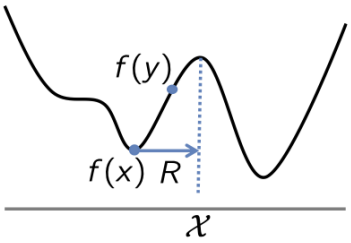
We can make the following remarks :

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

4.2.5 Optimality

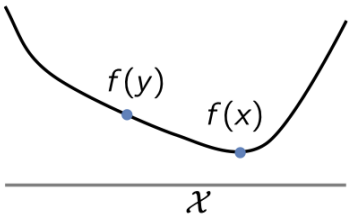
We say that $x \in \mathcal{X}$ is **locally optimal**if, for some $R > 0$, it satisfies

$$y \in \mathcal{X},\ \|y - x\| \leq R \Rightarrow f(y) \geq f(x)$$



We say that $x \in \mathcal{X}$ is **globally optimal** fi it satisfies

$$y \in \mathcal{X} \Rightarrow f(y) \geq f(x)$$



If $p^* = -\infty$ the problem is **unbounded below**. If $\mathcal{X}$ is empty, then the problem is said to be **infeasible** (convention : $p^* = \infty$). If $\mathcal{X} = \mathbb{R}^n$ the problem is said to be unconstrained.

4.2.6 "Easy" & "Hard" Problems

"Easy" : Linear Program (LP)
Linear cost and constraint functions :

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

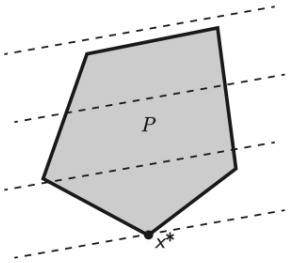


Figure 4.1: Linear optimization on a polytope

"Hard" : Mixed Integer Linear Program
Linear cost and constraint functions :

$$\begin{aligned} & \min_x c^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \\ & x \in \{0,1\}^n \quad \text{or} \quad x \in \mathbb{Z}^n \end{aligned}$$

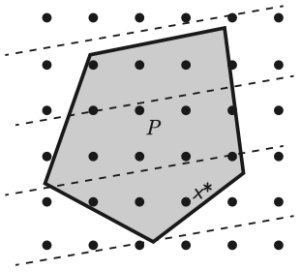


Figure 4.2: Linear optimization with integer constraints (dots)

Convex optimization problems can be solved efficiently and reliably.

4.2.7 Software Tolls for Optimization

A simple optimization problem :

$$\begin{aligned} & \min_{x_1,x_2} |x_1 + 5| + |x_2 - 3| \\ \text{subject to} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

This problem is equivalent to a linear program. A huge variety of software tools for solving standard optimization problem exist, such as CPLEX, Gurobi, GLPK, XPRESS, qpOASES, OOQP, FORCES, IPOPT...

There is no standard interface to solvers - they are almost all different. We will be using the Yalmip toolbox in Matlab. We can see an example in a simple optimisation problem :

$$\begin{aligned} & \min_{x_1,x_2} |x_1 + 5| + |x_2 - 3| \\ \text{subject to} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

Using the Yalmip toolbox, we would write :

```
% make variables
sdpvar x1 x2;

% define cost function
f = abs(x1 + 5) + abs(x2 - 3);

% define constraints
X = set(2.5 <= x1 <= 5) + ...
set( -1 <= x2 <= 1);

% solve
solvesdp(X,f);
```

4.3 Convex Sets

Definition 4.3.1. Convex Sets

A set $\mathcal{X}$ is **convex** if and only if for any pair of points x and y in $\mathcal{X}$:

$$\lambda x + (1 - \lambda)\, y \in \mathcal{X},\ \forall \lambda \in [0, 1]\,,\ \forall x, y \in \mathcal{X}$$

Interpretation : All line segments starting and ending in $\mathcal{X}$ stay within $\mathcal{X}$.

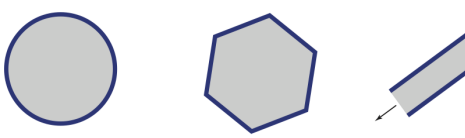


Figure 4.3: Examples of convex sets

Figure 4.4: Examples of non-convex sets

A **convex combination** of $x_1, \ldots, x_k$ is any point x of the form :

$$x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k \quad \text{with} \quad \theta_1 + \cdots + \theta_k = 1,\ \theta_i \geq 0$$

Definition 4.3.2. Hyperplane

A **hyperplane** is defined by $\{x \in \mathbb{R}^n | a^T x = b\}$ for $a \neq 0$, where $a \in \mathbb{R}^n$ is the normal vector to the hyperplane.

Definition 4.3.3. Half-Spaces

A **half-space** is everything on one side of a hyperplane $\{x \in \mathbb{R}^n | a^T x \leq b\}$ for $a \neq 0$. It can either be **open** (strict inequality) or **closed** (non-strict inequality).

For $n = 2$, hyperplanes define lines. For $n = 3$, hyperplanes define planes. Hyperplanes are affine and convex, half-spaces are convex.

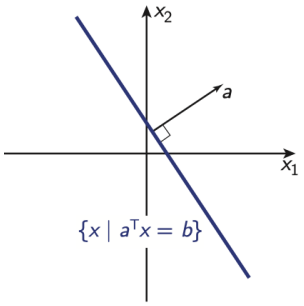


Figure 4.5: A hyperplane

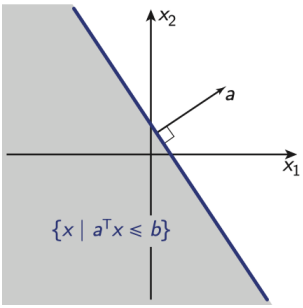


Figure 4.6: A closed half-space

Definition 4.3.4. Polyhedrons

A **polyhedron** is the intersection of a finite number os closed half-spaces :

$$P \coloneqq \{x | a_i^T x \leq b_i,\ i = 1, \ldots, n\} = \{x | Ax \leq b\}$$

where $A \coloneqq [a_1, a_2, \ldots, a_m]^T$ and $b \coloneqq [b_1, b_2, \ldots, b_m]^T$.

Definition 4.3.5. Polytopes

A **polytope** is a bounded polyhedron.

Polyhedra and polytopes are always convex.

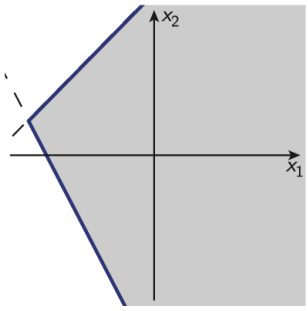


Figure 4.7: An (unbounded) polyhedron

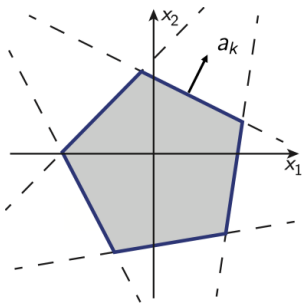


Figure 4.8: A polytope

Definition 4.3.6. Ellipsoids

An **ellipsoid** is a set defined as

$$\left\{x | \left(x - x_c\right)^T A^{-1} \left(x - x_c\right) \leq 1\right\}$$

where x_c is the centre of the ellipsoid, and $A > 0$ (i.e. A is positive definite).

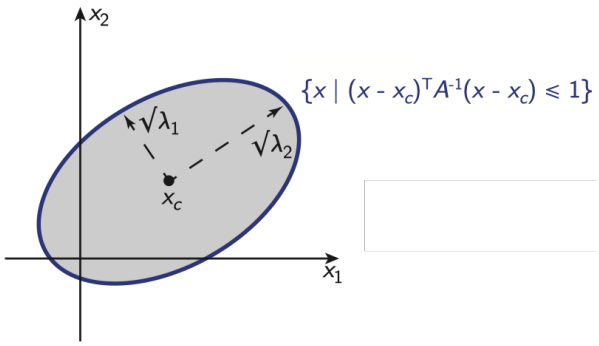


Figure 4.9: Semi-axis lengths are square roots of eigenvalues of A

The **Euclidian Ball** $B(x_c, r)$ is a special case of the ellipsoid, for which $A = r^2 I$, so that $B(x_c, r) \coloneqq \{x | \|x - x_c\|_2 \leq r\}$.

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

By far the most common l_p norms are :

- $p = 2$ - Euclidian norm :

$$\|x\|_2 = \sqrt{\sum_i x_i^2}$$

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

$$\|x\|_1 = \sum_i |x_i|$$

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

$$\|x\|_\infty = \max_i |x_i|$$

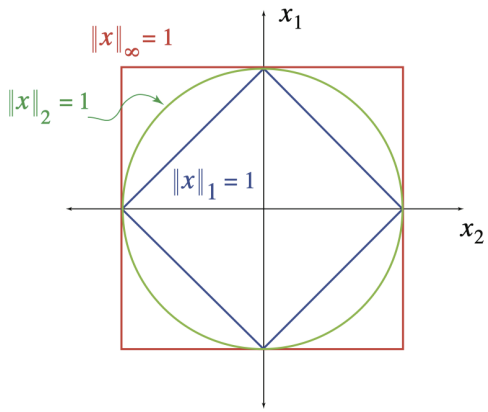


Figure 4.10

Theorem 4.3.1. ***Intersection of Convex Sets***

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

4.4 Convex Functions

Definition 4.4.1. Convex Functions

A function $f : \text{dom}(f) \to \mathbb{R}$ is **convex** if and only if $\text{dom}(f)$ is convex and

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda)\, f(y) \qquad \forall x, y \in \text{dom}(f)$$

The function $f : \text{dom}(f) \to \mathbb{R}$ is **strictly convex** if this inequality is strict.

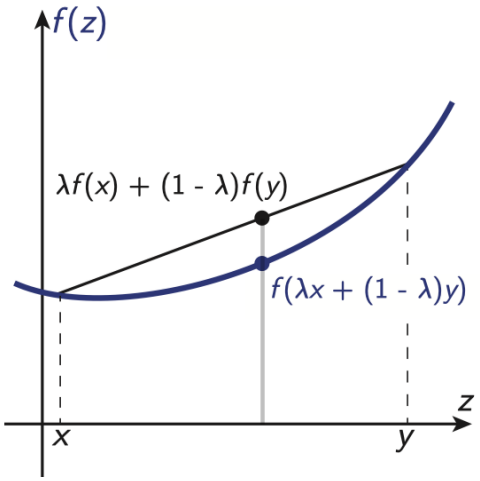


Figure 4.11

Definition 4.4.2. Concave Functions

The function f is **concave** if and only if $\text{dom}(f)$ is convex and $-f$ is convex.

4.4.1 First-Order Condition for Convexity

A differentiable function $f : \text{dom}(f) \to \mathbb{R}$ with a convex domain is **convex** if and only if :

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

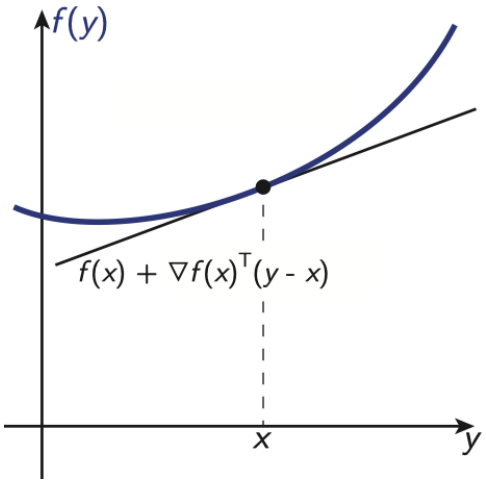


Figure 4.12

The first-order approximation of f around any point x is a global under-estimator of f. The gradient $\nabla f(x)$ is given by :

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}\right]^T$$

A twice-differentiable function $f : \text{dom}(f) \to \mathbb{R}$ with a convex domain $\text{dom}(f)$ is **convex** if and only if :

$$\nabla^2 f(x) \geq 0 \qquad \forall x \in \text{dom}(f)$$

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

$$\nabla^2 f(x)_{ij} = \frac{\partial f(x)}{\partial x_i \partial x_j}$$

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

4.4.2 Level & Sub-level Sets

Definition 4.4.3. Level Sets

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

$$L_\alpha \coloneqq \{x | x \in \text{dom}(f), f(x) = \alpha\}$$

For $f : \mathbb{R}^2 \to \mathbb{R}$ these are **contour lines** of constant "height".

Definition 4.4.4. Sub-Level Sets

The **sub-level set** C_α of a function f for value α is defined by

$$C_\alpha \coloneqq \{x | x \in \text{dom}(f), f(x) \leq \alpha\}$$

If the function f is convex $\Rightarrow$ sub-level sets of f are convex for all α. But not $\Leftarrow$!

4.4.3 Examples of Convex Functions : $\mathbb{R} \to \mathbb{R}$

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

- Affine : $ax + b$ for any $a, b \in \mathbb{R}$
- Exponential : e^{ax} for any $a \in \mathbb{R}$
- Powers : x^α on domain $\mathbb{R}_{++}$, for $\alpha \geq 1$ or $\alpha \leq 0$
- Vector norms on $\mathbb{R}^n$: $\|x\|_p = (\sum_{i=1}^n |x|^p)^{1/p}$, for $p \geq 1$, $\|x\|_\infty = \max_i |x|_i$

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

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

Certain operations preserve the convexity of functions :

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

and many other possibilities ...

4.5 Convex Optimization Problems

A **convex optimization problem** can be written in standard form :

$$\begin{aligned} & \min_{x \in \mathrm{dom}(f)} f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & a_i^T x = b_i \qquad i = 1, \ldots, p \end{aligned}$$

- $f, g_1, \ldots, g_m$ are convex functions
- $\mathrm{dom}(f)$ is a convex set
- Equality constraint functions $h_i(x) = a_i^T x - b$ are all affine

The affine constraints are typically gathered into matrix form :

$$\begin{aligned} & \min_{x \in \mathrm{dom}(f)} f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & Ax = b \qquad A \in \mathbb{R}^{p \times m} \end{aligned}$$

Important Property : A feasible set of a convex optimization problem is convex.

4.5.1 Local & Global Optimality for Convex Problems

Theorem 4.5.1. ***Optimal Solutions in Convex Problems***
For a convex optimization problem, ***any*** *locally optimal solution is globally optimal (local optima are global optima).*

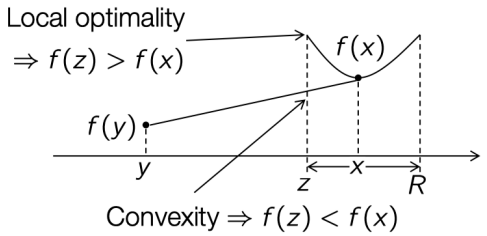
Proof.

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

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

- The problem therefore cannot be convex

□



4.5.2 Equivalent Optimization Problems

Two problems are (informally) called **equivalent** if the solution to one can be (easily) inferred from the solution to the other, and vice versa.

We introduce the principle of **equality constraints** :

$$\begin{aligned} & \min_{x} f(A_0 x + b_0) \\ \text{subject to} \quad & g_i(A_i x + b_i) \leq 0 \qquad i = 1, \ldots, m \end{aligned}$$

is equivalent to

$$\begin{aligned} & \min_{x_i, y_i} f(y_0) \\ \text{subject to} \quad & g_i(y_i) \leq 0 \qquad i = 1, \ldots, m \\ & A_i x + b_i = y_i \qquad i = 0, 1, \ldots, m \end{aligned}$$

Depending on the solver one of the two problems may be better than the other.

We also introduce the principle of **slack constraints** :

$$\begin{aligned} & \min_{x} f(x) \\ \text{subject to} \quad & A_i x \leq b_i \qquad i = 1, \ldots, m \end{aligned}$$

is equivalent to

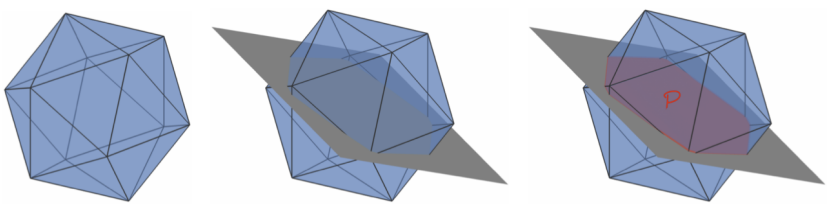
$$\begin{aligned} & \min_{x, s_i} f(x) \\ \text{subject to} \quad & A_i x + s_i \leq b_i \qquad i = 1, \ldots, m \\ & s_i \geq 0 \qquad i = 1, \ldots, m \end{aligned}$$

4.5.3 Linear Programs

Say we have the optimization problem below :

$$\begin{aligned} & \min_{x} f(x) \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

We can visualize the constraints of the optimization problem below :



(a) $Gx \leq h$ (b) $A_i^\top x = b_i$ (c) $Gx \leq h \cap A_i^\top x = b_i$

We can see that the constraint $Ax = b$ is a hyperplane, and the feasible set of the problem is P (shown in red). If P is empty, then the problem is infeasible.

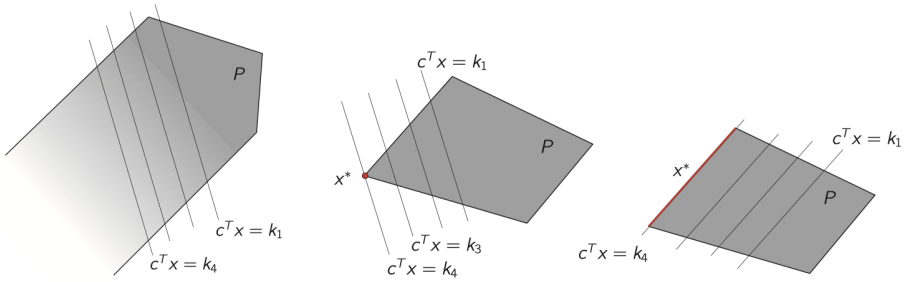
Graphical Interpretation & Solution Properties

We denote by p^* the optimal value and by X_{opt} the set of optimizers. We have a total of three possibilities for our solution :

Case 1: The LP solution is unbounded, i.e., $p^* = -\infty$

Case 2: The LP solution is bounded, i.e., $p^* > -\infty$ and the optimizer is unique. X_{opt} is a singleton.

Case 3: The LP solution is bounded and there are multiple optima. X_{opt} is a subset of $\mathbb{R}^s$, which can be bounded or unbounded.



(a) Case 1 (b) Case 2 (c) Case 3

4.5.4 Quadratic Programs

We can write the general quadratic program as :

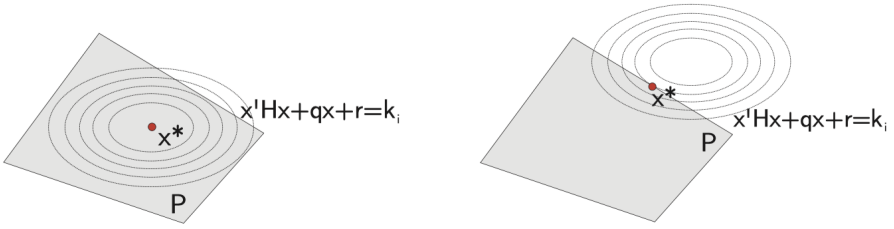
$$\begin{aligned} & \min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x + q^T x + r \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

- The constant r can be left out, since it has no effect on the optimal solution.
- The problem is convex if $H > 0$.
- The problems with concave objective $H \not\geq 0$ are quadratic programs, but difficult.

Two cases can occur if the feasible set P is not empty :

Case 1: The optimizer lies strictly inside the feasible polyhedron

Case 2: The optimizer lies on the boundary of the feasible polyhedron



4.6 Optimality Conditions

4.6.1 The Lagrange Dual Problem

Let's start by recalling our standard (possibly non-convex) optimization problem :

$$\begin{aligned} & \min_{x \in \mathrm{dom}(f)} f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & h_i(x) = 0 \qquad i = 1, \ldots, p \end{aligned}$$

with (primal) decision variable x, domain $\mathrm{dom}(f)$ and optimal value p^*.

The **Lagrangien function** is $L : \mathrm{dom}(f) \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

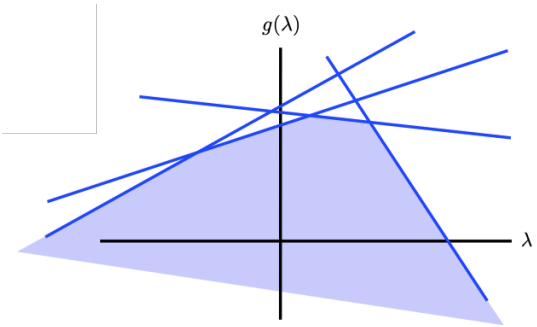
- The dual function $d(\lambda, \nu)$ is always a **concave** function (point-wise infimum of affine functions).
- The dual function generates lower bounds for p^* :

$$d(\lambda, \nu) \leq p^* \qquad \forall \, (\lambda \leq 0, \nu \in \mathbb{R}^p)$$

- $d(\lambda, \nu)$ might be $-\infty$:

$$\mathrm{dom}(d) \coloneqq \{\lambda, \nu | d(\lambda, \nu) > -\infty\}$$

We move our constraints into L, with the weights λ_i and ν_i.



4.6.2 The Primal & Dual Problem

A general optimization problem and its dual are listed below :

$$\begin{aligned} & \min_{x} f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & h_i(x) = 0 \qquad i = 1, \ldots, p \end{aligned} \qquad\qquad \begin{aligned} & \max_{\nu, \lambda} d(\nu, \lambda) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned}$$

Remarks :

- The dual problem is convex even if the primal problem is not.
- The dual problem has an optimal value $d^* \leq p^*$.
- The point (λ, ν) is **dual feasible** if $\lambda \geq 0$ and $(\lambda, \nu) \in \mathrm{dom}(d)$.
- We can often impose the constraint $(\lambda, \nu) \in \mathrm{dom}(d)$ explicity in the dual problem.

4.6.3 Example - Dual of a Linear Program

We have the following primal problem :

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} c^T x \\ \text{subject to} \quad & Ax = b \\ & Cx \leq e \end{aligned}$$

The dual function is :

$$\begin{aligned} d(\lambda,\nu) &= \min_{x\in\mathbb{R}^n} \overbrace{\left[c^T x + \nu^T(Ax-b) + \lambda^T(Cx-e)\right]}^{\text{Lagrangien function}} \\ &= \min_{x\in\mathbb{R}^n} \left[(A^T\nu + C^T\lambda + c)^T x - b^T\nu - e^T\lambda\right] \\ &= \begin{cases} -b^T\nu - e^T\lambda & \text{if } A^T\nu + C^T\lambda + c = 0 \\ -\infty & \text{otherwise} \end{cases} \end{aligned}$$

Lower bound property :

$$-b^T\nu - e^T\lambda \leq p^* \quad \text{whenever} \quad A^T\nu + C^T\lambda + c = 0 \quad \text{and} \quad \lambda \geq 0$$

The **dual problem** is :

$$\begin{aligned} & \max_{\lambda,\nu} -b^T\nu - e^T\lambda \\ \text{subject to} \quad & A^T\nu + C^T\lambda + c = 0 \\ & \lambda \geq 0 \end{aligned}$$

Remark : The dual of a linear program is also a linear program.

4.6.4 Example - Dual of a Quadratic Program

A **quadratic program** (QP) with $Q > 0$:

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} \frac{1}{2}x^T Qx + c^T x \\ \text{subject to} \quad & Cx \leq e \end{aligned}$$

The **dual function** is

$$\begin{aligned} d(\lambda) &= \min_{x\in\mathbb{R}^n} \left[\frac{1}{2}x^T Qx + c^T x + \lambda^T(Cx-e)\right] \\ &= \min_{x\in\mathbb{R}^n} \left[\frac{1}{2}x^T Qx + (c + C^T\lambda)^T x - e^T\lambda\right] \end{aligned}$$

The unconstrained minimization over x is convex for every λ. If $Q > 0$, then the optimal x satisfies

$$Qx + c + C^T\lambda = 0$$

If we substitute $x = -Q^{-1}(c + C^T\lambda)$ into the dual function :

$$d(\lambda) = -\frac{1}{2}\left(c + C^T\lambda\right)^T Q^{-1}\left(c + C^T\lambda\right) - e^T\lambda$$

The dual problem of the quadratic problem :

$$\begin{aligned} & \min_{\lambda} \frac{1}{2}\lambda^T C^T Q^{-1} C\lambda + \left(CQ^{-1}c + e\right)^T \lambda + \frac{1}{2}c^T Q^{-1} c \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned}$$

Remark : The dual of a QP is another QP.

4.6.5 Weak & Strong Duality

Weak Duality
With weak duality, it is always true that $d^* \leq p^*$.

Strong Duality

- It is **sometimes** true that $d^* = p^*$
- Strong duality usually does not hold for non-convex problems
- We can impose conditions on convex problems to guarantee that $d^* = p^*$
- Sometimes the dual is much easier to solve than the primal (or vice-versa)

Example : The dual of a mixed integer linear program (difficult to solve) is a standard LP (easy to solve).

Strong Duality for Convex Problems

An optimization problem with f and all g_i convex :

$$\begin{aligned} & \min f(x) & \\ \text{subject to} \quad & g_i(x) \leq 0 & i = 1, \ldots, m \\ & Ax = b & A \in \mathbb{R}^{p \times n} \end{aligned}$$

Slater Condition
If there is at least one **strictly feasible point**, i.e.

$$\{x | Ax = b, g_i(x) < 0, \forall i \in \{1, \ldots, m\}\} \neq \emptyset$$

Then $p^* = d^*$.

Other **constraint qualification** conditions can also be used to check strong duality in convex problems.

4.6.6 Optimality Conditions

Karush-Kuhn-Tucker Conditions

We assume that all g_i and h_i are differentiable. **Necessary** conditions for optimality :

1. Primal feasibility :

$$\begin{aligned} g_i(x^*) &\leq 0 \qquad i = 1, \ldots, m \\ h_i(x^*) &= 0 \qquad i = 1, \ldots, p \end{aligned}$$

2. Dual feasibility :

$$\lambda^* \geq 0$$

3. Complementary slackness :

$$\lambda_i^* g_i(x^*) = 0 \qquad i = 1, \ldots, m$$

4. Stationarity :

$$\nabla_x L\left(x^*, \lambda^*, \nu^*\right) = \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0$$

Notes on Complimentary Slackness

If we assume that strong duality holds, with an optimal solution x^* and (λ^*, ν^*).

1. From strong duality, $d^* = p^* \quad \rightarrow \quad d(\lambda^*, \nu^*) = f(x^*)$
2. From the definition of the dual function :

$$\begin{aligned} f(x^*) = d(\lambda^*, \nu^*) &= \min_x \left\{ f(x) + \sum_{i=1}^m \lambda_i^* g_i(x) + \sum_{i=1}^p \nu_i^* h_i(x) \right\} \\ &\leq f(x^*) + \sum_{i=1}^m \lambda_i^* g_i(x^*) + \sum_{i=1}^p \nu_i^* h_i(x^*) \leq f(x^*) \end{aligned}$$

$$\hookrightarrow \quad f(x^*) = d(\lambda^*, \nu^*) = f(x^*) + \underbrace{\sum_{i=1}^m \lambda_i^* g_i(x^*)}_{\leq 0} + \sum_{i=1}^p \nu_i^* h_i(x^*)$$

3. Complimentary slackness is made up of two conditions :

$$\begin{aligned} \lambda_i^* &= 0 && \text{for every } g_i(x^*) < 0 \\ g_i(x^*) &= 0 && \text{for every } \lambda_i^* > 0 \end{aligned}$$

KKT Conditions

For a general optimization problem, the necessary condition is :

> If x^* and (λ^*, ν^*) are primal and dual solutions, with zero duality gap, then x^* and (λ^*, ν^*) satisfy the KKT conditions.

For a convex optimization problem, the necessary and sufficient conditions are :

> If x^* and (λ^*, ν^*) satisfy the KKT conditions, then x^* and (λ^*, ν^*) are primal and dual optimal solutions.
>
> If Slater's conditions holds (i.e. strong duality holds), x^* and (λ^*, ν^*) are primal and dual optimal solutions if and only if they satisfy the KKT conditions.

Remark : KKT Conditions for Convex Problems

For a convex optimization problem, the KKT conditions are sufficient :
If (x^*, λ^*, ν^*) satisfy the KKT conditions, then $p^* = d^*$.

- $p^* = f(x^*) = L(x^*, \lambda^*, \nu^*)$ (due to complimentary slackness)
- $d^* = d(\lambda^*, \nu^*) = L(x^*, \lambda^*, \nu^*)$ (due to convexity of the functions and stationarity)

4.6.7 Example - KKT Conditions for a QP

If we consider a (convex) quadratic program with $Q \geq 0$:

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Q x + c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

The **Lagrangien** is : $L(x, \lambda, \nu) = \frac{1}{2}x^T Qx + c^T x + \nu^T (Ax - b) - \lambda^T x$

The KKT conditions are :

$$\begin{aligned} \nabla_x L(x, \lambda, \nu) = Qx + A^T\nu - \lambda + c = 0 \quad & \text{(stationarity)} \\ Ax = b \quad & \text{(primal feasibility)} \\ x \geq 0 \quad & \text{(primal feasibility)} \\ \lambda \geq 0 \quad & \text{(dual feasibility)} \\ x_i \lambda_i = 0 \qquad i = 1, \ldots, n \quad & \text{(complimentarity)} \end{aligned}$$

The final three conditions are often written together as $0 \leq x \perp \lambda \geq 0$.

4.6.8 Sensitivity Analysis

A general optimization problem and its dual are written below :

$$\begin{array}{lll} & \min\limits_x f(x) & \\ \text{subject to} & g_i(x) \leq 0 & i = 1, \ldots, m \\ & h_i(x) = 0 & i = 1, \ldots, p \end{array} \qquad \begin{array}{ll} & \max\limits_{\nu,\lambda} d(\nu, \lambda) \\ \text{subject to} & \lambda \geq 0 \end{array}$$

A perturbed optimization problem and its dual are written below :

$$\begin{array}{lll} & \min\limits_x f(x) & \\ \text{subject to} & g_i(x) \leq u_i & i = 1, \ldots, m \\ & h_i(x) = v_i & i = 1, \ldots, p \end{array} \qquad \begin{array}{ll} & \max\limits_{\nu,\lambda} d(\nu, \lambda) - u^T\lambda - v^T\nu \\ \text{subject to} & \lambda \geq 0 \end{array}$$

- x is the primal decision variable
- (λ, ν) are the dual decision variables
- u and v are parameters representing perturbations to the constraints
- $p^*(u, v)$ is the optimal value as a function of (u, v)

Sensitivity & Lagrange Multipliers

We assume strong duality for the perturbed problem with (ν^*, λ^*) as the dual optimal. Weak duality for the perturbed problem implies :

$$\begin{aligned} p^*(u, v) &\geq d^*(\nu^*, \lambda^*) - u^T\lambda^* - v^T\nu^* \\ &= p^*(0, 0) - u^T\lambda^* - v^T\nu^* \end{aligned}$$

We don't know if strong duality holds for the perturbed problem.

Global Sensitivity Analysis

λ_i^* large and $u_i < 0$	$\Rightarrow$	$p^*(u,v)$ increases greatly
λ_i^* small and $u_i > 0$	$\Rightarrow$	$p^*(u,v)$ does not increase that much
ν^* large and positive and $v_i < 0$ ν^* large and negative and $v_i > 0$	$\Rightarrow$	$p^*(u,v)$ increases greatly
ν^* small and positive and $v_i > 0$ ν^* small and negative and $v_i < 0$	$\Rightarrow$	$p^*(u,v)$ does not increase that much

Remark : These results are <u>not</u> symmetrical. We only have a lower bound on $p^*(u,v)$.

Local Sensitivity Analysis

If in addition $p^*(u,v)$ is differentiable at $(0,0)$ then,

$$\lambda_i^* = -\frac{\partial p^*(0,0)}{\partial u_i} \qquad\qquad \nu_i = -\frac{\partial p^*(0,0)}{\partial v_i}$$

- λ_i^* is the sensitivity of p^* relative to the i^{th} inequality
- ν_i^* is the sensitivity of p^* relative to the i^{th} equality

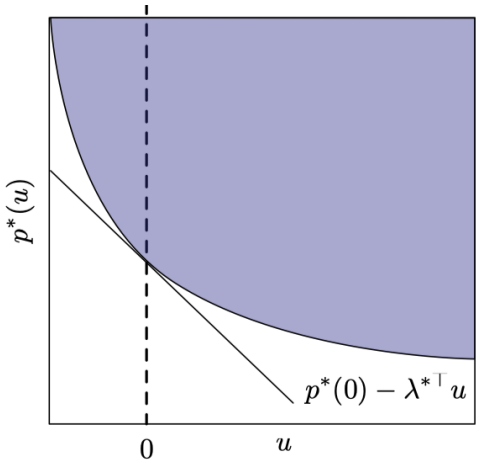


Figure 4.13

Summary - Convex Optimization

- Convex optimization problem :
  - Convex cost function
  - Convex inequality constraints
  - Affine equality constraints
- Benefit of convex problems : Local = Global Optimality
- We only need to find one minimum, which is the global minimum!

- For convex optimization problem : If Slater's conditions holds, x^* is optimal if and only if $\exists(\lambda^*, \nu^*)$ satisfying the KKT conditions
- Convex optimization problems can be solved efficiently
- Many problems can be written as convex optimization problems (with some effort)

Remark : Duality and optimality conditions similarly extend to Convex Cone Programs

Summary - Why did we need the dual problem?

- The dual problem is convex, even if the primal is not $\rightarrow$ it can be "easier" to solve then the primal problem
- The dual problem provides a lower bound for the primal problem :

  $$d^* \leq p^* \qquad \text{(and } d(\lambda, \nu) \leq p(x) \text{ for all feasible } x, \lambda, \nu)$$

  (provides sub-optimality bound)
- The dual provides a certificate of optimality via the KKT conditions for convex problems
- The KKT conditions lead to efficient optimization algorithms
- The Lagrange multipliers provide information about active constraints at the optimal solution : if $\lambda_i^* > 0$, then $g_i(x^*) = 0$
- The Lagrange multipliers provide information about sensitivity of optimal cost : if λ_i^* is large, then tightening the constraint will significantly increase the cost.

Chapter 5

Constrained Finite Time Optimal Control

Motivation

$$x(k+1) = g\left(x(k), u(k)\right) \qquad x, u \in \mathcal{X}, \mathcal{U}$$

With constrained control we wan to design a control law $u(k) = \kappa\left(x(k)\right)$ such that the system :

1. Satisfies the constraints : $\{x(k)\} \subset \mathcal{X}$, $\{x(k)\} \subset \mathcal{U}$
2. It is stable : $\lim_{k\to\infty} x(k) = 0$
3. Optimizes performance
4. Maximizes the set $\{x(0)|\text{Conditions 1-3 are met}\}$

What we would like to solve :

$$\begin{aligned} J_\infty^*\left(x\left(0\right)\right) \coloneqq & \min_{u(\cdot)} \sum_{i=0}^{\infty} l\left(x_i, u_i\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, \infty \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, \infty \\ & x_0 = x(0) \end{aligned}$$

- The **stage cost** $l(x, u)$: "cost" of being in the state x and applying the input u
- Optimizing over a trajectory provides a **tradeoff between short and long-term benefits** of actions

We'll see that such a control law has many beneficial properties, but we can't compute it : there are an **infinite number of variables**.

What we can sometimes solve :

$$\begin{aligned} J^*_{k\to k+N|k}\left(x\left(k\right)\right) = & \min_{U_{k\to k+N|k}} l_f\left(x_{k+N|k}\right) + \sum_{i=0}^{N-1} l\left(x_{k+i|k}, u_{k+i|k}\right) \\ \text{subject to} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k} \qquad i = 0, \ldots, N-1 \\ & x_{k+i|k} \in \mathcal{X} \quad u_{k+i|k} \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\ & x_{k+N|k} \in \mathcal{X}_f \\ & x_{k|k} = x(k) \\ \text{where } U_{k\to k+N|k} = & \left\{u_{k|k}, \ldots, u_{k+N-1|k}\right\} \end{aligned}$$

We truncate after a finite horizon :

- $l_f(x_{k+N|k})$ approximates the "tail" of the cost
- $\mathcal{X}_f$ approximates the "tail" of the constraints

Notation : We will shorten the name "Optimal Control" to "OC" and "Constrained Optimal Control" to "COC"

5.1 Constrained Linear OC

With the cost function :

$$J\left(x_0, U\right) = l_f(x_N) + \sum_{i=0}^{N-1} l\left(x_i, u_i\right)$$

- The squared Euclidean norm is $l_f(x_N) = x_N^T P x_N$ and $l(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i$, with $P \geq 0$, $Q \geq 0$ and $R > 0$.
- With $p = 1$ or $p = \infty$: $l_f(x_N) = \|Px_N\|_p$ and $l(x_i, u_i) = \|Qx_i\|_p + \|Ru_i\|_p$, with P, Q and R full column rank matrices.

The Constrained Finite Time Optimal Control problem (CFTOC) is then :

$$\begin{aligned} J^*\left(x\left(k\right)\right) = & \min_U J\left(x_0, U\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, N-1 \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(k) \end{aligned} \tag{5.1}$$

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

5.1.1 Feasible Sets

The set of initial states $x(0)$ for which the optimal control problem 5.1 is feasible :

$$\begin{aligned} \mathcal{X}_0 = \{x_0 \in \mathbb{R}^n | \exists \left(u_0, \ldots, u_{N-1}\right) \text{ such that } x_i \in \mathcal{X}, u_i \in \mathcal{U}, \\ i = 0, \ldots, N-1, x_N \in \mathcal{X}_f, \text{ where } x_{i+1} = Ax_i + bu_i\} \end{aligned}$$

In essence : $\mathcal{X}_0$ is the set where our initial conditions for which the problem has a solution are found.

In general $\mathcal{X}_j$ is the set of states x_j at time j for which problem 5.1 is feasible :

$$\begin{aligned}\mathcal{X}_j = \{x_j \in \mathbb{R}^n | \exists\,(u_j, \ldots, u_{N-1}) \text{ such that } x_i \in \mathcal{X}, u_i \in \mathcal{U},\\ i = j, \ldots, N-1, x_N \in \mathcal{X}_f, \text{ where } x_{i+1} = Ax_i + bu_i\}\end{aligned}$$

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

Reminder - Unconstrained Solution

For quadratic cost (squared Euclidean norm) and **no state and input constraints** :

$$\{x \in \mathcal{X}, u \in \mathcal{U}\} = \mathbb{R}^{n+m} \quad \mathcal{X}_f = \mathbb{R}^n$$

we have the **time-varying** linear control law

$$u_i^* = F_i x_i \quad i = 0, \ldots, N-1$$

If $N \to \infty$, we have the **time-variant** linear control law :

$$u_i^* = F_\infty x_i \quad i = 0, 1, \ldots$$

5.2 COC - Quadratic Cost

5.2.1 Transform Quadratic Cost CFTOC into QP

Our quadratic cost CFTOC :

$$\begin{aligned} J^*(x(k)) = &\min_U x_N^T P x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, N-1 \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(k) \end{aligned}$$

Our corresponding QP problem :

$$\begin{aligned} & \min_{z \in \mathbb{R}^n} \frac{1}{2} z^T H z + q^T z + r \\ \text{subject to} \quad & Gz \leq h \\ & Az = b \end{aligned}$$

5.2.2 Construction of the QP with Substitution

Idea : Substitute the state equations

$$x_{i+1} = Ax_i + Bu_i \quad x_0 = x(k)$$

Step 1 : Rewrite the cost as :

$$\begin{aligned} J(x(k), U) &= U^T H U + 2x(k)^T F U + x(k)^T Y x(k) \\ &= \begin{pmatrix} U^T & x(k)^T \end{pmatrix} \begin{pmatrix} H & F^T \\ F & Y \end{pmatrix} \begin{pmatrix} U^T \\ x(k)^T \end{pmatrix} \end{aligned}$$

Remark : $\begin{pmatrix} H & F^T \\ F & Y \end{pmatrix} \geq 0$ since $J(x(k), U) \geq 0$ by assumption.

Step 2 : Rewrite the constraints compactly as

$$GU \leq w + Ex(k)$$

Step 3 : Rewrite the constrained optimal control problem as

$$\begin{aligned} J^*\left(x\left(k\right)\right) = & \min_U \begin{pmatrix} U^T & x(k)^T \end{pmatrix} \begin{pmatrix} H & F^T \\ F & Y \end{pmatrix} \begin{pmatrix} U^T \\ x(k)^T \end{pmatrix} \\ \text{subject to} \quad & GU \leq w + Ex(k) \end{aligned}$$

The inequalities $GU \leq w + Ex(k)$ for $\mathcal{X}, \mathcal{U}$ and $\mathcal{X}_f$ are given by :

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

Then G, E and w are defined as follows :

$$G = \begin{pmatrix} A_u & 0 & 0 \cdots & 0 \\ 0 & A_u & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_u \\ 0 & 0 & \cdots & 0 \\ A_x B & 0 & \cdots & 0 \\ A_x AB & A_x B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_f A^{N-1} B & A_f A^{N-2} B & \cdots & A_f B \end{pmatrix} \quad E = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -A_f A^N \end{pmatrix} \quad w = \begin{pmatrix} b_u \\ b_u \\ \vdots \\ b_u \\ b_x \\ b_x \\ b_x \\ \vdots \\ b_f \end{pmatrix}$$

The **solution of the QP with substitution** is :

$$\begin{aligned} J^*\left(x\left(k\right)\right) = & \min_U \begin{pmatrix} U^T & x(k)^T \end{pmatrix} \begin{pmatrix} H & F^T \\ F & Y \end{pmatrix} \begin{pmatrix} U^T \\ x(k)^T \end{pmatrix} \\ \text{subject to} \quad & GU \leq w + Ex(k) \end{aligned}$$

For a given $x(k)$, U^* can be found via a QP solver.

5.2.3 Construction of the QP without Substitution

Idea : We keep the state equations as equality constraints (which is often more efficient).

We transform the CFTOC problem into the QP problem :

$$\begin{aligned} J^*\left(x\left(k\right)\right) = & \min_U \begin{pmatrix} z^T & x(k)^T \end{pmatrix} \begin{pmatrix} \bar{H} & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} z^T \\ x(k)^T \end{pmatrix} \\ \text{subject to} \quad & G_{in} z \leq w_{in} + E_{in} x(k) \\ & G_{eq} z = E_{eq} x(k) \end{aligned}$$

We define the variable :

$$z = \begin{pmatrix} x_1^T & \cdots & x_N^T & u_0^T \cdots & u_{N-1}^T \end{pmatrix}^T$$

From the equalities of the system dynamics, $x_{i+1} = Ax_i + Bu_i$ we have :

$$G_{eq} = \left(\begin{array}{ccccc|ccccc} I & & & & & -B & & & & \\ -A & I & & & & & -B & & & \\ & -A & I & & & & & -B & & \\ & & \ddots & \ddots & & & & & \ddots & \\ & & & -A & I & & & & & -B \end{array}\right) \quad E_{eq} = \begin{pmatrix} A \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The inequalities $G_{in}z \leq w_{in} + E_{in}x(k)$ for $\mathcal{X}, \mathcal{U}$ and $\mathcal{X}_f$ are given by :

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

The matrices G_{in}, w_{in} and E_{in} are :

$$G_{in} = \left(\begin{array}{ccccc|ccccc} 0 & & & & & 0 & & & & \\ A_x & & & & & 0 & & & & \\ & \ddots & & & & & \ddots & & & \\ & & A_x & & & & & 0 & & \\ & & & A_f & & & & & 0 & \\ \hline 0 & & & & & A_u & & & & \\ 0 & & & & & A_u & & & & \\ & \ddots & & & & & \ddots & & & \\ & & 0 & & & & & A_u & & \\ & & & 0 & & & & & A_u & \end{array}\right) \quad w_{in} = \left(\begin{array}{c} b_x \\ b_x \\ \vdots \\ b_x \\ b_f \\ \hline b_u \\ b_u \\ \vdots \\ b_u \\ b_u \end{array}\right) \quad E_{in} = \begin{pmatrix} -A_x^T \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

We build the cost function from the MPC cost : $x_N^T P x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i$

$$\bar{H} = \left(\begin{array}{cccc|ccc} Q & & & & & & \\ & \ddots & & & & & \\ & & Q & & & & \\ & & & P & & & \\ \hline & & & & R & & \\ & & & & & R & \\ & & & & & & \ddots \\ & & & & & & & R \end{array}\right)$$

Matlab Hint : `barH = blkdiag(kron(eye(N-1),Q), P, kron(eye(N),R))`

5.2.4 Quadratic Cost State Feedback Solution

$$\begin{aligned} J^*(x(k)) = & \min_U \begin{pmatrix} U^T & x(k)^T \end{pmatrix} \begin{pmatrix} H & F^T \\ F & Y \end{pmatrix} \begin{pmatrix} U^T \\ x(k)^T \end{pmatrix} \\ \text{subject to} \quad & GU \leq w + Ex(k) \end{aligned}$$

The CFTOC problem is a **multi-parametric quadratic program (mp-QP)** with the following solution properties :

- The first component of the optimal solution has the form :

$$u_0^* = \kappa(x(k)) \qquad \forall x(k) \in \mathcal{X}_0$$

  $\kappa : \mathbb{R}^n \to \mathbb{R}^m$, is a continuous and PieceWise Affine on Polyhedra

$$\kappa(x) = F^j x + g^j \quad \text{if} \quad x \in CR^j \quad j = 1, \ldots, N^r$$

- The polyhedral sets $CR^j = \left\{x \in \mathbb{R}^n | H^j x \leq K^j\right\}$, $j = 1, \ldots, N^r$ are a partition of the feasible polyhedron $\mathcal{X}_0$.
- The value function $J^*(x(k))$ is convex and piecewise quadratic on polyhedra.

$\Rightarrow$ Explicit MPC addresses how to compute this solution.

5.2.5 Example - Solving the CFTOC

Consider the double integrator :

$$x(t+1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t)$$
$$y(t) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(t)$$

subject to the constraints :

$$\begin{aligned} -1 \leq & u(k) \leq 1 && k = 0, \dots, 5 \\ \begin{pmatrix} -10 \\ -10 \end{pmatrix} \leq & x(k) \leq \begin{pmatrix} 10 \\ 10 \end{pmatrix} && k = 0, \dots, 5 \end{aligned}$$

Compute the **state feedback** optimal controller $u^*(x(k))$ solving the CFTOC problem with $N = 6$, $Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $R = 0.1$ and P the solution of the ARE, $\mathcal{X}_f = \mathbb{R}^2$.

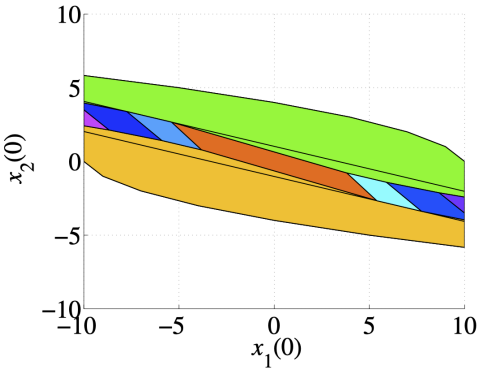


Figure 5.1: Partition of the state space for the affine control law $u^*(x)$ ($N_0^r = 13$)

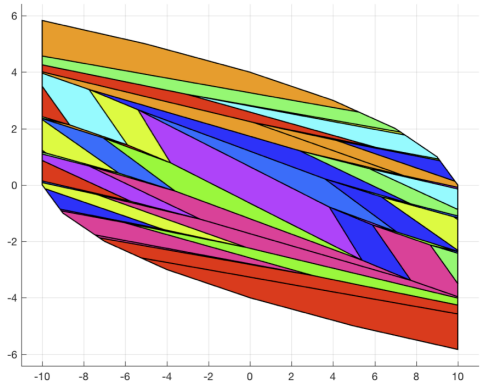


Figure 5.2: Partition of the state space for the affine control law $u^*(x)$ ($N_0^r = 61$)

We notice that the LQR controller would only be able to operate within the purple space in the middle of the total primal state space that we see in figure 5.2.

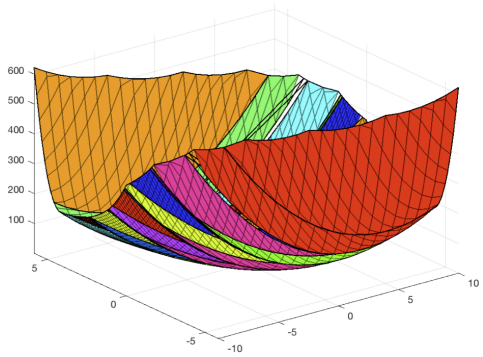


Figure 5.3: Value function for the affine control law $u^*(x)$ $(N_0^r = 61)$

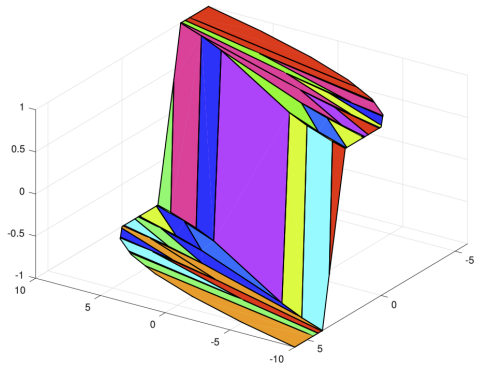


Figure 5.4: Optimal control input for the affine control law $u^*(x)$ $(N_0^r = 61)$

5.3 COC - 1 & ∞-Norm Cost

5.3.1 Transform 1-/∞-Norm Cost CFTOC into LP

Our 1-Norm / ∞-Norm Cost CFTOC :

$$
\begin{aligned}
J^*\left(x\left(k\right)\right) = & \min_U \|Px_N\|_p + \sum_{i=0}^{N-1} \|Qx_i\|_p + \|Ru_i\| \\
\text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, N-1 \\
& x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\
& x_N \in \mathcal{X}_f \\
& x_0 = x(k)
\end{aligned}
$$

Our corresponding LP problem is :

$$
\begin{aligned}
& \min_{z \in \mathbb{R}^n} c^T z \\
\text{subject to} \quad & Gz \leq h \\
& Az = b
\end{aligned}
$$

5.3.2 l_∞ Minimization

Constrained l_∞ (Chebyshev) minimization :

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} \|x\|_\infty \\ \text{subject to} \quad & Fx \leq g \end{aligned}$$

We write this as a maximum of linear functions, the problem above is equivalent to :

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} \left[\max\{x_1,\ldots,x_n,-x_1,\ldots,-x_n\}\right] \\ \text{subject to} \quad & Fx \leq g \end{aligned}$$

Which is also equivalent to :

$$\begin{aligned} & \min_{x,t} t \\ \text{subject to} \quad & x_i \leq t \qquad i=1,\ldots,n \\ & =-x_i \leq t \qquad i=1,\ldots,n \\ & Fx \leq g \end{aligned} \quad \Rightarrow \quad \begin{aligned} & \min_{x,t} t \\ \text{subject to} \quad & -\mathbf{1}t \leq x \leq \mathbf{1}t \\ & Fx \leq g \end{aligned}$$

The last formulation is known as the **epigraph formulation**. The notation $\mathbf{1}$ indicates a vector of ones.

The constraint $-\mathbf{1}t \leq x \leq \mathbf{1}t$ bounds the absolute value of every element of x with a common scalar variable t.

5.3.3 l_1 Minimization

Constrained l_1 Minimization

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} \|x\|_1 \\ \text{subject to} \quad & Fx \leq g \end{aligned}$$

We write this as a maximum of linear functions, the problem above is equivalent to :

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} \left[\sum_{i=1}^{m} \max\{x_i,-x_i\}\right] \\ \text{subject to} \quad & Fx \leq g \end{aligned}$$

Which is also equivalent to :

$$\begin{aligned} & \min_{x\in\mathbb{R}^n,t\in\mathbb{R}^n} t_1+\cdots+t_n \\ \text{subject to} \quad & x_i \leq t \qquad i=1,\ldots,n \\ & =-x_i \leq t \qquad i=1,\ldots,n \\ & Fx \leq g \end{aligned} \quad \Rightarrow \quad \begin{aligned} & \min_{x\in\mathbb{R}^n,t\in\mathbb{R}^n} \mathbf{1}^T t \\ \text{subject to} \quad & -t \leq x \leq t \\ & Fx \leq g \end{aligned}$$

The last formulation is known as the **epigraph formulation**. The notation $\mathbf{1}$ indicates a vector of ones.

The constraint $-\mathbf{1}t \leq x \leq \mathbf{1}t$ bounds the absolute value of every element of x with a common scalar variable t.

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

5.3.4 Construction of the LP with Substitution

Recall that the ∞-norm problem can be equivalently formulated as :

$$\begin{aligned}
&\min_z \varepsilon_0^x + \cdots + \varepsilon_N^x + \varepsilon_0^u + \cdots \varepsilon_{N-1}^u \\
\text{subject to} \quad & -\mathbf{1}_n \varepsilon_i^x \leq \pm Q \underbrace{\left[A^i x_0 + \sum_{j=0}^{i-1} A^j B u_{i-1-j} \right]}_{\tilde{Q} S^u u} \\
& -\mathbf{1}_r \varepsilon_N^x \leq \pm P \underbrace{\left[A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \right]}_{\tilde{Q} S^u u} \\
& -\mathbf{1}_m \varepsilon_i^u \leq \pm R u_i \\
& A^i x_0 + \sum_{j=0}^{i-1} A^j B u_{i-1-j} \in \mathcal{X} \qquad u_i \in \mathcal{U} \\
& A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \in \mathcal{X}_f \\
& x_0 = x(k) \qquad i = 0, \ldots, N-1
\end{aligned}$$

The problem results in the following standard LP :

$$\begin{aligned}
&\min_z c^T z \\
\text{subject to} \quad & \bar{G} z \leq \bar{w} + \bar{S} x(k)
\end{aligned}$$

where $z \coloneqq \left\{ \varepsilon_0^x, \ldots, \varepsilon_N^x, \varepsilon_0^u, \ldots, \varepsilon_{N-1}^u, u_0^T, \ldots, u_{N-1}^T \right\} \in \mathbb{R}^s$
$s \coloneqq (m+1)N + N + 1$ and

$$\bar{G} = \begin{pmatrix} G_\varepsilon & 0 \\ 0 & G \end{pmatrix} \qquad \bar{S} = \begin{pmatrix} S_\varepsilon \\ S \end{pmatrix} \qquad \bar{w} = \begin{pmatrix} w_\varepsilon \\ w \end{pmatrix}$$

For a given $x(k)$, U^* can be obtained via an LP solver.

$\hookrightarrow$ The 1-norm case works similarly.

5.3.5 1-Norm / ∞-Norm State Feedback Solution

$$\begin{aligned}
&\min_z c^T z \\
\text{subject to} \quad & \bar{G} z \leq \bar{w} + \bar{S} x(k)
\end{aligned}$$

The CFTOC problem is a **multi-parametric linear program (mp-LP)** with the following solution properties :

- The first component of the multi-parametric solution has the form

$$u_0^* = \kappa\left(x(0)\right) \quad \forall x(0) \in \mathcal{X}_0$$

$\kappa : \mathbb{R}^n \to \mathbb{R}^m$, is continuous and PieceWise Affine on Polyhedra

$$k(x) = F^j x + g^j \quad \text{if} \quad x \in CR^j \qquad j = 1, \ldots, N^r$$

- The polyhedral sets $CR^j = \left\{x \in \mathbb{R}^n | H^j x \leq K^j\right\},\ j = 1, \ldots, N^r$ are a partition of the feasible polyhedron $\mathcal{X}_0$.
- In case of multiple optimizers a piecewise affine control law exists.
- The value function $J^*(x(0))$ is convex and piecewise linear on polyhedra.

5.3.6 Solution Properties : Quadratic Vs. 1-/∞-Norm Cost

Let n be the number of optimization variables.

The quadratic cost (pos. def) : Solution is either

- Unique and in the interior of the feasible set $\rightarrow$ no constraints active
- Unique and on the boundary of the feasible set $\rightarrow$ at least 1 active constraint

Linear cost : Solution is either :

- Unbounded
- Unique at a vertex of the feasible set $\rightarrow$ at least n active constraints
- A set of multiple optima $\rightarrow$ at least 1 active constraint

5.4 Receding Horizon

Let us start by considering the DT model :

$$\begin{aligned} x(k+1) &= Ax(k) + bu(k) \\ y(k) &= Cx(k) \\ x(k) \in \mathcal{X}, u(k) &\in \mathcal{U}, \forall k \geq 0 \end{aligned}$$

The Constrained Finite Time Optimal Control (CFTOC) problem

$$\begin{aligned} J^*\left(x\left(k\right)\right) = & \min_U J\left(x_0, U\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, N-1 \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(k) \end{aligned} \tag{5.2}$$

is solved at time k with $U_{k \rightarrow k+N|k} = \left\{u_{k|k}, \ldots, u_{k+N-1|k}\right\}$.

- $x(k)$ is the state of the system at time k.
- $x_{i+k|k}$ is the state of the model at time $k+i$, predicted at time k obtained by starting from the current state $x_{k|k} = x(k)$ and applying to the system model

  $$x_{k+1|k} = Ax_{k|k} + Bu_{k|k}$$

  the input sequence $u_{k|k}, \ldots, u_{k+i-1|k}$.
- For instance, $x_{3|1}$ represents the predicted state at time 3 when the prediction is done at time $k = 1$ starting from the current state $x(1)$. It is different, in general, from $x_{3|2}$ which is the predicted state at time 3 when the prediction is done at time $k = 2$ starting from the current state $x(2)$.

- Similarly, $u_{k+i|k}$ is read as "the input u at time $k+i$ computed at time k".
- Let $U^*_{k\to k+N|k} = \left\{u^*_{k|k}, \ldots, u^*_{k+N-1|k}\right\}$ be the optimal solution. The first element of $U^*_{k\to k+N|k}$ is applied to the system

$$u(k) = u^*_{k|k}\left(x(k)\right)$$

- The CFTOC problem is reformulated and solved at time $k+1$, based on the new state $x_{k+1|k+1} = x(k+1)$

The receding horizon control law is :

$$\kappa_k\left(x(k)\right) = u^*_{k|k}\left(x(k)\right)$$

The closed-loop system becomes :

$$x(k+1) = Ax(k) + B\kappa_k\left(x(k)\right) \coloneqq g_{cl}\left(x(k)\right) \quad k \geq 0$$

5.4.1 RHC - Time-Invariant Systems

As the system, the constraints and the cost function are time-invariant, the solution $\kappa_k\left(x(k)\right)$ becomes a time-invariant function of the initial state $x(k)$. Thus, we can simplify the notation as

$$\begin{aligned} J^*\left(x\left(k\right)\right) = & \min_U l_f\left(x_N\right) + \sum_{i=0}^{N-1} l\left(x_i, u_i\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, N-1 \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(k) \end{aligned} \tag{5.3}$$

where $U = \{u_0, \ldots, u_{N-1}\}$.

The control law and closed-loop system are **time-invariant** as well, and we write $\kappa\left(x(k)\right)$ for $\kappa_k\left(x(k)\right)$.

Chapter 6

Invariance

6.1 Objectives of Constrained Control

$$x(k+1) = g\left(x(k), u(k)\right) \qquad x, u \in \mathcal{X}, \mathcal{U}$$

With constrained control we wan to design a control law $u(k) = \kappa\left(x(k)\right)$ such that the system :

1. Satisfies the constraints : $\{x(k)\} \subset \mathcal{X}$, $\{x(k)\} \subset \mathcal{U}$
2. It is stable : $\lim_{k\to\infty} x(k) = 0$
3. Optimizes performance
4. Maximizes the set $\{x(0)|\text{Conditions 1-3 are met}\}$

In this chapter, we will focus on how to ensure that criterion 1 is met.

6.1.1 Limitations of Linear Controllers

If we consider the system :

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

with the constraints :

$$\mathcal{X} \coloneqq \{x | \|x\|_\infty \leq 5\}$$
$$\mathcal{U} \coloneqq \{u | \|u\|_\infty \leq 1\}$$

Is we consider an LQR controller, with $Q = I$ and $R = 1$, does the linear controller work?

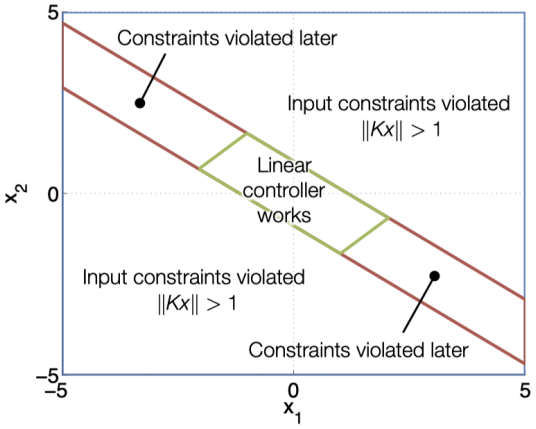


Figure 6.1

As we can see in Figure 6.1, the linear controller will work, however the region where it works is very small. Instead of a linear controller, **we can use non-linear control (MPC) to increase the region of attraction**, as shown in Figure 6.2.

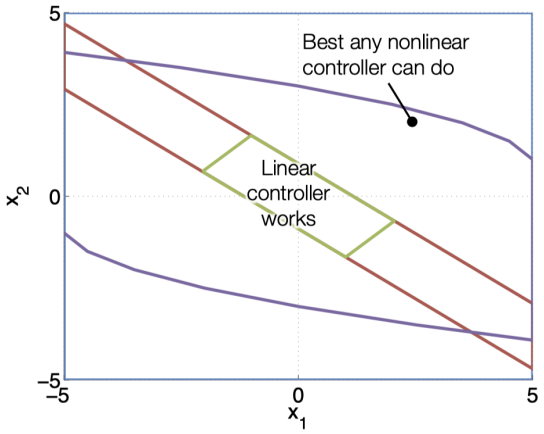


Figure 6.2

Invariance - Which states are "good"?

If we take the system :

$$\dot{x} = \begin{pmatrix} -2\zeta\omega & -\omega^2 \\ 1 & 0 \end{pmatrix}$$

where $\omega = 10$ and $\zeta = 0.01$ sampled at 10 Hz, with the constraints :

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

We see in Figure 6.3 that the initial state is in the constraints, as well in the next 3 states :

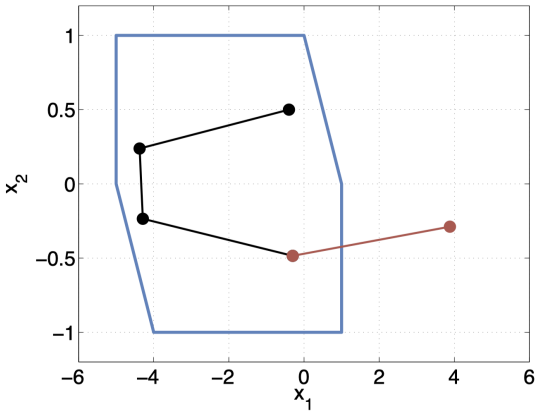


Figure 6.3

We would need to be able to look into the future to determine if the trajectory beginning at the current state always remains within the constraints.

Controlled Invariance - Does a good input exist?

If we take the system :

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

with the constraints :

$$\begin{aligned} \|u(k)\|_\infty &\leq 0.1 \\ \|x(k)\|_\infty &\leq 1 \\ \| \begin{pmatrix} 1 & 1 \end{pmatrix} x(k)\|_\infty &\leq 1 \end{aligned}$$

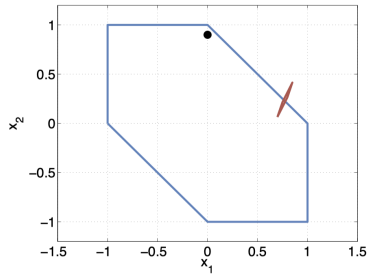


Figure 6.4

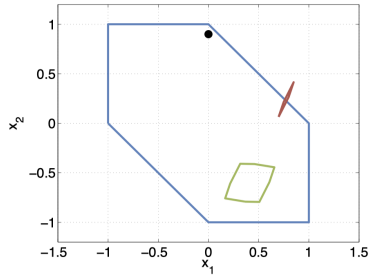


Figure 6.5

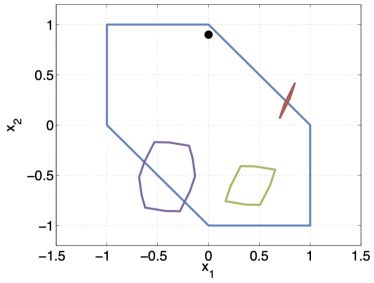


Figure 6.6

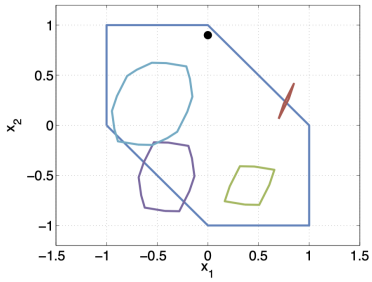


Figure 6.7

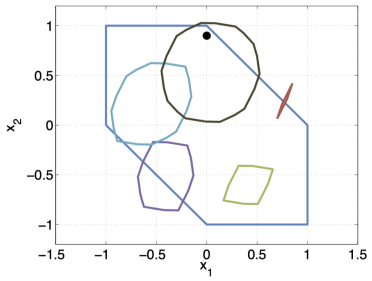


Figure 6.8

We can choose from a set of inputs ⇒ The set of possible next states.

Controlled invariance : Will there always exist a valid input that will maintain the constraints?

6.1.2 Invariance

Goal : We want to satisfy our constraints, for an **autonomous system** $x(k+1) = g(x(k))$, or **closed-loop system** $x(k+1) = g(x(k), \kappa(k))$ for a **given** controller κ.

Definition 6.1.1. Positive Invariant Sets

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

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

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.

Definition 6.1.2. Maximal 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.

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 6.1.3. Pre-Sets

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

$$\text{pre}(S) \coloneqq \{x | g(x) \in S\}$$

6.1.3 Pre-Set Example - Pendulum

If we take our system to be a pendulum :

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

discretized with forward Euler at 1 Hz, and with the target set :

$$T \coloneqq \{x | \|x\|_2 \leq 1\}$$

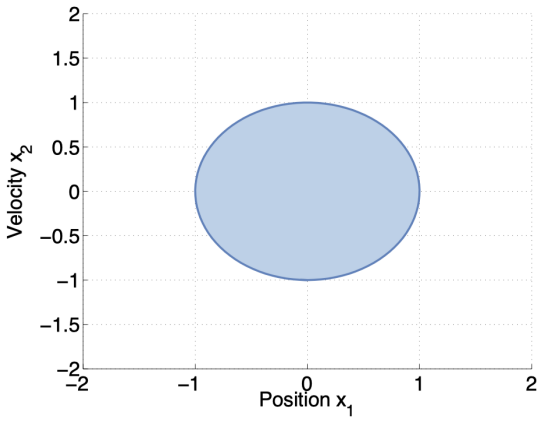


Figure 6.9

Which states will be in the target set at the next point in time?

We consider the phase diagram (see Figure 6.10)

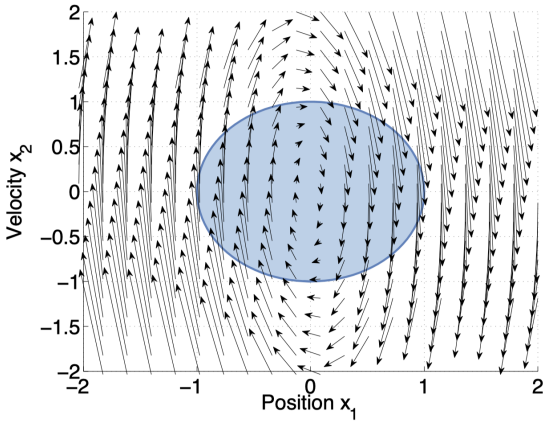


Figure 6.10

We collect all of the points where the arrows end within the blue set. This collection is the pre-set. The pre-set is made up of those states that will be in T in one time step.

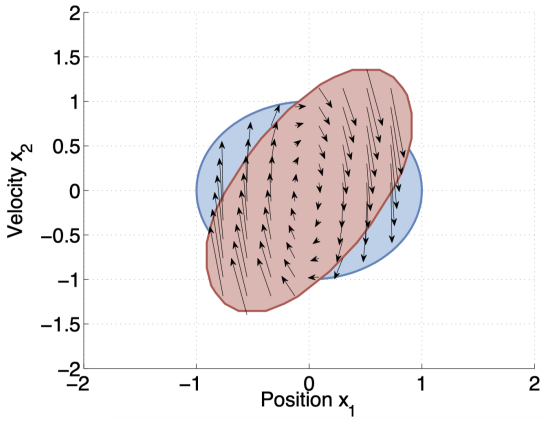


Figure 6.11

This is extremely difficult to compute, except in some special cases.

6.1.4 Pre-Set Computation - Linear Autonomous System

Definition 6.1.4. Pre-Set

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

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

If $S \coloneqq \{x | Fx \leq f\}$, then $\text{pre}(S) = \{x | FAx \leq f\}$.

For the system :

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

with the constraints :

$$\begin{pmatrix} -5 \\ -10 \end{pmatrix} \leq x(k) \leq \begin{pmatrix} 5 \\ 10 \end{pmatrix} \qquad \|u(k)\|_\infty \leq 0.1$$

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

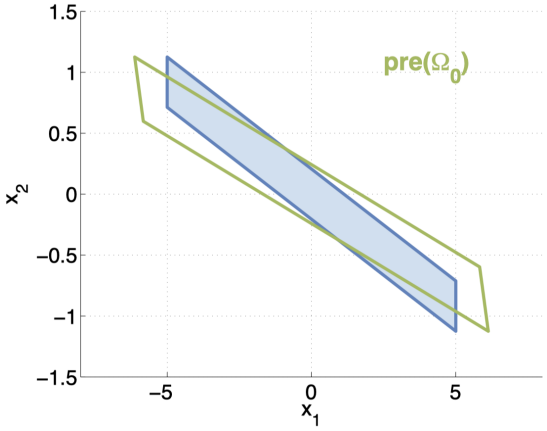


Figure 6.12

6.1.5 Invariant Set Conditions

Theorem 6.1.1. ***Positive Invariant Sets***
A set $\mathcal{O}$ is a ***positive invariant set*** *if and only if :*

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

We prove the contrapositive for both the necessary and sufficient conditions.

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

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

Remark : Note that $\mathcal{O} \subseteq \text{pre}(\mathcal{O}) \Leftrightarrow \text{pre}(\mathcal{O}) \cap \mathcal{O} = \mathcal{O}$.

6.1.6 Computing Invariant Sets

Algorithm 1: Conceptual Algorithm to Compute Invariant Set

Input : g, $\mathcal{X}$
Output: $\mathcal{O}_\infty$

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

Requirements :

- Represent set Ω_i (often and polytopes)
- Pre-set computation
- Intersection
- Equality test (bi-directional subset)

If we apply this algorithm to our previous system :

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

with the constraints :

$$\begin{pmatrix} -5 \\ -10 \end{pmatrix} \leq x(k) \leq \begin{pmatrix} 5 \\ 10 \end{pmatrix}$$
$$-0.1 \leq u(k) \leq 0.1$$

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

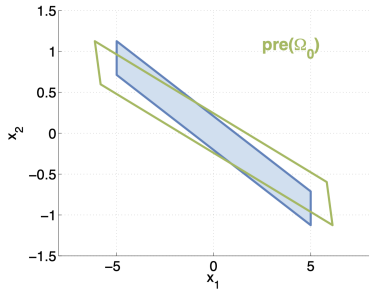


Figure 6.13

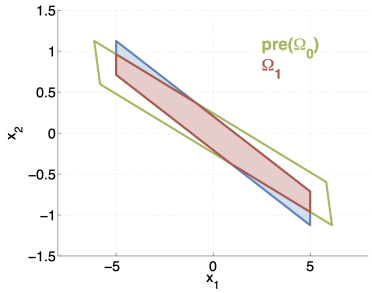


Figure 6.14

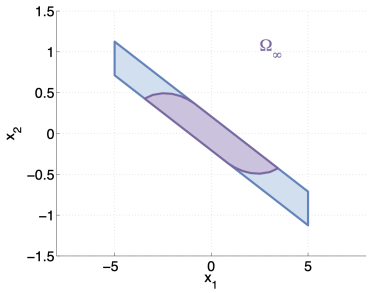


Figure 6.15

6.2 Controlled Invariance

Definition 6.2.1. Control Invariant Set

A set $\mathcal{C} \subseteq \mathcal{X}$ is said to be a **control invariant set** is

$$x(k) \in \mathcal{C} \quad \rightarrow \quad \exists u(k) \in \mathcal{U} \text{ such that } g\left(x(k), u(k)\right) \in \mathcal{C} \quad \forall k \in \mathbb{N}^{+}$$

This defines the states for which there exists a **controller** that will satisfy constraints for **all time**.

Definition 6.2.2. Maximal Control Invariant Set

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

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.

6.2.1 Conceptual Calculation of Control Invariant Sets

The concept of a pre-set extends to systems with exogenous inputs :

$$\text{pre}(S) \coloneqq \{x | \exists u \in \mathcal{U} \text{ subject to } g(x, u) \in S\}$$

The same geometric condition holds for control invariant sets :

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

As a result, the same conceptual algorithm as Algorithm1 can be used, however, it is much harder to compute the pre-set!

Pre-Set Computation - Controlled System

If we consider the system $x(k+1) = Ax(k) + Bu(k)$ under the constraints $u(k) \in \mathcal{U} \coloneqq \{u | Gu \leq g\}$ and the set $S \coloneqq \{x | FX \leq f\}$.

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

This is a **projection** operation.

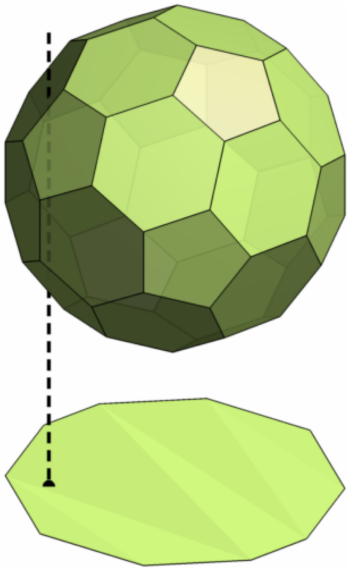


Figure 6.16

Computing Control Invariant Sets

Applying Algorithm 1 to our system :

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

with the constraints :

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

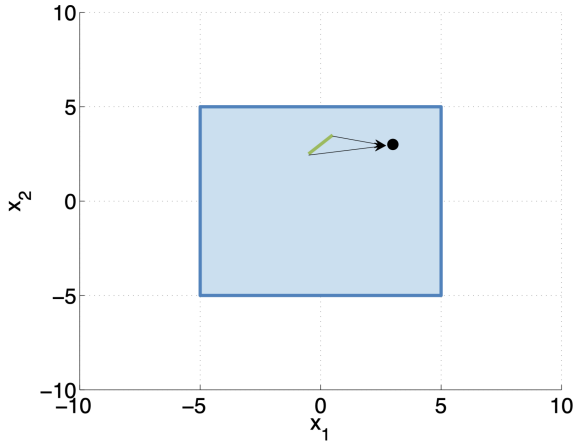


Figure 6.17

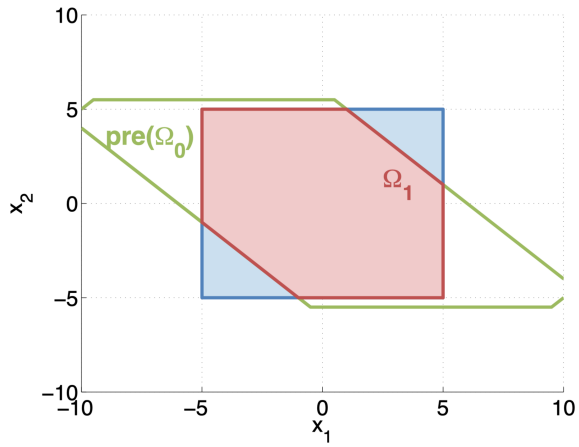


Figure 6.18

An entire set of states can map into each point. The pre-set is a lot larger, but much more difficult to compute.

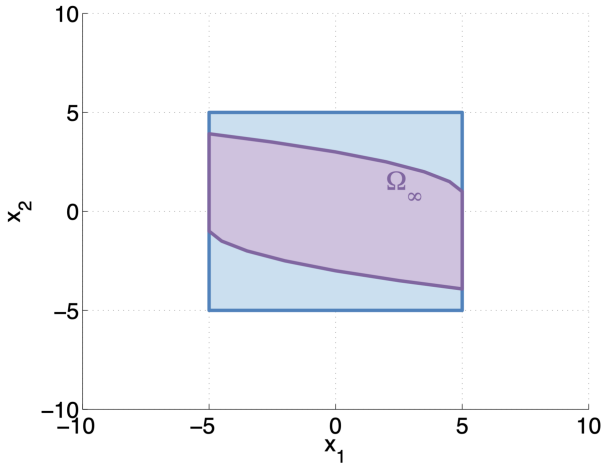


Figure 6.19

The maximum control invariant set is the best any controller can do.

Control Invariant Set → Control Law

Let C be a control invariant set for the system $x(k+1) = g\left(x(k), u(k)\right)$.

A control law $\kappa(x(k))$ will guarantee that the system $x(k+1) = g\left(x(k), u(k)\right)$ will satisfy the constraints **for all time** if :

$$g\left(x, \kappa(k)\right) \in C \quad \forall x \in C$$

We can use this fact to **synthesise** a control law from a control invariant set by solving an optimisation problem :

$$\kappa(x) \coloneqq \arg\min \left\{ f(x,u) | g(x,u) \in C \right\}$$

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

This doesn't ensure that the system will converge, but it will satisfy the constraints.

6.2.2 Relation to MPC

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

So why don't we compute them?

We can't ...

- The constrained linear systems are often too complex
- The constrained non-linear systems are often always too complex.

MPC **implicitly** describes a control invariant set such that it is easy to represent and compute.

6.3 Practical Computation of Invariant Sets

6.3.1 Ellipsoids

Definition 6.3.1. Ellipsoids

Let $P \geq 0$ by a symmetric and positive-definite matrix in $\mathbb{R}^{n\times n}$ and $x_c \in \mathbb{R}^n$. The set

$$E \coloneqq \left\{x|\,(x-x_c)^T\,P\,(x-x_c) \leq 1\right\}$$

is an **ellipse**.

Ellipsoids are useful because the complexity of evaluating containment is always quadratic in the dimension, whereas polyhedra can be arbitrarily complex.

6.3.2 Invariant Sets for Lyapunov Functions

Lemma 6.3.1. If $V : \mathbb{R}^n \to \mathbb{R}$ is a Lyapunov function for the system $x(k+1) = g\,(x(k))$, then

$$Y \coloneqq \{x|V(x) \leq \alpha\}$$

is an invariant set for all $\alpha \geq 0$.

Proof. We have the basic properties :

- $V(x) \geq 0$ for all x
- $V\,(g(x)) - V(x) < 0$

The second property implies that once $V\,(x(k)) \leq \alpha$, $V\,(x(j))$ will be less than α for all $j \geq k \to$ Invariance. □

We often want the largest invariant set contained in our constraints.

If V is a Lyapunov function for the system $x(k+1) = g\,(x(k))$, and our constraints are given by the set $\mathcal{X}$, then we maximise α such that

$$Y_\alpha \coloneqq \{x|V(x) \leq \alpha\} \subseteq \mathcal{X}$$

If we consider the system $x(k+1) = Ax(k)$, and assume $P > 0$ satisfies the condition :

$$A^T P A - P < 0$$

Then the function $V(x) = x(k)^T P x(k)$ is a Lyapunov function.

Our goal is to find the largest α such that the invariant set Y_α is contained in the system constraints $\mathcal{X}$:

$$Y_\alpha \coloneqq \left\{x|x^T P x \leq \alpha\right\} \subset \mathcal{X} \coloneqq \{x|Fx \leq f\}$$

Equivalently, we want to solve the problem :

$$\begin{aligned} &\max_{\alpha} \alpha \\ \text{subject to} \quad &h_{Y_\alpha}\,(F_i) \leq f_i \qquad \text{for all } i \in \{1,\ldots,n\} \end{aligned}$$

where h_{Y_α} is the support function fo an ellipse.

Maximum Ellipsoidal Invariant Sets

Support of an ellipse :

$$\begin{aligned} H_{Y_\alpha} &= \max_x \gamma^T x \\ \text{subject to} \quad & x^T P x \leq \alpha \end{aligned}$$

Change of variables $y \coloneqq P^{1/2}x$:

$$\begin{aligned} H_{Y_\alpha}(\gamma) &= \max_x \gamma^T P^{1/2} y \\ \text{subject to} \quad & y^T y \leq \sqrt{\alpha}^2 \end{aligned}$$

Which can be solved by inspection :

$$h_{Y_\alpha} = \gamma^T P^{-1/2} \frac{P^{-1/2}\gamma}{\|P^{-1/2}\gamma\|}\sqrt{\alpha} = \|P^{-1/2}\gamma\|\sqrt{\alpha}$$

The largest ellipse in a polytope is now a one-dimensional optimization problem :

$$\begin{aligned} \alpha^* &= \max_\alpha \alpha \\ \text{subject to} \quad & \|P^{-1/2}F_i^T\|^2 \alpha \leq f_i^2 \quad \text{for all } i \in \{1,\dots,n\} \\ \hookrightarrow \quad \alpha^* &= \min_{i\in\{1,\dots,n\}} \frac{f_i^2}{F_i P^{-1} F_i^T} \end{aligned}$$

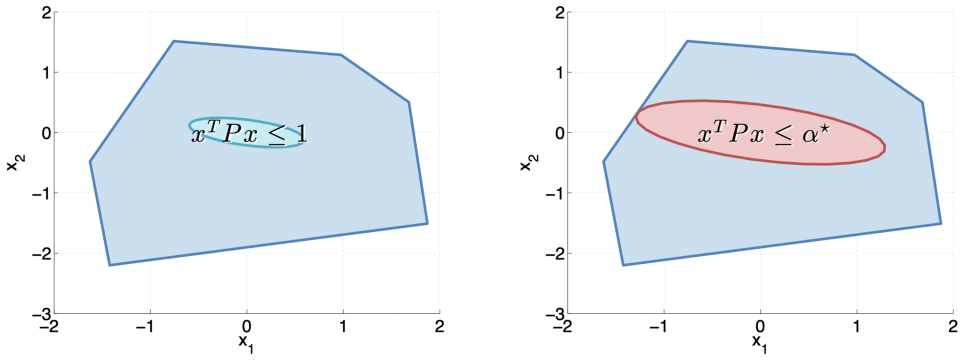


Figure 6.20

It is possible to optimize over P, maximising the volume of the ellipse, subject to stability and containment constraints (convex semi-definite program).

Chapter 7

Feasibility & Stability

Motivation

$$x(k+1) = g\left(x(k), u(k)\right) \qquad x, u \in \mathcal{X}, \mathcal{U}$$

With constrained control we wan to design a control law $u(k) = \kappa\left(x(k)\right)$ such that the system :

1. Satisfies the constraints : $\{x(k)\} \subset \mathcal{X}$, $\{x(k)\} \subset \mathcal{U}$
2. It is stable : $\lim_{k\to\infty} x(k) = 0$
3. Optimizes performance
4. Maximizes the set $\{x(0)|\text{Conditions 1-3 are met}\}$

In this chapter, we will demonstrate that these objectives can be met in a predictive control framework.

Constrained Infinite Time Optimal Control (What we would like to solve)

$$\begin{aligned} J^*_\infty\left(x(0)\right) = \min_{u(\cdots)} & \sum_{i=0}^{\infty} l\left(x_i, u_i\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \ldots, \infty \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, \infty \\ & x_0 = x(0) \end{aligned}$$

- The **stage cost** $l(x, u)$ is the "cost" of being in a state x and applying an input u.
- Optimizing over a trajectory provides a **tradeoff between short and long-term benefits** of actions.
- We'll see that such a control law has many beneficial properties, but we can't compute it, there are an **infinite number of variables**.

Constrained Infinite Time Optimal Control (What we can sometimes solve)

We approximate the rest of the sum to simulate an infinite horizon. We also add the constraint so that the final step is within the set $\mathcal{X}_f$, then we are sure that we will stay within the

constraints for all time.

$$
\begin{aligned}
J^*_{k\to k+N|k}\left(x\left(k\right)\right) = \min_{U_{k\to k+N|k}} \; & l_f\left(x_{k+N|k}\right) + \sum_{i=0}^{N-1} l\left(x_{k+i|k}, u_{k+i|k}\right) \\
\text{subject to} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k} \qquad i = 0, \ldots, N-1 \\
& x_{k+i|k} \in \mathcal{X} \quad u_{k+i|k} \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\
& x_{k+N|k} \in \mathcal{X}_f \\
& x_{k|k} = x(k) \\
\text{where } U_{k\to k+N|k} = & \left\{u_{k|k}, \ldots, u_{k+N-1|k}\right\}
\end{aligned}
$$

We truncate after a finite horizon :

- $l_f\left(x_{k+N|k}\right)$ approximates the "tail" of the cost
- $\mathcal{X}_f$ approximates the "tail" of the constraints

7.1 MPC - Key Points Illustrated

7.1.1 Example - Cessna Citation Aircraft

The linearized continuous-time model of the Cessna (at an altitude of 5000 m and at a speed of 182.2 m/s) is :

$$
\dot{x} = \begin{pmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{pmatrix} u
$$

$$
y = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} x
$$

- Input : Elevator Angle
- States : x_1 : Angle of Attack, x_2 : Pitch Angle, x_3 : Pitch Rate, x_4 : Altitude
- outputs : Pitch angle and altitude
- Constraints : Elevator Angle ± 0.262 rad ($\pm 15°$), Elevator Rate ± 0.524 rad/s ($\pm 60°/s$), Pitch Angle ± 0.349 rad ($\pm 39°$)

The open-loop response is unstable (open-loop poles : 0, 0, -1.5594$\pm$2.29i).
The LQR and MPC controllers will be respectively :

$$
\begin{aligned}
J^*_\infty\left(x(k)\right) = \min \; & \sum_{i=0}^{\infty} x_i^T Q x_i + u_i^T R u_i \\
\text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \\
& x_0 = x(k)
\end{aligned}
\qquad
\begin{aligned}
J^*_\infty\left(x(k)\right) = \min_U \; & \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\
\text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \\
& x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \\
& x_0 = x(k)
\end{aligned}
$$

We assume $Q = Q^T \geq 0$, $R = R^T > 0$

Let us take a linear quadratic regulator with saturated inputs. At time $t = 0$ the plane is flying with deviation of 10 m of the desired altitude, i.e. $x_0 = [0; 0; 0; 10]$.
With the problem parameters : Sampling time 0.25 seconds, $Q = I$, $R = 10$

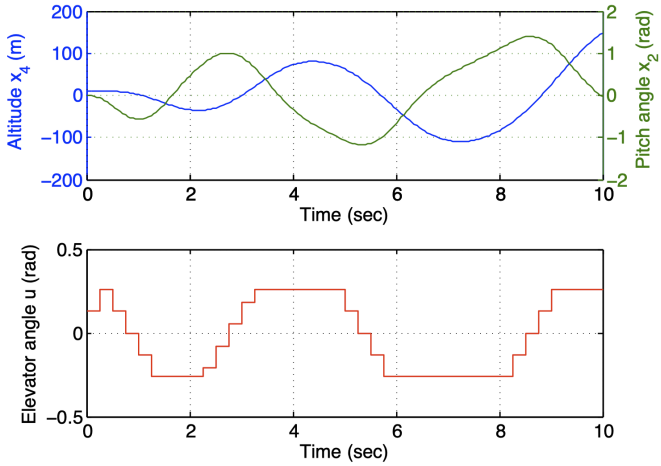


Figure 7.1

The closed-loop system is unstable. Applying LQR control and saturating the controller can lead to instability!

If we take an MPC controller with the input constraints : $|u_k| \leq 0.262$ (note that we have not applied the rate constraints). Our sampling time is 0.25 seconds, $Q = I$, $R = 10$ and $N = 10$.

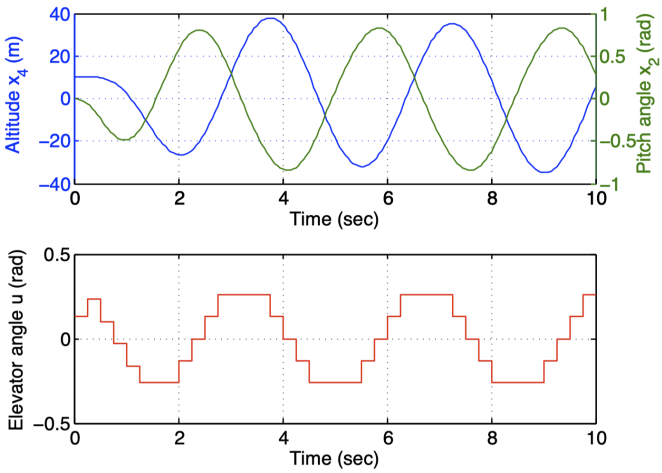


Figure 7.2

The MPC uses the knowledge that the elevator will saturate, but it does not consider the rate constraints. The system therefore does not converge to the desired steady-state but to a limit cycle.

Now if we take the MPC with the input constraints $|u_k| \leq 0.262$ and rate constraints $|\dot{u}_k| \leq 0.349$, approximated by $|u_i - u_{i-1}| \leq 0.349 T_s$. Our sampling time is 0.25 seconds, $Q = I$, $R = 10$ and $N = 10$.

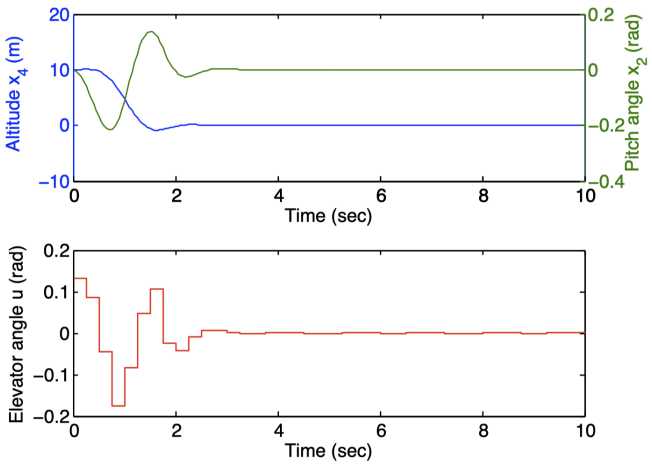


Figure 7.3

The MPC considers all constraints on the actuator :

- Closed-loop system is stable
- Efficient use of the control authority

If we increase the step, so at time $t = 0$ the plane is flying with a deviation of 100 m of the desired altitude, i.e. $x_0 = [0; 0; 0; 100]$. We notice that the pitch angle is too large during transient.

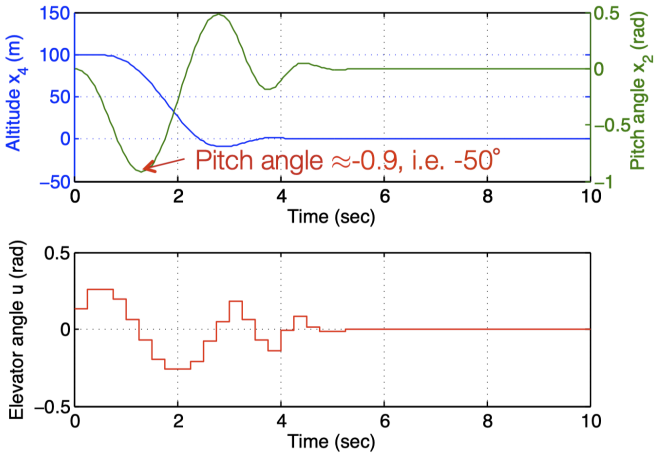


Figure 7.4

To avoid this, we add a state constraint for passenger comfort : $|x_2| \leq 0.349$

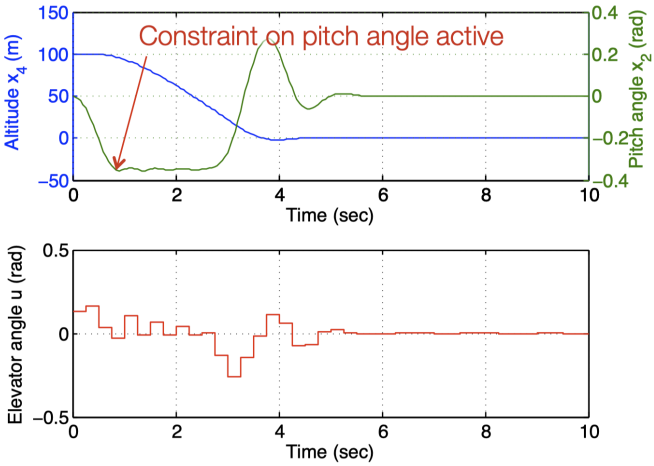


Figure 7.5

However, we need to choose our prediction horizon length carefully. The decrease in the prediction horizon causes loss of the stability properties.

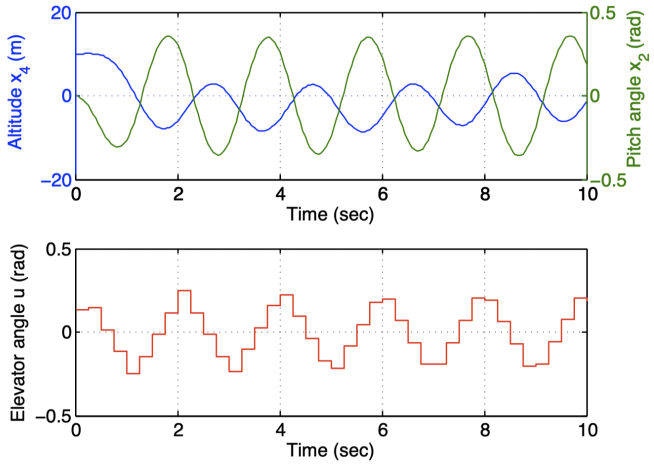


Figure 7.6: We have a prediction horizon of $N = 4$, instead of $N = 10$ as we did previously.

7.2 Loss of Feasibility & Stability

What can go wrong with a "standard" MPC?

- No feasibility guarantee, i.e., the MPC problem may not have a solution.
- No stability guarantee, i.e., the trajectories may not converge to the origin.

7.2.1 Example - Loss of Feasibility, Double Integrator

If we consider the double integrator :

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

subject to the input constraints

$$-0.5 \leq u(k) \leq 0.5$$

and the state constraints

$$\begin{pmatrix} -5 \\ -5 \end{pmatrix} \leq x(k) \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$

Compute a receding horizon controller with quadratic objective with

$$N = 3 \qquad P = Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = 10$$

The QP problem associated with the RHC is :

$$H = \begin{pmatrix} 13.5 & -10 & -0.5 \\ -10 & 22 & -10 \\ -0.5 & -10 & 31.5 \end{pmatrix} \quad F = \begin{pmatrix} -10.5 & 10 & -0.5 \\ -20.5 & 10 & 9.5 \end{pmatrix} \quad Y = \begin{pmatrix} 14.5 & 23.5 \\ 23.5 & 54.5 \end{pmatrix}$$

$$G = \begin{pmatrix} 0.5 & -1 & 0.5 \\ -0.5 & 1 & -0.5 \\ -0.5 & 0 & 0.5 \\ -0.5 & 0 & -0.5 \\ 0.5 & 0 & -0.5 \\ 0.5 & 0 & 0.5 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -0.5 & 0 & 0.5 \\ 0 & 0 & 0 \\ 0.5 & 0 & -0.5 \\ -0.5 & 0 & 0.5 \\ 0.5 & 0 & -0.5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad E = \begin{pmatrix} 0.5 & 0.5 \\ -0.5 & -0.5 \\ 0.5 & 0.5 \\ -0.5 & -0.5 \\ -0.5 & -0.5 \\ 0.5 & 0.5 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ -0.5 & -0.5 \\ -1 & -1 \\ 0.5 & 0.5 \\ -0.5 & -1.5 \\ 0.5 & 1.5 \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad w = \begin{pmatrix} 0.5 \\ 0.5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 0.5 \\ 0.5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 0.5 \\ 0.5 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$$

We then apply the MPC algorithm :

1. We **measure** the state $x(k)$ at time instance k
2. We **obtain** $U^*(x(k))$ by solving the CFTOC
3. **If** $U^*(x(k)) = \emptyset$ **then** the problem is "infeasible" $\rightarrow$ STOP
4. **Apply** the first element u_0^* of U^* to the system
5. **Wait** for the new sampling time $k+1$, **goto** 1.

Time step 1 :

$$x_0 = [-4; 3] \qquad u_0^*(x(0)) = -0.5$$

Time step 2 :

$$x_0 = [-1; 2.5] \qquad u_0^*(x(1)) = -0.5$$

Time step 3 :

$$x_0 = [1.5; 2] \qquad \text{Problem Infeasible}$$

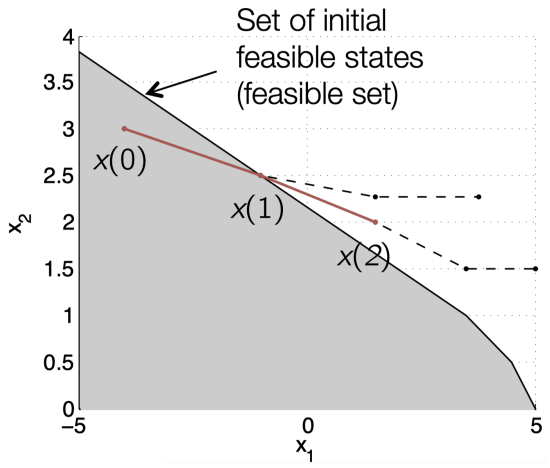


Figure 7.7

Depending on the initial conditions, the closed-loop trajectory may lead to states for which the optimization problem is infeasible.

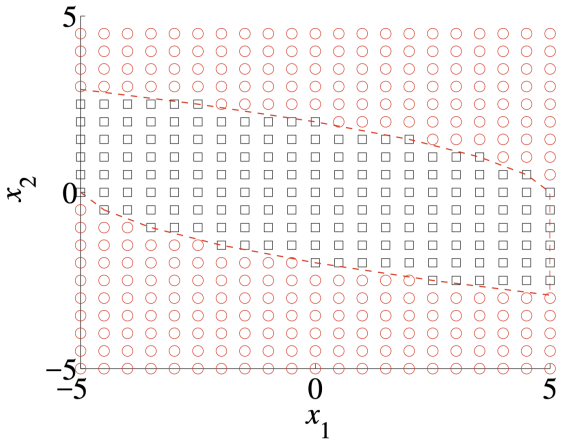


Figure 7.8

The boxes (circles) are the initial points leading (not leading) to feasible closed-loop trajectories.

7.2.2 Example - Feasibility and Stability are Functions of Tuning

If we take the unstable system ;

$$x(k+1) = \begin{pmatrix} 2 & 1 \\ 0 & 0.5 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(k)$$

with the input constraints :

$$-1 \leq u(k) \leq 1$$

and the state constraints :

$$\begin{pmatrix} -10 \\ -10 \end{pmatrix} \leq x(k) \leq \begin{pmatrix} 10 \\ 10 \end{pmatrix}$$

with the parameter : $Q = I_2$

Investigate the stability properties for different horizons N and weights R by solving the finite-horizon MPC problem in a receding horizon fashion.

1. $R = 10$, $N = 2$: All trajectories are unstable
2. $R = 2$, $N = 3$: Some trajectories are unstable
3. $R = 1$, $N = 4$: We have more stable trajectories

$*$: Initial points with convergent trajectories
$\circ$: Initial points that diverge

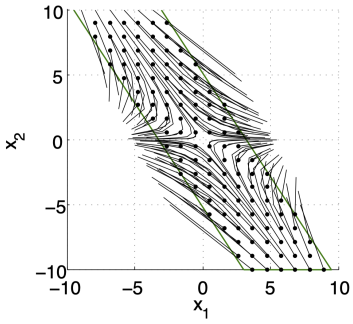


Figure 7.9: Case 1

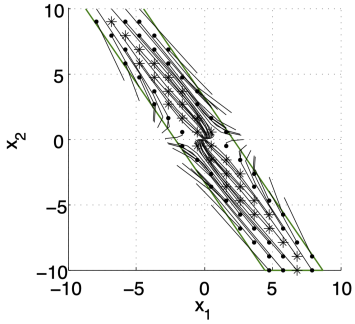


Figure 7.10: Case 2

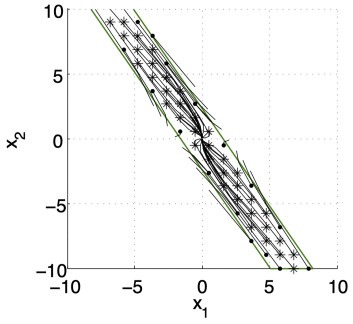


Figure 7.11: Case 3

The green lines denote the set of all feasible initial points. They depend on the horizon N but not the cost R. The parameters have complex effects and trajectories.

Summary - Feasibility & Stability

The problems originate from the use of a "short sighted" strategy. The finite horizon causes deviation between the open-loop prediction and the closed-loop system.

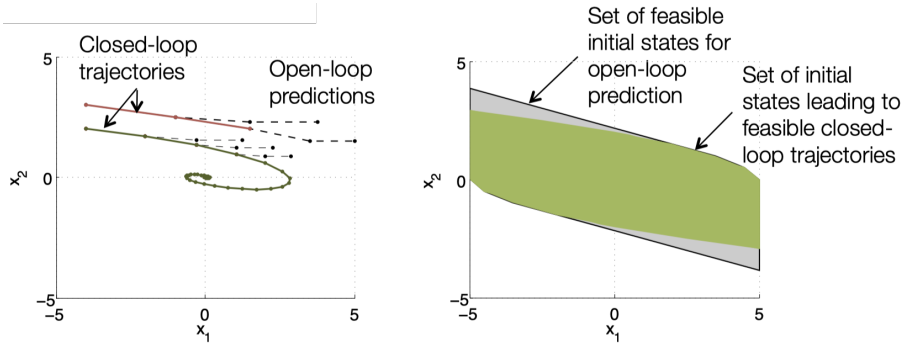


Figure 7.12

Ideally, we would solve the MPC problem with an infinite horizon, but that is computationally intractable. We therefore design a finite horizon problem such that it approximates the infinite horizon.

- Infinite Horizon
  If we solve the RHC problem for $N = \infty$ (as done for LQR), then the open loop trajectories are the same as the closed loop trajectories. Hence
  - If the problem is feasible, the closed-loop trajectories will always be feasible
  - If the cost is finite, then the states and inputs will converge asymptotically to the origin

- Finite Horizon
  RHX is the "short-sighted" strategy approximating an infinite horizon controller. However,
  - **Feasibility** - After some steps the finite horizon optimal control problems may become infeasible (infeasibility occurs without disturbances and model mismatch).
  - **Stability** - The generated control inputs may not lead to trajectories that converge to the origin.

7.2.3 Feasibility & Stability in MPC - Solution

The main idea here is that we introduce a terminal cost and contraints to explicitly ensure feasibility and stability.

$$J^*\left(x(k)\right) = \min_U \left[l_f\left(x_N\right) + \sum_{i=0}^{N-1} l\left(x_i, u_i\right) \right] \qquad \text{Terminal Cost}$$

$$\begin{aligned} \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \qquad i = 0, \dots, N-1 \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \qquad \text{Terminal Constraint} \\ & x_0 = x(k) \end{aligned}$$

Where we choose $l_f(\cdot)$ and $\mathcal{X}_f$ to mimic an infinite horizon.

7.3 Feasibility & Stability Guarantees in MPC

Feasibility & Stability - Proof

Main steps :

- We prove recursive feasibility by showing the existence of a feasible control sequence at all time instants when starting from a feasible initial point.
- We prove stability by showing that the optimal cost function is a Lyapunov function.

First, let's remind ourselves about Lyapunov stability :

Definition 7.3.1. Lyapunov Function
Consider the equilibrium point $x = 0$. Let $\Omega \subset \mathbb{R}^n$ be a closed and bounded set for containing the origin. A function $V : \mathbb{R}^n \to \mathbb{R}$. continuous at the origin, finite for every $x \in \Omega$, and such that

$$\begin{aligned} V(0) = 0 \text{ and } V(x) > 0, \forall x \in \Omega n \{0\} \\ V\left(g(x)\right) - V(x) \leq -\alpha(x) \ \forall x \in \Omega n \{0\} \end{aligned}$$

where $\alpha : \mathbb{R}^n \to \mathbb{R}$ is continuous positive definite, is called a **Lyapunov function**.

Theorem 7.3.1. ***Asymptotic Stability***
If a system admits a Lyapunov function $V(x)$, then $x = 0$ is ***asymptotically stable*** *in Ω.*

We have two possible cases that we need to take into account for our proof :

1. Terminal constraint at zero $x_N = 0$

2. Terminal constraint in some (convex) set : $x_N \in \mathcal{X}_f$

We use the general notation :

$$J^*(x(k)) = \min_U \underbrace{l_f(x_N)}_{\text{terminal cost}} + \sum_{i=0}^{N-1} \underbrace{l(x_i, u_i)}_{\text{stage cost}}$$

7.3.1 Stability of MPC - Zero Terminal State Constraint

We take the case where the terminal constraint : $x_N \in \mathcal{X}_f = 0$

- We assume feasibility of $x(k)$ and let $\{u_0^*, \ldots, u_{N-1}^*\}$ be the optimal control sequence computed at $x(k)$ and let $\{x(k), x_1^*, \ldots, x_N^*\}$ be the corresponding state trajectory
- We apply $u(k) = u_0^*$ and let the system evolve into $x(k+1) = Ax(k) + Bu(k)$
- At $x(k+1) = x_1^*$ the control sequence $\tilde{U} = \{u_1^*, \ldots, u_{N-1}^*\}$ is feasible (apply 0 control input $\rightarrow A \underbrace{x_N^*}_{=0} + B \cdot 0 = 0$)

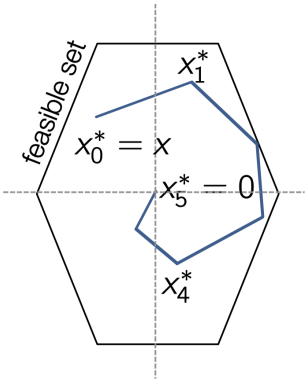


Figure 7.13

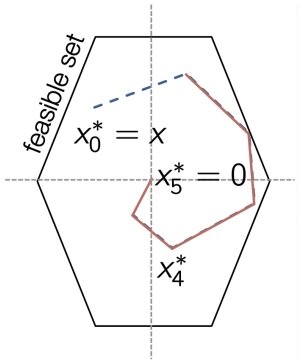


Figure 7.14

$\Rightarrow$ We have recursive feasibility ✓

Goal : Show that $J^*(x(k+1)) < J^*(x(k)) \quad \forall x(k) \neq 0$

$$J^*(x(k)) = \underbrace{l_f(x_N^*)}_{=0} + \sum_{i=0}^{N-1} l(x_i^*, u_i^*)$$

$$\begin{aligned} J^*(x(k+1)) \leq \tilde{J}(x(k+1)) &= \sum_{i=0}^{N-1} l(x_i^*, u_i^*) + l(x_N^*, 0) \\ &= \sum_{i=0}^{N-1} l(x_i^*, u_i^*) - l(x_0^*, u_0^*) + l(x_N^*, 0) \\ &= J^*(x(k)) - \underbrace{l(x(k), u_0^*)}_{\substack{\text{subtract cost} \\ \text{at stage } k}} + \underbrace{l(0,0)}_{\substack{\text{add cost for} \\ \text{staying at 0}}} \end{aligned}$$

$\Rightarrow$ $J^*(x(k))$ is a Lyapunov function $\rightarrow$ We have Lyapunov stability ✓

7.3.2 Example - Impact of Horizon with Zero Terminal Constraint

We have a system with the dynamics :

$$x(k+1) = \begin{pmatrix} 1.2 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u(k)$$

and the constraints :

$$\mathcal{X} \coloneqq \{x| -50 \leq x_1 \leq 50, \ -10 \leq x_2 \leq 10\} = \{x|A_x x \leq b_x\}$$
$$\mathcal{U} \coloneqq \{u|\|u\|_\infty \leq 1\} = \{u|A_u u \leq b_u\}$$

and the stage cost :

$$l\left(x_i, u_i\right) \coloneqq x_i^T \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x_i + u_i^T u_i$$

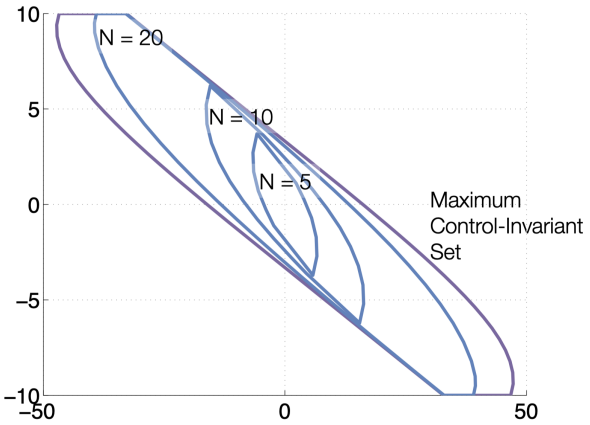


Figure 7.15

Remark : As we see in the example above, the horizon can have a strong impact on the region of attraction.

7.3.3 General Terminal Sets

Problem : The terminal constraint $x_N = 0$ reduces the size of the feasible set.
Goal : Use the convex set $\mathcal{X}_f$ to increase the region of attraction.

If we consider the double integrator :

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

subject to the input constraints

$$-0.5 \leq u(k) \leq 0.5$$

and the state constraints

$$\begin{pmatrix} -5 \\ -5 \end{pmatrix} \leq x(k) \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$

and the parameters

$$N = 5 \qquad Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = 10$$

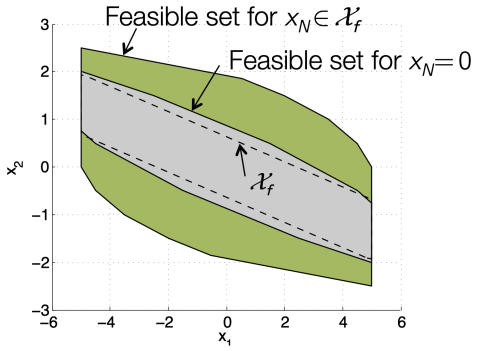


Figure 7.16

Our goal is to generalize the proof to the constraint $x_N \in \mathcal{X}_f$.

Definition 7.3.2. Positively Invariant Set

A set $\mathcal{O}$ is called **positively invariant** for the system $x(k+1) = g_{cl}(x(k))$, if

$$x(0) \in \mathcal{O} \Rightarrow x(k) \in \mathcal{O} \quad \forall k \in \mathbb{N}_+$$

The positively invariant set that contains every closed positively invariant set is called the **maximal positively invariant set** $\mathcal{O}_\infty$.

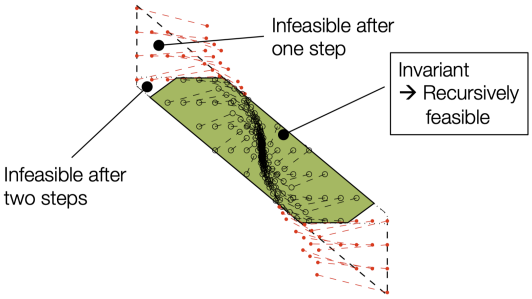


Figure 7.17

We take three assumptions for the proof stability of our MPC :

1. Stage cost is positive definite, i.e., it is strictly positive and only zero at the origin.
2. The terminal set is **invariant** under the local control law $\kappa_f(x_i)$:

   $$x_{i+1} = Ax_i + B\kappa_f(x_i) \in \mathcal{X}_f \quad \text{for all } x_i \in \mathcal{X}_f$$

   All state and input **constraints are satisfied** in $\mathcal{X}_f$:

   $$\mathcal{X}_f \subset \mathcal{X},\ \kappa_f(x_i) \in \mathcal{U} \quad \text{for all } x_i \in \mathcal{X}_f$$

3. The terminal cost is a continuous **Lyapunov function** in the terminal set $\mathcal{X}_f$ and satisfies:

   $$l_f(x_{i+1}) - l_f(x_i) \leq -l(x_i, \kappa_f(x_i)) \quad \text{for all } x_i \in \mathcal{X}_f$$

Under those three assumptions we can state the following :

Theorem 7.3.2. ***Closed-Loop System Stability***

The closed-loop system under the MPC control law $u_0^*(x)$ is asymptotically stable and the set $\mathcal{X}_N$ is positive invariant for the system

$$x(k+1) = Ax(k) + Bu_0^*(x(k))$$

Stability of MPC - Outline of the Proof

- We assume feasibility of $x(k)$ and let $\{u_0^*, \dots, u_{N-1}^*\}$ be the optimal control sequence computed at $x(k)$ and $\{x(k), x_1^*, \dots, x_N^*\}$ the corresponding state trajectory.

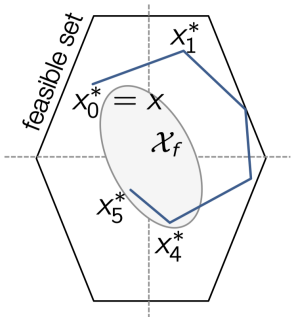


Figure 7.18

- At $x(k+1) = x_1^*$, the control sequence $\tilde{U} = \{u_1^*, u_2^*, \dots, \kappa_f(x_N^*)\}$ is feasible :

$$x_N^* \text{ is in } \mathcal{X}_f \rightarrow \kappa_f(x_N^*) \text{ is feasible and } Ax_N^* + B\kappa_f(x_N^*) \text{ in } \mathcal{X}_f$$

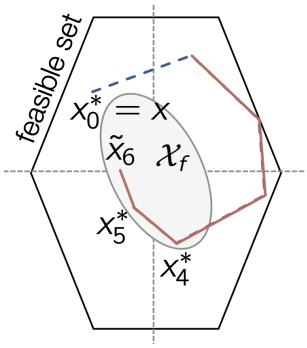


Figure 7.19

$\Rightarrow$ The terminal constraint provides recursive feasibility.

$$J^*(x(k)) = \sum_{i=0}^{N-1} l(x_i^*, u_i^*) + l_f(x_N^*)$$

At $x(k+1) = x_1^*$, $\tilde{U} = \{u_1^*, u_2^*, \dots, \kappa_f(x_N^*)\}$ is feasible and sub-optimal.

$$\begin{aligned} J^*(x(k+1)) &\leq \sum_{i=1}^{N-1} l(x_i^*, u_i^*) + l(x_N^*, \kappa_f(x_N^*)) + l_f(Ax_N^* + B\kappa_f(x_N^*)) \\ &= \underbrace{\sum_{i=0}^{N-1} l(x_i^*, u_i^*)}_{J^*(x(k)) - l_f(x_N^*)} - l(x_0^*, u_0^*) + l(x_N^*, \kappa_f(x_N^*)) + l_f(Ax_N^* + B\kappa_f(x_N^*)) \\ &= J^*(x(k)) - l(x(k), u_0^*) + \underbrace{l_f(Ax_N^* + B\kappa_f(x_N^*)) - l_f(x_N^*) + l(x_N^*, \kappa_f(x_N^*))}_{\leq 0 \text{ from assumption 3}} \end{aligned}$$

$$\hookrightarrow \quad J^*(x(k+1)) - J^*(x(k)) \leq l(x(k), u_0^*) \qquad l(x,u) > 0 \text{ for } x, u \neq 0$$

$J^*(x)$ is a **Lyapunov function**.
$\Rightarrow$ The closed-loop system under the MPC control law is asymptotically stable.

Choice of Terminal Sets & Cost - Linear System, Quadratic Cost

- Design an unconstrained LQR control law :

$$F_\infty = -\left(B^T P_\infty B + R\right)^{-1} B^T P_\infty A$$

where P_∞ is the solution to the discrete-time algebraic Riccati equation :

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B \left(B^T P_\infty B + R\right)^{-1} B^T P_\infty A$$

- Choose the terminal weight $P = P_\infty$
- Choose the terminal set $\mathcal{X}_f$ to be the maximum invariant set for the closed-loop system $x_{k+1} = (A + BF_\infty)\, x_k$:

$$x_{k+1} = Ax_k + BF_\infty(x_k) \in \mathcal{X}_f, \text{ for all } x_k \in \mathcal{X}_f$$

All state and input **constraints are satisfied** in $\mathcal{X}_f$:

$$\mathcal{X}_f \subseteq \mathcal{X} \quad F_\infty x_k \in \mathcal{U} \quad \text{for all } x_k \in \mathcal{X}_f$$

1. The stage cost is a positive definite function.
2. By construction the terminal set is **invariant** under the local control law $\kappa_f(x) = F_\infty x$
3. The terminal cost is a continuous **Lyapunov function** in the terminal set $\mathcal{X}_f$ and satisfies:

$$\begin{aligned} x_{k+1}^T P x_{k+1} - x_k^T P x_k &= x_k^T \left(-P_\infty + A^T P_\infty A + F_\infty^T B^T P_\infty A - F_\infty^T R F_\infty\right) x_k \\ &= x_k^T \left(-P_\infty + A^T P_\infty A - A^T P_\infty B \left(B^T P_\infty B + R\right)^{-1} B^T P_\infty A - F_\infty^T R F_\infty\right) x_k \\ &= -x_k^T \left(Q + F_\infty^T R F_\infty\right) x_k \qquad \text{(with LQR we have equality)} \end{aligned}$$

All of the assumptions of the feasibility and stability theorem are verified.

Example - Unstable Linear System

With the system dynamics

$$x(k+1) = \begin{pmatrix} 1.2 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u_k$$

and the constraints

$$\begin{aligned} \mathcal{X} &:= \{x | -50 \le x_1 \le 50,\ -10 \le x_2 \le 10\} = \{x | A_x x \le b_x\} \\ \mathcal{X} &:= \{u | \|u\|_\infty \le 1\} = \{u | A_u u \le b_u\} \end{aligned}$$

and the stage cost

$$l\left(x_i, u_i\right) := x_i^T \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x_i + u_i^T u_i$$

and a horizon length of $N = 10$.

1. Compute the optimal LQR controller and cost matrices : F_∞ and 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 constraints :

$$\mathcal{X}_{cl} := \left\{ x \,\middle|\, \begin{pmatrix} A_x \\ A_u F_\infty \end{pmatrix} x \leq \begin{pmatrix} b_x \\ b_u \end{pmatrix} \right\}$$

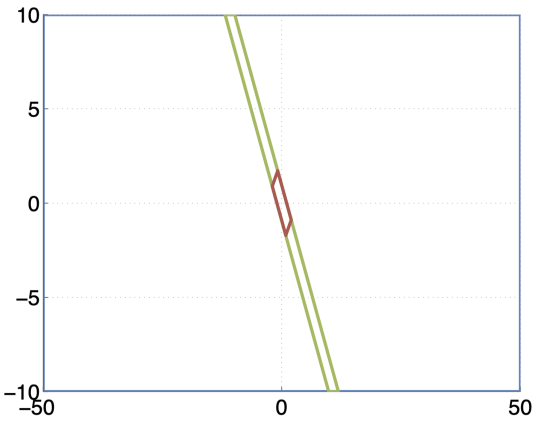


Figure 7.20

We show the closed-loop behavior :

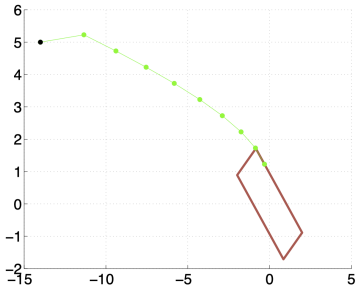


Figure 7.21

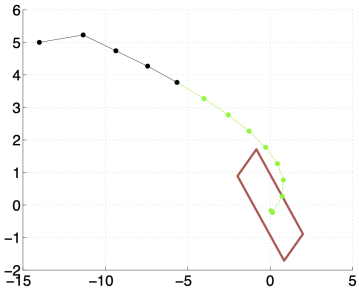


Figure 7.22

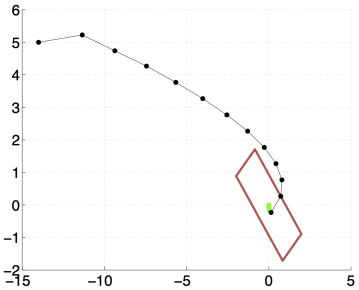


Figure 7.23

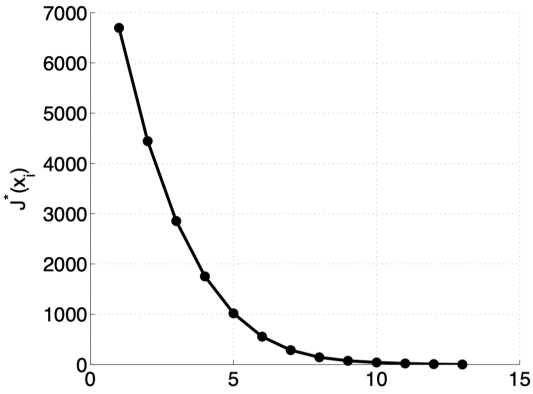


Figure 7.24: Predicted cost of MPC

Summary - Choice of Terminal Set & Cost

- Terminal constraint provides a sufficient condition for feasibility and stability

- The region of attraction without terminal constraint may be larger than for MPC with terminal constraint but the characterization of the region of attraction is extremely difficult.
- $\mathcal{X}_f = 0$ is the simplest choice but a small region of attraction for a small N
- Solutions are available for linear system with a quadratic cost
- In practice, we prefer to enlarge the horizon and check stability by sampling
- With a larger horizon N, the region of attraction approaches a maximum control invariant set

7.3.4 Example - Short Horizon

We take an MPC controller with input constraints $|u_k| \leq 0.262$ and rate constraints $|\dot{u}_k| \leq 0.349$ approximated by $|u_i - u_{i-1} \leq 0.349 T_s|$. We also have the following problem parameters :

Sampling time : 0.25 sec $\qquad Q = I \qquad R = 10 \qquad N = 4$

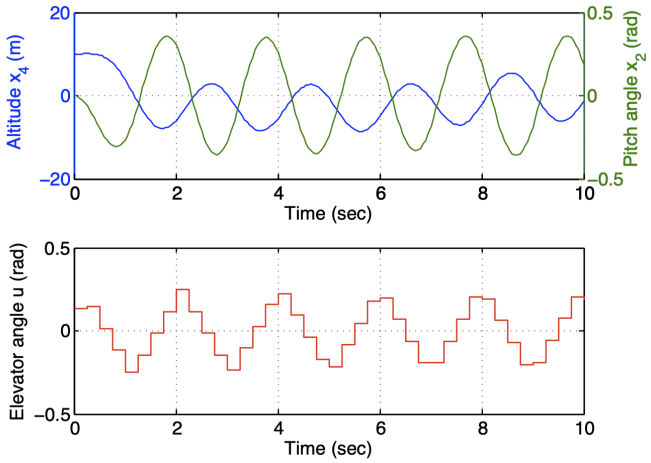


Figure 7.25

We notice that the decrease in the prediction horizon causes the loss of the stability properties. The inclusion of a terminal cost and constraint provides stability.

7.4 Extension to Non-Linear MPC

Let us consider the non-linear system dynamics : $x(k+1) = g\left(x(k), u(k)\right)$

$$J^*\left(x(k)\right) = \min_U \left[l_f\left(x_N\right) + \sum_{i=0}^{N-1} l\left(x_i, u_i\right) \right]$$

$$\begin{aligned} \text{subject to} \quad & x_{i+1} = g(x_i, u_i) \qquad i = 0, \ldots, N-1 \\ & x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(k) \end{aligned}$$

- The presented assumptions on the terminal set and cost did not rely on linearity

- Lyapunov stability is a general framework to analyse the stability of non-linear dynamic systems

→ The results can therefore be directly extended to non-linear systems

However, computing the sets $\mathcal{X}_f$ and function l_f can be very difficult!

Summary

- Finite-horizon MPC may not be stable!
- Finite-horizon MPC may not satisfy constraints for all time!
- We know that an infinite horizon provides stability and invariance.
- We "fake" infinite-horizon by forcing the final state to be in an invariant set for which there exists an invariance-inducing controller, whose infinite-horizon cost can be expressed in closed-form.
- These ideas extend to non-linear systems, but the sets are difficult to compute.

Chapter 8

Practical MPC

8.1 Practical Issues with MPC

8.1.1 Tracking

Regulation to the origin is classic MPC problem. The common task in practice is tracking of non-zero output set points.

Example - Double Integrator

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

$$\begin{aligned} -0.5 \leq u \leq 0.5 \\ -5 \leq x_i \leq 5 \qquad i = 1, 2, \dots \end{aligned}$$

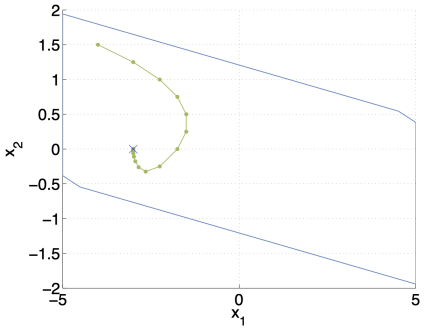


Figure 8.1

↪ We want to use MPC for tracking.

8.1.2 Disturbance Rejection

Constant disturbance causes offset from the origin / the desired set point.

Example - Double Integrator

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

$$\begin{aligned} d &= 0.2 \\ -0.5 \leq u \leq 0.5 \\ -5 \leq x_i \leq 5 \qquad i = 1, 2, \dots \end{aligned}$$

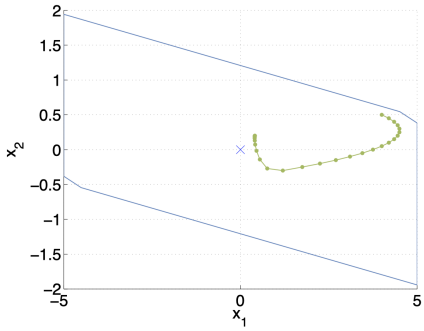


Figure 8.2

↪ We want to remove the offset such that the system converges to the desired set point.

8.1.3 Feasible Set

Constraints restrict the set of states for which the optimization problem is feasible. The MPC controller is only defined in the feasible set, where a solution exists.

Example - Double Integrator

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

$$-0.5 \leq u \leq 0.5$$
$$-5 \leq x_i \leq 5 \qquad i = 1, 2, \dots$$

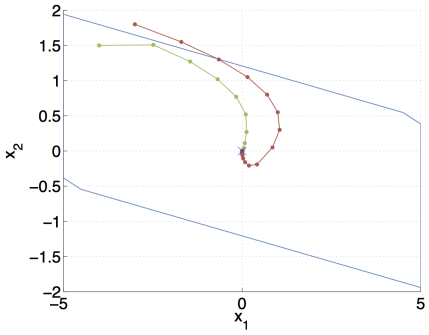


Figure 8.3

↪ We want the feasible set to be as large as possible.

We will learn the following in this chapter :

- Understand the difference between MPC for tracking and regulation
- Formulate the MPC tracking problem
- Address the constant disturbances to achieve offset-free tracking / regulation
- Analyse MPC without the terminal set
- Formulate the soft-constrained MPC problem and understand its properties

8.2 Reference Tracking

Let us consider the linear system model

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

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$.

- Simplifying assumption : The state can be measured
- Constraints : $x \in \mathcal{X}$, $u \in \mathcal{U}$ with the constraint sets

$$\mathcal{X} = \{x | G_x x \leq h_x\} \qquad \mathcal{U} = \{u | G_u u \leq h_u\}$$

Goal : Track the given reference r such that $z(k) = Hx(k) \to r \in \mathbb{R}^{n_r}$ as $k \to \infty$.

Remark : Using this framework we can track a sequence of contraints targets, i.e. a piecewise constant reference. We will not cover time-varying references in this course.

8.2.1 Introduction

The standard MPC problem regulates the system to the origin :

$$U^*\left(x(k)\right) \coloneqq \min_{U_k} \left[l_f\left(x_N\right) + \sum_{i=0}^{N-1} l\left(x_{k+i}, u_{k+i}\right) \right]$$

$$\begin{aligned}
\text{subject to} \quad & x_k = x(k) && \text{Measurement} \\
& x_{k+i+1} = Ax_{k+i} + Bu_{k+i} && \text{System Model} \\
& x_{k+i} \in \mathcal{X} && \text{State Constraints} \\
& u_{k+i} \in \mathcal{U} && \text{Input Constraints} \\
& U_k = \{u_k, u_{k+1}, \ldots, u_{k+N-1}\} && \text{Optimization Variables}
\end{aligned}$$

Common Task : Tracking of non-zero references.

↪ How can we modify the MPC problem to achieve tracking?

Regulation to origin :

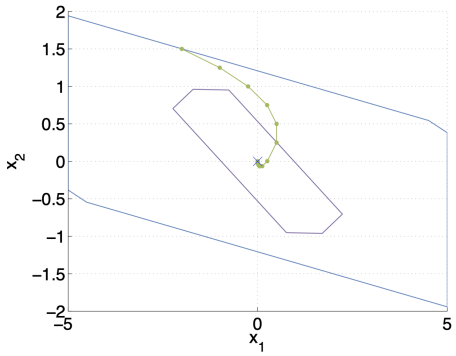


Figure 8.4

Non-zero target :

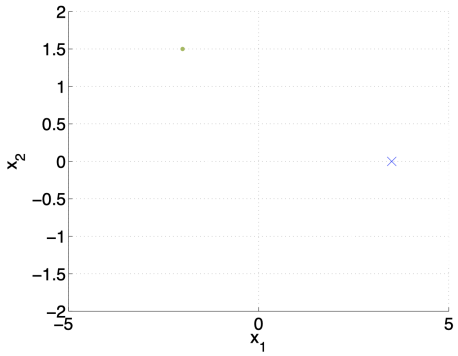


Figure 8.5

Given a target r, how do we obtain a corresponding state target x_s?

Regulation to origin :

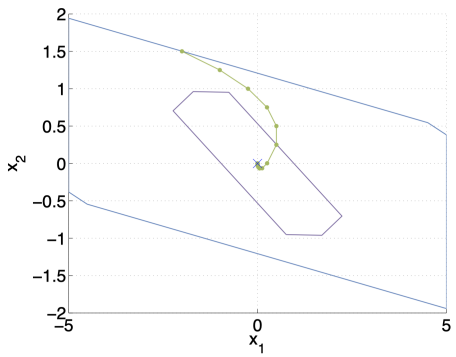


Figure 8.6

Non-zero target :

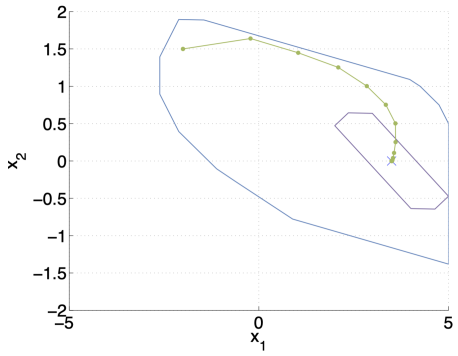


Figure 8.7

How can we adapt the MPC cost to control the system to the target state?

How do we choose the terminal set and when is the MPC problem feasible with respect to the target?

8.2.2 The Steady-State Target Problem

Target State Corresponding to Output Reference

- The reference is achieved by the target state x_s if $z_s = Hx_s = r$
- The target state should be a steady-state, such that there exists an input that keeps the system at target, i.e. $x_s = Ax_s + Bu_s$

$\hookrightarrow$ Target condition :

$$\begin{matrix} x_s = Ax_s + Bu_s \\ Hx_s = r \end{matrix} \quad \Leftrightarrow \quad \underbrace{\begin{pmatrix} I - A & -B \\ H & 0 \end{pmatrix}}_{(n_x+n_r)\times(n_x+n_u)} \begin{pmatrix} x_s \\ u_s \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$$

Steady-State Target Problem

- In the presence of constraints : (x_s, u_s) has to satisfy the state and input constraints.
- In the case of multiple feasible inputs u_s, we compute the "cheapest" steady-state (x_s, u_s) corresponding to the reference r :

$$\begin{aligned} & \min u_s^T R_s u_s \\ \text{subject to} \quad & \begin{pmatrix} I - A & -B \\ H & 0 \end{pmatrix} \begin{pmatrix} x_s \\ u_s \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix} \\ & x_s \in \mathcal{X} \\ & u_s \in \mathcal{U} \end{aligned}$$

In general, we assume that the target problem is feasible. If no solution exists, we compute the reachable set point that is "closest" to r :

$$\begin{aligned} & \min \left(Hx_s - r\right)^T Q_s \left(Hx_s - r\right) \\ \text{subject to} \quad & x_s = Ax_s + Bu_s \\ & x_s \in \mathcal{X} \\ & u_s \in \mathcal{U} \end{aligned}$$

8.2.3 MPC for Reference Tracking

We now use our MPC to bring the system to a desired steady-state condition (x_s, u_s) yielding the desired output $z(k) \to r$.

The MPC is designed as follows :

$$\begin{aligned} & \min \left[\|z_n - Hx_s\|_{P_z}^2 + \sum_{i=0}^{N-1} \|z_i - Hx_s\|_{Q_z}^2 + \|u_i - u_s\|_R^2 \right] \\ \text{subject to} \quad & \text{[model, constraints]} \\ & x_0 = x(k) \end{aligned}$$

Delta-Formulation for Tracking

Idea : Treat set point tracking as regulation problem with a coordinate transformation.

- We define deviation variables that (in the linear case) satisfy the same model equations :

$$\begin{aligned} \Delta x &= x - x_s \\ \Delta u &= u - u_s \end{aligned} \quad \Rightarrow \quad \begin{aligned} \Delta x_{k+1} &= x_{k+1} - x_s \\ &= Ax_k + bu_k - (Ax_s + Bu_s) \\ &= A\Delta x_k + B\Delta u_k \end{aligned}$$

- Constraints for deviation variables :

$$\begin{aligned} G_x x &\leq h_x \\ G_u u &\leq h_u \end{aligned} \quad \Rightarrow \quad \begin{aligned} G_x \Delta x &\leq h_x - G_x x_s \\ G_u \Delta u &\leq h_u - G_u u_s \end{aligned}$$

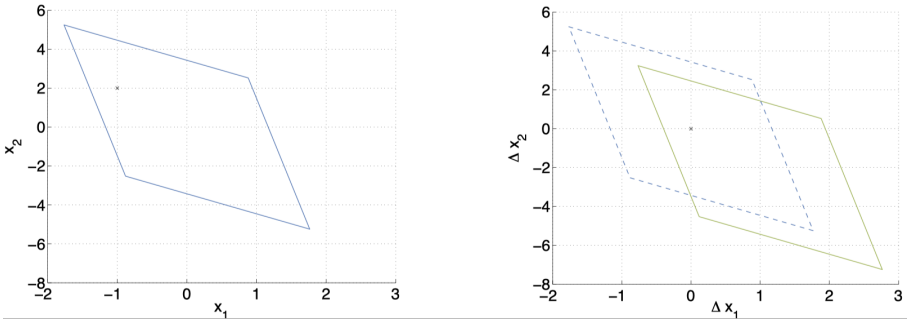


Figure 8.8

MPC Problem for Tracking in Delta Formulation

- We obtain a target steady-state corresponding to the reference r[1].
- The initial state is $\Delta x(k) = x(k) - x_s$
- We then apply the regulation problem to the new system in delta-formulation :

$$\begin{aligned} & \min \sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i + V_f(\Delta x_N) \\ \text{subject to} \quad & \Delta x_0 = \Delta x(k) \\ & \Delta x_{i+1} = A\Delta x_i + B\Delta u_i \\ & G_x \Delta x_i \leq h_x - G_x x_s \\ & G_u \Delta u_i \leq h_u - G_u u_s \\ & \Delta x_N \in \mathcal{X}_f \end{aligned}$$

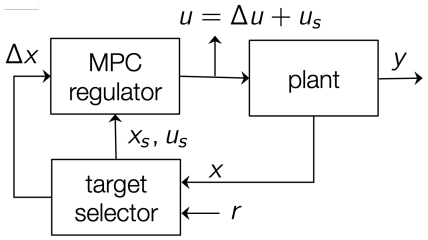


Figure 8.9

[1] If the target steady-state is uniquely defined by the reference, we can also include the target condition as a constraint in the MPC problem.

- We find the optimal sequence of ΔU^*
- The input applied to the system is $u_0^* = \Delta u_0^* + u_s$

MPC for Tracking - Convergence

We assume that the target is feasible with $x_s \in \mathcal{X}$, $u_s \in \mathcal{U}$ and choose the terminal weight $V_f(x)$ and constraint $\mathcal{X}_f$ as in the regulation case satisfying :

- $\mathcal{X}_f \subseteq \mathcal{X}$, $Kx \in \mathcal{U}$ for all $x \in \mathcal{X}_f$
- $V_f\left(x(k+1)\right) - V_f\left(x(k)\right) \leq -l\left(x(k), Kx(k)\right)$ for all $x \in \mathcal{X}_f$

If in addition the target reference x_s, u_s is such that

$$x_s \oplus \mathcal{X}_f \subseteq \mathcal{X},\ K\Delta x + u_s \in \mathcal{U} \text{ for all } \Delta x \in \mathcal{X}_f$$

then the closed-loop system converges to the target reference, i.e. $x(k) \to x_s$ and therefore $z(k) = Hx(k) \to r$ for $k \to \infty$.

Proof. We choose the local control law $\Delta u = K\Delta x$

- Invariance under the local control law is directly inherited from the regulation case.
- The constraint satisfaction is provided by extra conditions :
  - $x_s \oplus \mathcal{X}_f \subseteq \mathcal{X} \quad \to \quad x \in \mathcal{X} \forall \Delta x = x - x_s \in \mathcal{X}_f$
  - $K\Delta x + u_s \in \mathcal{U} \forall \Delta x \in \mathcal{X}_f \quad \to \quad u \in \mathcal{U}$
- From the asymptotic stability of the regulation problem : $\Delta x(k) \to 0$ for $k \to \infty$

□

MPC for Tracking - Terminal Set

For the following consideration, we consider only the state constraints :

Regulation case :

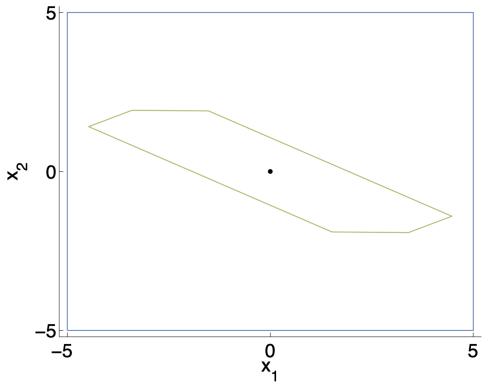


Figure 8.10

Tracking using a shifted terminal set :

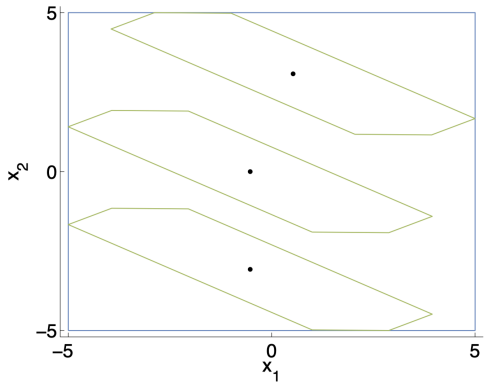
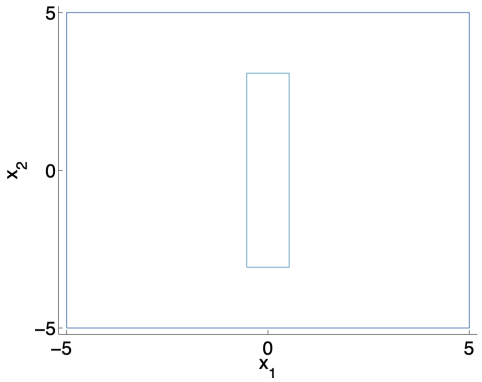


Figure 8.11

Set of feasible targets :



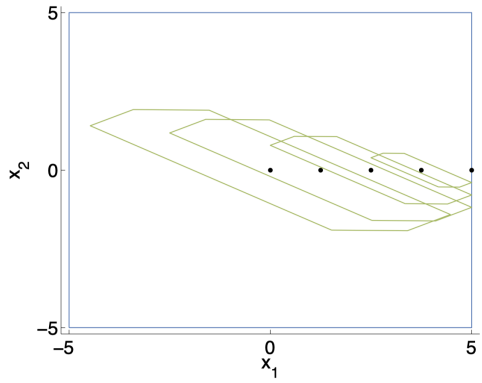
- Blue : State constraints
- Green : Terminal set

↪ The set of feasible targets may be significantly reduced.

Figure 8.12

We can enlarge the set of feasible targets by scaling the terminal set for regulation :

Set of feasible targets :



- We scale the terminal set by a scaling factor α, i.e. $\mathcal{X}_f^{\text{scaled}} = \alpha \mathcal{X}_f$
- Invariance is maintained. If $\mathcal{X}_f$ is invariant, then $\alpha \mathcal{X}_f$ is as well.
- We choose the scaling factor α such that the state and input constraints are still satisfied.

↪ Scaling is dependent on the target

Figure 8.13

All targets x_s, $u_s \in \mathcal{X} \times \mathcal{U}$ are feasible for constraints.

Remark : The steady-state condition still limits the set of admissible targets.

For targets at the boundary of the constraints : $x_N = x_s$, which corresponds to a zero terminal set in the regulation case.

Remarks :

- So far we have assumed that we can measure each state directly and the goal is $z(k) = Hx(k) \to r$ for $k \to \infty$
- In practice, the state cannot always be measured directly :

$$y(k) = Cx(k)$$

  i.e. the measured output $y(k) \in \mathbb{R}^{n_y}$ is a linear combination of the states.

- This is typically handled by designing a state estimator and running the MPC based on the estimated states.

- This approach is also used in offset.free tracking discussed in the following, where we assume the controlled variables to be a linear combination of measured outputs :

$$z(k) = Hy(k) \to r \quad \text{for} \quad k \to \infty$$

8.2.4 MPC for Reference Tracking without Offset

Constant Disturbances

Let us take the case where a constant disturbance is acting on the system causing the system trajectory to deviate from the nominal dynamics :

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

Objective : If the system is stabilised in the presence of the disturbance then it converges to the set point with zero offset.

Recall :

- In the classic unconstrained control we introduce an integrating mode to remove the offset.
- The input constraints and actuator limitations require anti-windup techniques.

Our approach in constrained control :

- We model the disturbance.
- We use the measurements and model to estimate the state and disturbance.
- We find the control inputs that use the disturbance estimate to remove the offset.

Augmented Model

We incorporate the disturbance model assuming the integral disturbance dynamics :

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k \\ d_{k+1} &= d_k \qquad \to \quad \text{disturbance is constant} \\ y_k &= Cx_k + C_d d_k \end{aligned}$$

with $d \in \mathbb{R}^{n_d}$.

The only restriction on choice of B_d, C_d : Observability of the augmented model.

The augmented system is observable if and only if (A, C) is observable and

$$\begin{pmatrix} A - I & B_d \\ C & C_d \end{pmatrix} \text{ has full column rank, i.e. } \mathbf{rank} = n_x + n_d$$

$\Rightarrow$ The maximal dimension of the disturbance : $n_d \leq n_y$

Intuition : At steady-state :

$$\begin{pmatrix} A - I & B_d \\ C & C_d \end{pmatrix} \begin{pmatrix} x_s \\ d_s \end{pmatrix} = \begin{pmatrix} 0 \\ y_s \end{pmatrix}$$

and given y_s, d_s must be uniquely defined.

Linear State Estimation

The state observer for the augmented model is :

$$\begin{pmatrix}\hat{x}(k+1)\\ \hat{d}(k+1)\end{pmatrix} = \begin{pmatrix}A & B_d\\ 0 & I\end{pmatrix}\begin{pmatrix}\hat{x}(k)\\ \hat{d}(k)\end{pmatrix} + \begin{pmatrix}B & 0\end{pmatrix}u(k) + \begin{pmatrix}L_x\\ L_d\end{pmatrix}\left[-y(k) + C\hat{x}(k) + C_d\hat{d}(k)\right]$$

where $\hat{x}$, $\hat{d}$ are estimates of the state and disturbance, y is the measured output.

Error dynamics :

$$\begin{aligned}\begin{pmatrix}x(k+1)-\hat{x}(k+1)\\ d(k+1)-\hat{d}(k+1)\end{pmatrix} &= \begin{pmatrix}A & B_d\\ 0 & I\end{pmatrix}\begin{pmatrix}x(k)-\hat{x}(k)\\ d(k)-\hat{d}(k)\end{pmatrix} - \begin{pmatrix}L_x\\ L_d\end{pmatrix}\left[C\hat{x}(k) + C_d\hat{d}(k) - Cx(k) - C_d d(k)\right]\\ &= \left[\begin{pmatrix}A & B_d\\ 0 & I\end{pmatrix} + \begin{pmatrix}L_x\\ L_d\end{pmatrix}\begin{pmatrix}C & C_d\end{pmatrix}\right]\begin{pmatrix}x(k)-\hat{x}(k)\\ d(k)-\hat{d}(k)\end{pmatrix}\end{aligned}$$

$\hookrightarrow$ We choose $L = \begin{pmatrix}L_x\\ L_d\end{pmatrix}$ such that the error dynamics are stable and converge to zero, i.e. the estimator is asymptotically stable.

Offset-Free Tracking

Lemma 8.2.1. If we suppose that the observer is asymptotically stable and the number of outputs n_y equals the dimension of the constant disturbance n_d. The observer steady-state satisfies :

$$\begin{pmatrix}A-I & B\\ C & 0\end{pmatrix}\begin{pmatrix}\hat{x}_\infty\\ u_\infty\end{pmatrix} = \begin{pmatrix}-B_d\hat{d}_\infty\\ y_\infty - C_d\hat{d}_\infty\end{pmatrix}$$

where y_∞ and u_∞ are the steady-state measured outputs and inputs.

Goal : We want to track the constant reference r, i.e. $z(k) = Hy(k) \to r$ for $k \to \infty$.

- We have a new condition at steady-state :

$$\begin{aligned}x_s &= Ax_s + Bu_s + B_d\hat{d}_\infty\\ z_s &= H\left(Cx_s + C_d\hat{d}_\infty\right) = r\end{aligned}$$

  - The system at steady-state is modified to account for the ffect of disturbance on state evolution.
  - The target is modified to account for the effect of disturbance on tracked variables

- The best forecast for steady-state disturbance is current estimate $\hat{d}_\infty = \hat{d}$
- We adapt the target condition accordingly to account for disturbance :

$$\begin{pmatrix}A-I & B\\ HC & 0\end{pmatrix}\begin{pmatrix}x_s\\ u_s\end{pmatrix} = \begin{pmatrix}-B_d\hat{d}\\ r - HC_d\hat{d}\end{pmatrix}$$

Remark : We have the same procedure for the regulation case with $r = 0$.

At each sampling time :

1. We estimate the state and disturbance $\hat{x}$, $\hat{d}$
2. We obtain (x_s, u_s) from steady-state target problem using the disturbance estimate.

3. We solve the MPC problem for tracking using the disturbance estimate $\hat{d}$:

$$\begin{aligned}
&\min \sum_{i=0}^{N-1} (x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s) + V_f (x_N - x_s) \\
\text{subject to} \quad & x_0 = \hat{x}(k) \\
& d_0 = \hat{d}(k) \\
& x_{i+1} = Ax_i + Bu_i + B_d d_i \qquad i = 0, \ldots, N \\
& d_{i+1} = d_i \qquad i = 0, \ldots, N \\
& x_i \in \mathcal{X} \quad u_i \in \mathcal{U} \qquad i = 0, \ldots, N-1 \\
& x_N - x_s \in \mathcal{X}_f
\end{aligned}$$

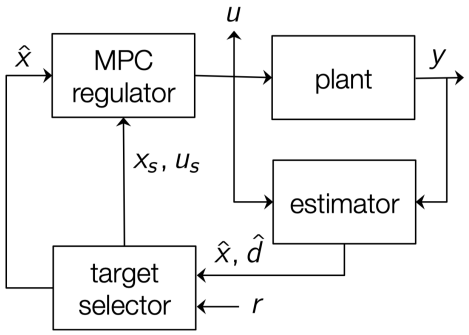


Figure 8.14

We can write this problem in delta-formulation.

Offset-Free Tracking - Main Result

We denote by $\kappa\left(\hat{x}(k), \hat{d}(k), r\right) = u_0^*$ the control law when the estimated state and disturbance are $\hat{x}(k)$ and $\hat{d}(k)$, respectively.

Theorem 8.2.2. ***Zero Offset Tracking***

Consider the case where the number of constant disturbances equals the number of outputs $n_d = n_y$. Assume the RHC is recursively feasible and unconstrained for $k \geq j$ with $j \in \mathbb{N}^+$ and the closed-loop system

$$\begin{aligned}
x(k+1) &= Ax(k) + B\kappa\left(\hat{x}(k), \hat{d}(k), r\right) + B_d d \\
\hat{x}(k+1) &= (A + L_x C)\,\hat{x}(k) + (B_d + L_x C_d)\,\hat{d}(k) + B\kappa\left(\hat{x}(k), \hat{d}(k), r\right) - L_x y(k) \\
\hat{d}(k+1) &= L_d C \hat{x}(k) + (I + L_d C_d)\,\hat{d}(k) - L_d y(k)
\end{aligned}$$

converges, i.e. $\hat{x}(k) \to \hat{x}_\infty$, $\hat{d}(k) \to \hat{d}_\infty$, $y(k) \to y_\infty$ as $k \to \infty$.

Then $z(k) = Hy(k) \to r$ as $k \to \infty$ (we have zero offset tracking).

8.2.5 Example - Offset-Free Control

Let us consider a double integrator :
First set point : $r = 0$, second set point : $r = 3$. We have 1 input, 1output and 1 modeled

disturbance.

Standard tracking :
Model :

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

$$-0.5 \leq u(k) \leq 0.5$$
$$-5 \leq y(k) \leq 5$$

Target condition :

$$\begin{pmatrix} I - A & -B \\ C & 0 \end{pmatrix} \begin{pmatrix} x_s \\ u_s \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$$

Offset-free tracking :
Model :

$$x(k+1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u(k) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} d(k)$$
$$z(k) = y(k) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(k)$$

$$-0.5 \leq u(k) \leq 0.5$$
$$-5 \leq y(k) \leq 5$$

$\begin{pmatrix} I - A & -B_d \\ C & 0 \end{pmatrix}$ has full column rank

Target condition :

$$\begin{pmatrix} I - A & -B \\ C & 0 \end{pmatrix} \begin{pmatrix} x_s \\ u_s \end{pmatrix} = \begin{pmatrix} -B_d \hat{d} \\ r - C_d \hat{d} \end{pmatrix}$$

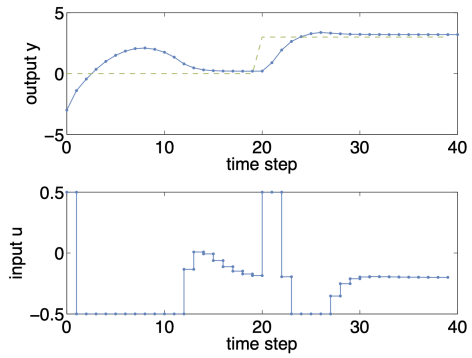


Figure 8.15

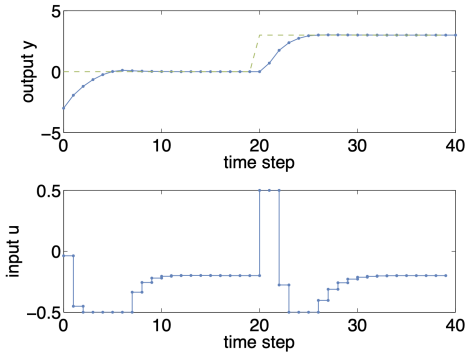
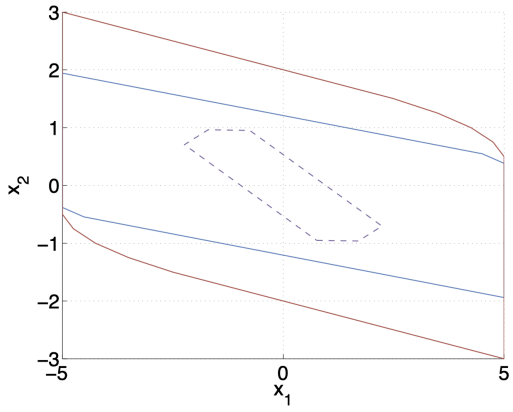


Figure 8.16

8.3 Enlarging the Feasible Set

8.3.1 MPC without Terminal Set

We know that the terminal sets reduces the feasible set :



- Blue Line : Feasible set with terminal constraint
- Red Line : Feasible set without terminal constraint
- Dashed Line : Terminal Constraint

Figure 8.17

- Potentially adds large number of extra constraints
- Adds state constraints to problems with only input constraints

Goal : Design an MPC without terminal constraints with guaranteed stability.

Remark : A Feasible set without a terminal constraint is not invariant.

We can remove the terminal constraint while maintaining stability if :

- The initial state lies in a sufficiently small subset of feasible sets
- N is sufficiently large

such that the terminal state satisfies the terminal constraint without enforcing it in the optimization.

↪ The solution of the finite horizon MPC problem corresponds to the infinite horizon solution.

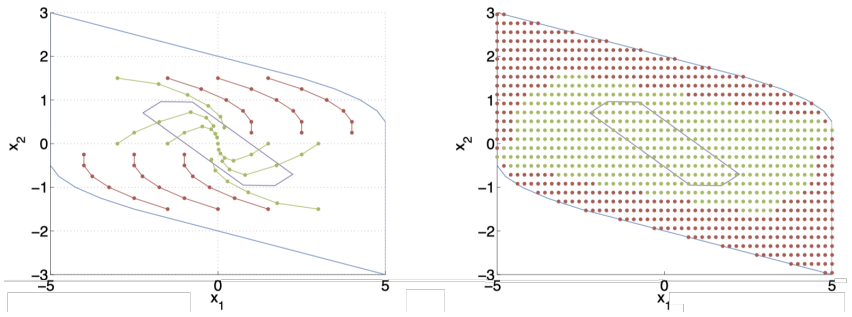


Figure 8.18

Discussion

Advantage : The controller is defined in a larger feasible set.
Disadvantage : The characterization of a region of attraction or specification of the required horizon length is extremely difficult.

Remarks :

- The terminal constraint provides a sufficient condition for stability : The region of attraction without a terminal constraint may be larger than for MPC with a terminal constraint
- In practice : Enlarge the horizon and check stability by sampling
- With a larger horizon length N, the region of attraction approaches a maximum control invariant set.

8.3.2 Soft Constrained MPC

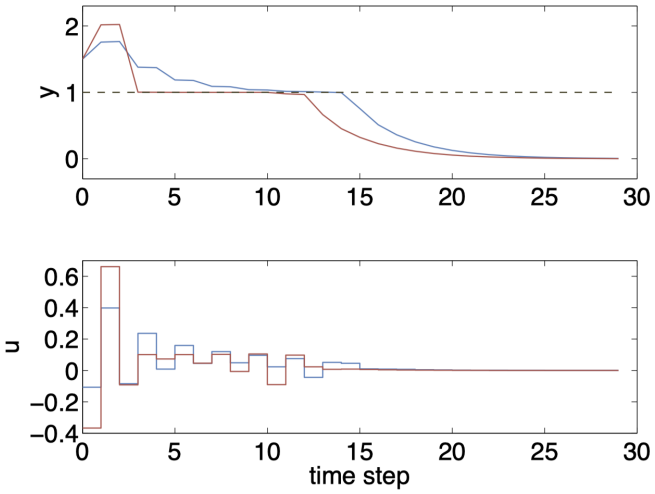
Motivation

- The input constraints are dictated by physical constraints on the actuators and are usually "hard" constraints.
- The state/output constraints arise from practical restrictions on the allowed operating range and are **rarely hard constraints**.
- Hard state / output constraints always lead to **complications in the controller implementation**
  - Feasible operating regime is constrained even for stable systems
  - The controller patches must be implemented to generate reasonable control action when measured / estimated states move outside of the feasible range because of disturbances.
- In industrial implementations, typically, state constraints are **softened**.

Goals :

- Minimize the duration of the violation
- Minimize the size of the violation

These can be conflicting goals, i.e. reduction in size of the violation can only be achieved at the cost of a large increase in the duration of the violation. We therefore have a multi-objective problem.



Red : Minimum time of violation
Blue : Minimum size of violation

Figure 8.19

For a given system and horizon we plot the optimal size / duration curve for different initial

conditions :

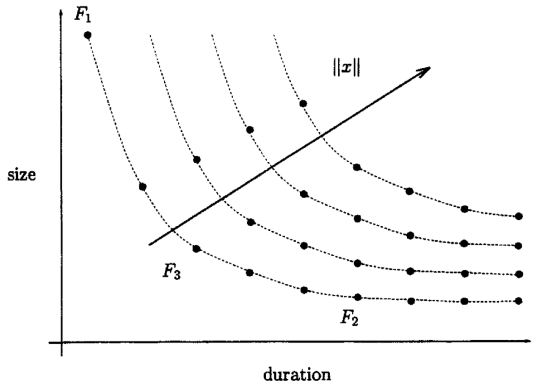


Figure 8.20

The best operation points lie on the optimal curve :

- Points below cannot be attained
- Points above are inferior

→ Operation at optimality is in general difficult and only approximately achieved

What is the best operation point minimum time or minimum violation?
That depends on the application, e.g. :

- If a product must be discarded during constraint violation, the goal is to minimize the time of violation.
- If large constraint violations can lead to process shutdowns or exceptions, the goal is to minimize the size of the violation.

Soft Constrained MPC Problem Setup

$$\min \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T P x_N + \underbrace{l_\varepsilon(\varepsilon_N)}_{\text{penalty}}$$

subject to
$$x_{i+1} = Ax_i + Bu_i$$
$$H_x x_i \leq k_x + \varepsilon_i \qquad \leftarrow \varepsilon_i \text{ is our slack variable}$$
$$H_u u_i \leq k_u$$
$$\varepsilon_i \geq 0$$

- We relax our state constraints be introducing **slack variables** $\varepsilon_i \in \mathbb{R}^p$
- We penalize the amount of constraint violation in the cost by means of the penalty $l_\varepsilon(\varepsilon_i)$

How can we choose the penalty?

- Quadratic penalty : $l_\varepsilon(\varepsilon_i) = \varepsilon_i^T S \varepsilon_i$
- Quadratic and linear norm penalty : $l_\varepsilon(\varepsilon_i) = \varepsilon_i^T S \varepsilon_i + v\|\varepsilon_i\|_{1/\infty}$ ← Better choice

Mathematical Formulation

- **Original** problem :

$$\min_z f(z)$$
$$\text{subject to} g(z) \leq 0$$

We assume for now that $g(z)$ is scalar valued.

- "**Softened**" problem :

$$\begin{aligned} & \min_{z,\varepsilon} f(z) + l_\varepsilon(\varepsilon) \\ \text{subject to} \quad & g(z) \leq \varepsilon \\ & \varepsilon \geq 0 \end{aligned}$$

If the original problem has a feasible solution z^*, then the softened problem should have the same solution z^*, and $\varepsilon = 0$.

Remark : $l_\varepsilon(\varepsilon) = s \cdot \varepsilon^2$ does not meet this requirement for and $s > 0$ as demonstrated next.

Quadratic Penalty

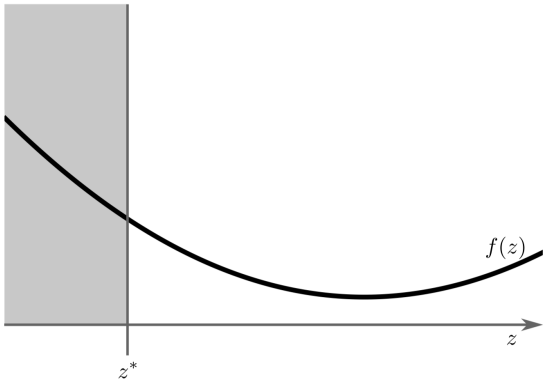


Figure 8.21

The constraint function $g(z) \triangleq z - z^* \leq 0$ induces a feasible region (grey).
$\Rightarrow$ The minimizer of the original problem is z^*.

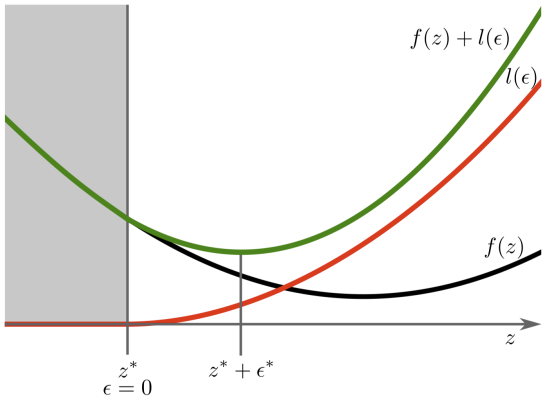


Figure 8.22: We will never obtain a minimum with $\varepsilon = 0$. We need to use a linear penalty.

Quadratic penalty $l_\varepsilon(\varepsilon) = s \cdot \varepsilon^2$ for $\varepsilon \geq 0$, $l_\varepsilon(\varepsilon) = 0$ otherwise
$\Rightarrow$ The minimizer of $f(z) + l_\varepsilon(\varepsilon)$ is $(z^* + \varepsilon^*, \varepsilon^*)$ instead of $(z^*, 0)$

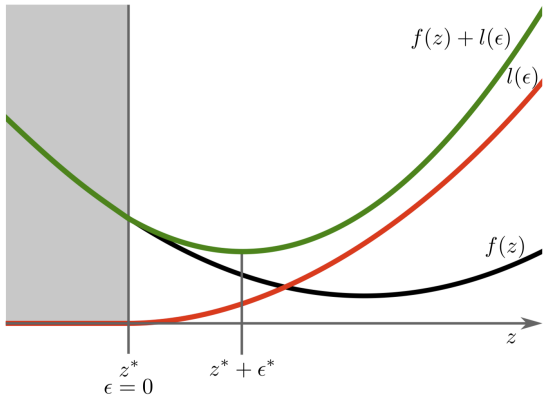


Figure 8.23: We will never obtain a minimum with $\varepsilon = 0$. We need to use a linear penalty.

Linear penalty $l_\varepsilon(\varepsilon) = v \cdot \varepsilon$ for $\varepsilon \geq 0$, $l_\varepsilon(\varepsilon) = 0$ otherwise, with$v$ chosen large enough so that $v + \lim_{z \to z^*} f^(z) > 0$
$\Rightarrow$ The minimizer of $f(z) + l_\varepsilon(\varepsilon)$ is $(z^*, 0)$

Theorem 8.3.1. ***Exact Penalty Function***
$l_\varepsilon(\varepsilon) = v \cdot \varepsilon$ satisfies the requirement for any $v > \lambda^* \geq 0$, where λ^* is the optimal Lagrange multiplier for the original problem.

- In practice, the combined cost is used for the exact penalty and tuning capabilities

$$l_\varepsilon(\varepsilon) = v \cdot \varepsilon + s \cdot \varepsilon^2$$

with $v > \lambda^*$ and $s > 0$.

- Extension to multiple constraints $g_j(z) \leq 0$, $j = 1, \ldots, r$:

$$l_\varepsilon(\varepsilon) = v \cdot \|\varepsilon\|_{1/\infty} + \varepsilon^T S \varepsilon \tag{8.1}$$

where $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_r]$ and $v > \|\lambda^*\|_D$ and $S > 0$ can be used to weigh the violations differently. $\|\cdot\|_D$ denotes the dual of the chosen norm (the dual of $\|\cdot\|_1$ is $\|\cdot\|_\infty$ and vice versa).

Example

Consider the third-order non-minimum phase system :

$$x(k+1) = \begin{pmatrix} 2 & -1.45 & 0.35 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u(k)$$
$$y(k) = \begin{pmatrix} -1 & 0 & 2 \end{pmatrix} x(k)$$

Initial condition $x(0) = \begin{pmatrix} 0.8 & 0.8 & 0.8 \end{pmatrix}^T$:

- Constraint : $-1 \leq y \leq 1$
- Controller parameters : $Q = C^T C$, $R = 1$ and $N = 20$

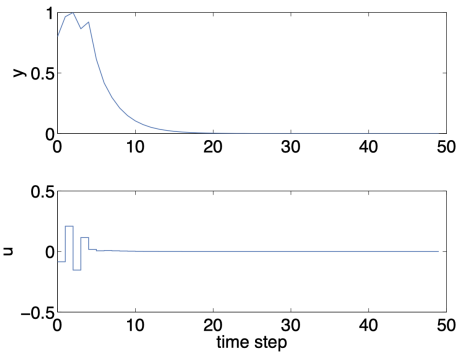


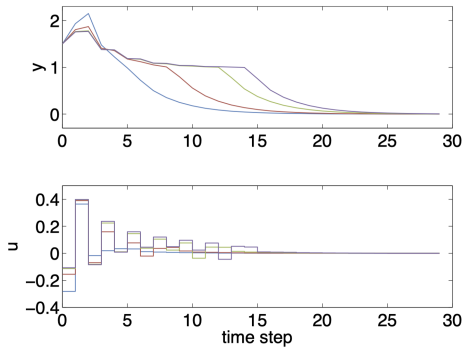
Figure 8.24

Soft Constraints with Quadratic Penalty

Properties of the linear penalty :

- Well-posed quadratic problem (positive definite Hessian)
- Increase in S leads to "hardening" of the soft constraints

Example



Initial condition : $x(0) = \begin{pmatrix} 1.5 & 1.5 & 1.5 \end{pmatrix}^T$

Blue : $S = 1$
Red : $S = 10$
Green : $S = 50$
Violet : $S = 100$

Figure 8.25

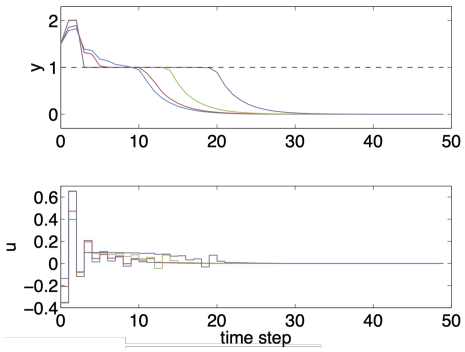
↪ The increase in S leads to the reduced size of violation but a longer duration.

Soft Constraints with Quadratic & Linear Penalty

Properties of the linear penalty :

- This allows for exact penalties. If the chosen weight v is large enough, constraints are satisfied if possible.
- Increasing v results in increasing the peak violations and decreasing the duration
- Large linear penalties make tuning difficult and cause numerical problems

Example



Initial condition : $x(0) = \begin{pmatrix}1.5 & 1.5 & 1.5\end{pmatrix}^T$
Penalty : 1-norm, $S = 20$

Blue : $S = 0.1$
Red : $S = 10$
Green : $S = 25$

Figure 8.26

Simplification - Separation of Objectives

1. Minimize violation over horizon :

$$\varepsilon = \arg\min_{U,\varepsilon} \left[\varepsilon_i^T S \varepsilon_i + v^T \varepsilon_i\right]$$

subject to
$$x_{i+1} = Ax_i + Bu_i$$
$$H_x x_i \leq K_x + \varepsilon_i$$
$$H_u u_i \leq K_u$$
$$\varepsilon_i \geq 0$$

2. Optimize controller performance :

$$\min_u \sum_{i=0}^{N-1} [x_i^T Q x_i + u_i^T R u_i] + x_N P x_N$$

subject to
$$x_{i+1} = Ax_i + Bu_i$$
$$H_x x_i \leq K_x + \varepsilon_i^{\min}$$
$$H_u u_i \leq K_u$$
$$\varepsilon_i \geq 0$$

↪ Advantage : Simplifies tuning, constraints will be satisfied if possible.
↪ Disadvantage : Requires solution of two optimization problems.

Summary - Soft Contraints

- Soft constraints recover feasibility of the optimization when constraints cannot be satisfied
- They allow for a variety of violation duration vs. size trade-offs
- They provide good closed-loop properties
- They are generally applied in practice

Remark : Standard methods for soft constrained MPC do not provide a stability guarantee for open-loop unstable systems.

↪ There have been recent developments towards soft constrained MPC with stability guarantees (M. Zeilinger).

Summary

In general the state cannot be measured, so we use a Kalman filter to estimate the state.

The design tracking problem :

- Rewrite the problem in delta-formulation
- Setup the target steady-state problem

- Calculate the terminal weight and scale the terminal constraint to guarantee convergence

Extension to offset-free tracking :

- Augment the model including a disturbance tool
- Augment the estimator to estimate the state and the disturbance
- Adapt the target steady-state problem using the disturbance estimate

We could possible remove the terminal constraint while choosing a long horizon.

Introduce soft constraints to ensure feasibility at all times :

- Introduce slack variables for constant relaxation
- Choose a penalty on the slack variables (quadratic, linear)

Chapter 9

Robust MPC

Motivation

MPC relies on a model, however these models that we make are far from reality. These inaccurate models, combined with noise can cause :

- Constraint violation
- Sub-optimal behaviour

Persistent noise prevents the system from converging to a single point. We can incorporate some noise models into the MPC formulation, however solving the resulting optimal control problem can be extremely difficult.

In this chapter we will understand the impact of additive noise on the trajectory, reformulate our constraints to ensure a robust constraint satisfaction and formulate a robust open-loop MPC problem.

9.1 Uncertainty Models

MPC Model Vs. The Real World

Predictive control assumption :

$$x(k+1) = g\left(x(k), u(k)\right)$$

In this case the system evolves in a predictable fashion.

The real world :

$$x(k+1) = g\left(x(k), u(k), w(k), \theta\right)$$

- Random noise w changes the evolution of the system
- The true model structure is unknown
- Unknown parameters θ impact the dynamics

(w changes with time, θ is unknown but constant)

What can hope to do in this real situation?

Goals of Robust Constrained Control

If we have the uncertain constrained system :

$$x(k+1) = g\left(x(k), u(k), \theta\right) \qquad x, u \in \mathcal{X}, \mathcal{U} \quad w \in \mathcal{W} \quad \theta \in \Theta$$

We want to design the control law $u(k) = \kappa(x(k))$ such that the system :

1. Satisfies the constraints : $\{x(k)\} \subset \mathcal{X}$, $\{u(k)\} \subset \mathcal{U}$ for all disturbance realizations
2. Is stable : Converges to a neighborhood of the origin
3. Optimizes (expected / worst case) "performance"
4. Maximizes the set $\{x(0)|\text{Conditions 1-3 are met}\}$

Meeting these goals requires some knowledge/assumptions about the random values w and θ.

Examples of Common Uncertainty Models

- **Measurement / Input Bias** (previous chapter) :

$$g(x(k), u(k), w(k), \theta) = \tilde{g}(x(k), u(k)) + \theta$$

  θ unknown but constant.

- **Linear Parameter Varying System** :

$$g(x(k), u(k), w(k), \theta) = \sum_{j=0}^{t} \theta_j A_j x(k) + \sum_{k=0}^{t} \theta_j B_j u(k) \qquad \mathbf{1}^T\theta = 1, \quad \theta \geq 0$$

  A_k and B_k are known, θ_k is unknown, but with a fixed value at each sampling time.

- **Additive Stochastic Noise** :

$$g(x(k), u(k), w(k), \theta) = Ax(k) + Bu(k) + w(k)$$

  Distribution of w is known

For additive bounded noise, the values of A and B are known, w is unknown and changing at each sampling instance.

- The dynamics are linear, but impacted by random, bounded noise at each time step.
- We can model many non-linearities in this fashion, but often a conservative model.
- The noise is *persistent*, i.e., it does not converge to zero in the limit.

We will focus on uncertainty models of this form in this course.

9.2 Impact of Bounded Additive Noise

9.2.1 Goals of Robust Constrained Control

If we take the uncertain constrained linear system :

$$x(k+1) = Ax(k) + Bu(k) + w(k) \qquad\qquad x, u \in \mathcal{X},\mathcal{U} \quad w \in \mathcal{W}$$

We said that we want to design the control law $u(k) = \kappa(x(k))$ such that the system :

1. Satisfies the constraints : $\{x(k)\} \subset \mathcal{X}$, $\{u(k)\} \subset \mathcal{U}$ for all disturbance realizations
2. Is stable : Converges to a neighborhood of the origin

3. Optimizes (expected / worst case) "performance"
4. Maximizes the set $\{x(0)|\text{Conditions 1-3 are met}\}$

Challenge : We cannot predict where the state of the system will evolve. We can only compute a set of trajectories that the system *may* follow.

Idea : We design a control law that will satisfy constraints and stabilize the system *for all possible disturbances.*

9.2.2 Uncertain State Evolution

Given the current state $x(0)$, the model $x(k+1) = Ax(k) + Bu(k) + w(k)$ and the set $\mathcal{W}$, where can the state be i steps in the future?

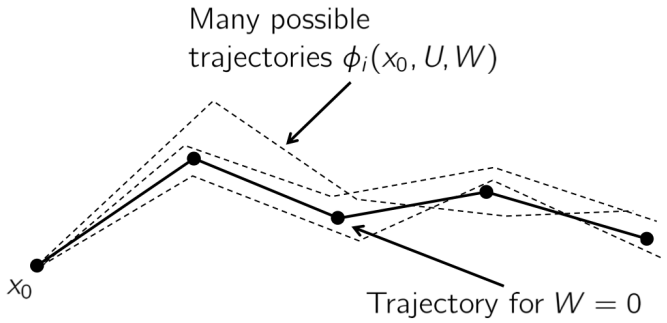


Figure 9.1

We define $\phi_i(x_0, U, W)$ as the state that the system will be in at time i if the state at time zero is x_0. We apply the input $U \coloneqq \{u_0, \ldots, u_{N-1}\}$ and we observe the disturbance $W \coloneqq \{w_0, \ldots, w_{N-1}\}$.

Nominal System :

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

$$\begin{aligned} x_1 &= Ax_0 + Bu_0 \\ x_2 &= A^2x_0 + ABu_0 + Bu_1 \\ &\vdots \\ x_i &= A^ix_0 + \sum_{j=0}^{i-1} A^jBu_{i-1-j} \end{aligned}$$

Uncertain System :

$$x(k+1) = Ax(k) + Bu(k) + w(k) \quad w \in \mathcal{W}$$

$$\begin{aligned} \phi_1 &= Ax_0 + Bu_0 + w_0 \\ \phi_2 &= A^2x_0 + ABu_0 + Bu_1 + Aw_0 + w_1 \\ &\vdots \\ \phi_i &= \underbrace{A^ix_0 + \sum_{j=0}^{i-1} A^jBu_{i-1-j} + \sum_{j=0}^{i-1} A^jw_{i-1-j}}_{\text{written as a function of initial conditions}} \end{aligned}$$

$$\phi_i = \underbrace{x_i + \sum_{j=0}^{i-1} A^jw_{i-1-j}}_{\text{written as a function of } x_i}$$

Uncertain evolution is the nominal system + offset caused by the disturbance (follows from linearity).

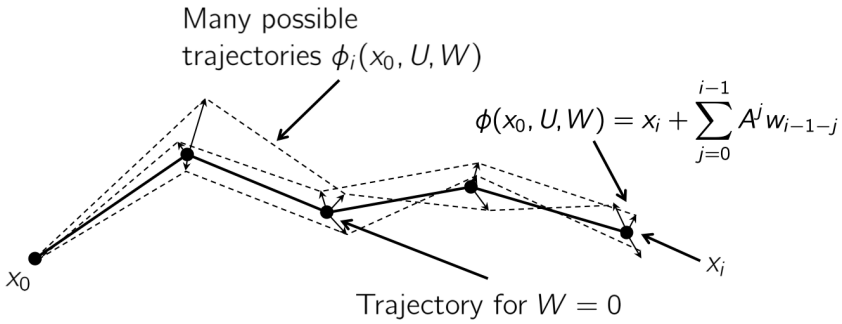


Figure 9.2

Defining a Cost to Minimize

Previously, we defined some functions that describes a "good" trajectory :

$$J(x_0, U) \coloneqq \sum_{i=0}^{N-1} l(x_i, u_i) + l_f(x_N)$$

However, there are now many trajectories that *may* occur, depending on the disturbance W.

The cost is now a function of the disturbance seen, and therefore *each possible trajectory has a different cost* :

$$J(x_0, U, W) \coloneqq \sum_{i=0}^{N-1} l(\phi_i(x_0, U, W)) + l_f(\phi_N(x_0, U, W))$$

We need to "eliminate" the dependance on W. There are several common options :

- Minimize the expected value (requires some assumptions on the distribution)

$$J_N(x_0, U) \coloneqq E[J(x_0, U, W)]$$

- Take the worst case

$$J_N(x_0, U) \coloneqq \max_{W \in \mathcal{W}^{n-1}} J(x_0, U, W)$$

- Take the nominal case

$$J_N(x_0, U) \coloneqq J(x_0, U, 0)$$

In this chapter we will assume the nominal case for simplicity.

Robust Constraint Satisfaction

Recall that we can break the MPC prediction into two parts.

$$\begin{aligned} \phi_{i+1} &= A\phi_i + Bu_i + w_i \\ u_i &\in \mathcal{U} \\ \phi_i &\in \mathcal{X} \forall W \in \mathcal{W}^N \end{aligned}$$

- $i = 0, \ldots, N-1$
- Optimize over control actions $\{u_0, \ldots, u_{N-1}\}$
- Enforce constraints explicitly by imposing $\phi_i \in \mathcal{X}$ and $u_i \in \mathcal{U}$ for all sequences W

$$\phi_N \in \mathcal{X}_f$$
$$\phi_{i+1} = (A + BK)\,\phi_i + w_i$$

- $i = N, \ldots$
- Assume control law to be linear $u_i = K\phi_i$
- Enforce constraints implicitly by constraining ϕ_N to be in a *robust invariant set* $\mathcal{X}_f \subseteq \mathcal{X}$ and $K\mathcal{X}_f \subseteq \mathcal{U}$ for the system $\phi_{i+1} = (A + BK)\,\phi_i + w_i$

In the following :

- Robustly enforcing the constraints of a linear system
- Robustly ensuring of the sequence $\phi_1, \ldots, \phi_{N-1}$

Robust Invariance

We want robust constraint satisfaction, for an **autonomous** system $x(k+1) = g\,(x(k), w(k))$ or **closed-loop** system $x(k+1) = g\,(x(k), \kappa\,(x(k))\,, w(k))$ for a **given** controller κ.

Definition 9.2.1. Robust Positive Invariant Set
A set $\mathcal{O}^{\mathcal{W}}$ is said to be a **robust positive invariant set** for the autonomous system $x(k+1) = g\,(x(k), w(k))$ if

$$x \in \mathcal{O}^{\mathcal{W}} \quad \rightarrow \quad g\,(x, w) \in \mathcal{O}^{\mathcal{W}} \quad \text{for all } w \in \mathcal{W}$$

Definition 9.2.2. Robust Pre-Set
Given a set Ω and the dynamic system $x(k+1) = g\,(x(k), w(k))$, the **robust pre-set** of Ω is the set of states that evolve into the target set Ω in one time step *for all values of the disturbance* $w \in \mathcal{W}$:

$$\text{pre}^{\mathcal{W}}\,(\Omega) \coloneqq \{x | g\,(x, w) \in \Omega \text{ for all } w \in \mathcal{W}\}$$

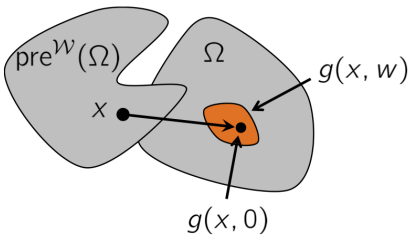


Figure 9.3

Computing Robust Pre-Sets for Linear Systems

Goal : Given the system $g\,(x(k), w(k)) = Ax(k) + w(k)$, and the set $\Omega \coloneqq \{x | Fx \leq f\}$, compute the $\text{pre}^{\mathcal{W}}\,(\Omega)$.

$$\text{pre}^{\mathcal{W}}\,(\Omega) = \{x | Ax + w \in \Omega,\, \forall w \in \mathcal{W}\} = \{x | FAx + Fw \leq ,\, \forall w \in \mathcal{W}\}$$

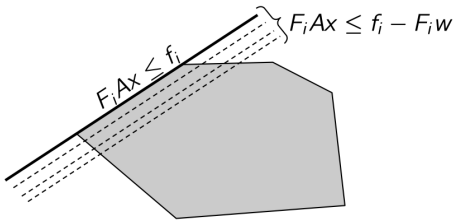


Figure 9.4

Wee can see in Figure 9.4 that adding in F_iw make s the constraint tighter / looser. The worst case of tightening our constraints would be if we had $F_iAx \leq f_i - \max_{w \in \mathcal{W}} F_iw$.

In Figure 9.5 we can see that the possible trajectory with uncertainty does not exit the set!

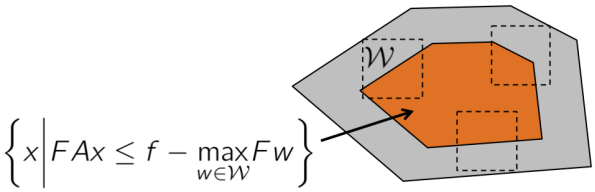


Figure 9.5

$$\text{pre}^{\mathcal{W}}(\Omega) = \left\{x \middle| FAx \leq f - \max_{w \in \mathcal{W}} Fw\right\} = \{x | FAx \leq f - h_{\mathcal{W}}(F)\}$$

where $h_{\mathcal{W}}$ is the *support function.*

Theorem 9.2.1. ***Geometric Condition for Robust Invariance***
A set $\mathcal{O}^{\mathcal{W}}$ is a robust positive invariant set if and only if :

$$\mathcal{O}^{\mathcal{W}} \subseteq pre^{\mathcal{W}}(\mathcal{O}^{\mathcal{W}})$$

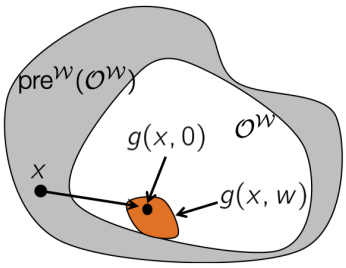


Figure 9.6

Computing Robust Invariant Sets

In order to compute the robust invariant set, we can use the following algorithm :

Algorithm 2: Conceptual Algorithm to Compute Invariant Set

Input : g, $\mathcal{X}$, $\mathcal{W}$
Output: $\mathcal{O}_\infty^{\mathcal{W}}$

$\Omega_0 \leftarrow \mathcal{X}$
while $\Omega_i \neq \Omega_{i-1}$ **do**
  $\Omega_{i+1} \leftarrow \text{pre}^{\mathcal{W}}(\Omega_i) \cap \Omega_i$
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{O}_\infty^{\mathcal{W}} = \Omega_i$
  end
end

We can see that this is the same algorithm as Algorithm 1, however we have replaced $\text{pre}(\Omega)$ with $\text{pre}^{\mathcal{W}}(\Omega)$.

Example

If we take the following system :

$$x(k+1) = (A + BK)\, x(k) + w(k)$$

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$$

$$\mathcal{X} = \{x \,|\, \|x\|_\infty \leq 5,\ \|Kx\|_\infty \leq 1\} \qquad \mathcal{W} = \{w \,|\, \|w\|_\infty \leq 0.3\}$$

where K is the LQR controller for $Q = 0.1 \cdot I$ and $R = 1$, we can represent the sets shown in Figure 9.7

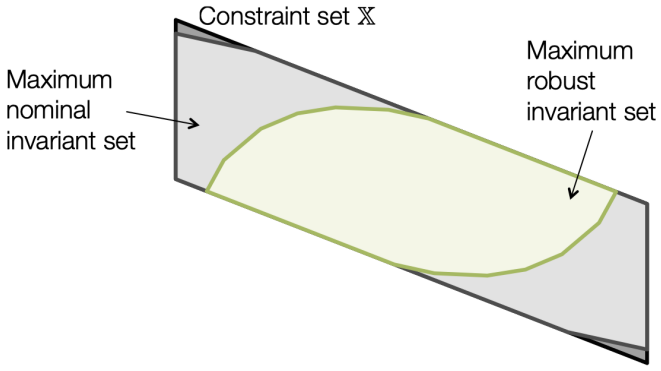


Figure 9.7

We can see that if we start outside of the maximum robust invariant set, chances are that we will violate the constraints imposed by our problem :

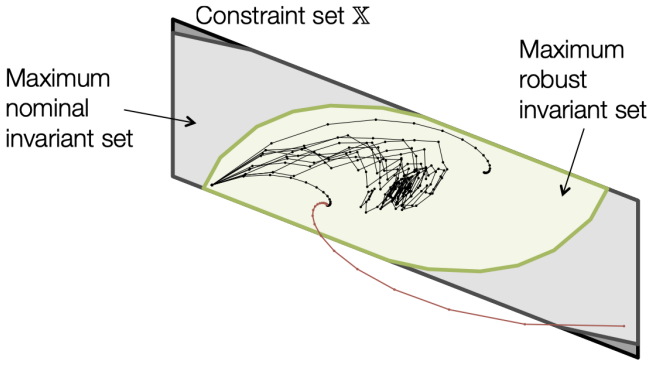


Figure 9.8: The red line violates the constraints

9.2.3 Robust Constraint Satisfaction

As well as robustly enforcing the constraints of a linear system, we also want to robustly ensure the constraints of the sequence $\phi_1, \ldots, \phi_{N-1}$.

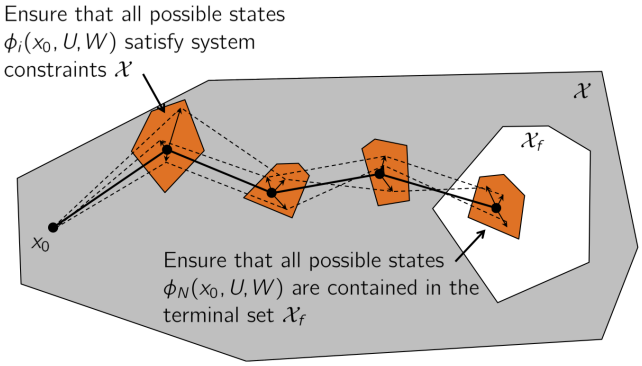


Figure 9.9: Note that in reality the uncertainty sets (in orange) will grow larger the further we predict into the future.

The idea : We compute a set of tighter constraints such that if the **nominal system** meets these constraints, then the uncertain system will too. We then solve the MPC problem **on the nominal system**.

Goal : Ensure that the constraints are satisfied for the MPC sequence.

$$\phi_i\left(x_0, U, W\right) = \left\{ x_i + \sum_{j=0}^{i-1} A^j w_{i-1-j} \,\middle|\, W \in \mathcal{W}^i \right\} \subseteq \mathcal{X}$$

We assume that $\mathcal{X} = \{x | Fx \leq f\}$, then this is equivalent to

$$Fx_i + F\sum_{j=0}^{i-1} A^j w_{i-1-j} \leq f \quad \forall W \in \mathcal{W}^i$$

We've seen this before while computing the robust pre-set :

$$Fx_i \leq f - \max_{W \in \mathcal{W}^i} F\sum_{j=0}^{i-1} A^j w_{i-1-j} = f - h_{\mathcal{W}^i}\left(F\sum_{j=0}^{i-1} A^j\right)$$

This support function can be pre-computed offline. **All we're doing is tightening the constraints on the nominal system**.

Terminal State Constraint

We also need to ensure that the N^{th} state $\phi_N(x_0, U, W)$ is contained in the robust invariant set $\mathcal{X}_f$:

$$\phi_N(x_0, U, W) \subseteq \mathcal{X}_f$$

This is handled in exactly the same fashion.

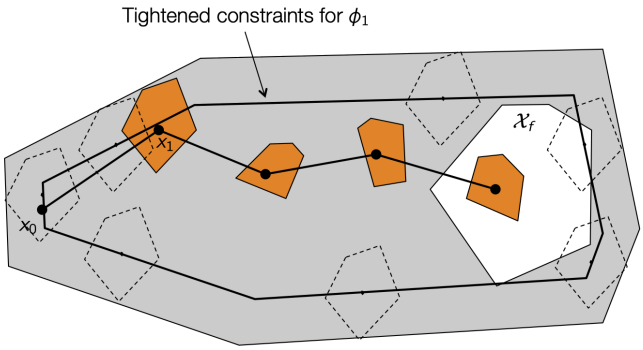


Figure 9.10

The nominal x_i satisfies the **tighter constraints** $\rightarrow$ **The uncertain state does too**.

Definition 9.2.3. Minkowski Sum

Let A and B be subsets of $\mathbb{R}^n$. The **Minkowski sum** is

$$A \oplus B \coloneqq \{x + y | x \in A, y \in B\}$$

Definition 9.2.4. Pontryagin Difference

Let A and B be subsets of $\mathbb{R}^n$. The **Pontryagin difference** is

$$A \ominus B \coloneqq \{x | x + e \in A \quad \forall e \in B\}$$

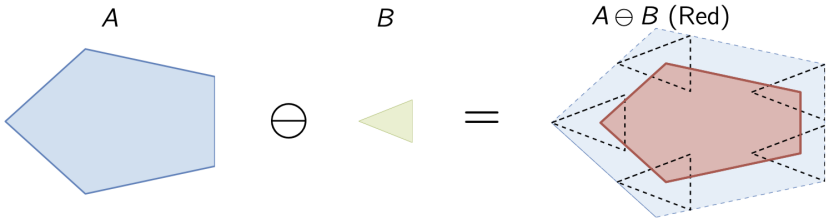


Figure 9.11

Remark : $A \ominus B \oplus B \neq A$

Notation

$$\phi_i(x_0, U, W) = \left\{ x_i + \sum_{j=0}^{i-1} A^j w_{i-1-j} \,\middle|\, W \in \mathcal{W}^i \right\} \subseteq \mathcal{X}$$

can also be written as

$$\phi_i(x_0, U, W) \in x_i \oplus \left(\mathcal{W} \oplus A\mathcal{W} \oplus \cdots \oplus A^{i-1}\mathcal{W}\right) \subseteq \mathcal{X}$$

In order to enforce this condition we require :

$$x_i \in \mathcal{X} \ominus \left(\mathcal{W} \oplus A\mathcal{W} \oplus \cdots \oplus A^{i-1}\mathcal{W}\right)$$

also denoted as $x_i \in \mathcal{X} \ominus \left(\bigoplus_{j=0}^{i-1} A^j \mathcal{W}\right)$.

9.3 Robust Open-Loop MPC

$$\begin{aligned} & \min_U \sum_{i=0}^{N-1} l\left(x_i, u_i\right) + l_f\left(x_N\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \\ & x_i \in \mathcal{X} \ominus \left(\mathcal{W} \oplus A\mathcal{W} \oplus \cdots \oplus A^{i-1}\mathcal{W}\right) \\ & u_i \in \mathcal{U} \\ & x_i \in \mathcal{X}_f \ominus \left(\mathcal{W} \oplus A\mathcal{W} \oplus \cdots \oplus A^{N-1}\mathcal{W}\right) \end{aligned}$$

where $\mathcal{X}_f$ is a robust invariant set for the system $x(k+1) = (A + BK)\, x(k)$ for some stabilizing K.

We do **nominal MPC**, but with tighter constraints on the states and inputs. We can be sure that if the nominal system satisfies the tighter constraints, then the uncertain system will satisfy the real constraints.

9.3.1 Properties of Robust Open-Loop MPC

If $U^*\left(x(k)\right)$ is the optimizer of the robust open-loop MPC problem for $x(k) \in \mathcal{X}_0$, then the system $Ax(k) + Bu_0^*\left(x(k)\right) + w(k) \in \mathcal{X}_0$ for all $w \in \mathcal{W}$.

This follows because the trajectory we computed at the current time is feasible for *any* disturbance, and therefore is feasible for the one that we actually observe.

Robust open-loop MPC potentially has a *very* small region of attraction, in particular for unstable systems.

9.4 Closed-Loop Predictions

9.4.1 MPC as a Game

We can imagine our MPC as a game between two players, the controller versus the disturbance.

$$x(k+1) = g\left(x(k), u(k)\right) + w(k)$$

1. The controller chooses his move u
2. The disturbance decides on his move w **after seeing the controller's move**

What are we assuming when making open-loop robust predictions?

1. The controller chooses a **sequence** of N moves in the future $\{u_0, \ldots, u_{N-1}\}$
2. The disturbance chooses N moves **knowing all N moves of the controller**

We are assuming that the controller will do the same thing in the future no matter what the disturbance does! But can we do better?

9.4.2 Closed-Loop Predictions

What should the future prediction look like?

1. Controller decides his first move u_0
2. Disturbance chooses his first move w_0
3. Controller decides his second move $u_1(x_1)$ **as a function of the first disturbance** w_0 **(recall** $x_1 = Ax_0 + Bu_0 + w_0$**)**
4. Disturbance chooses his second move w_1 as a function of u_1
5. Controller decides his second move $u_2(x_2)$ **as a function of the first two disturbances** w_0**,** w_1
6. ...

We want to optimize over a **sequence of functions** $\{u_0, \mu_1(\cdot), \dots, \mu_{N-1}(\cdot)\}$, where $\mu_i(x_i) : \mathbb{R}^n \to \mathbb{R}^m$ is called the **control policy**, and maps the state at time i to an input a time i.

Remarks :

- This is the same as making μ a function of the disturbances to time i, since the state is a function of the disturbances up to that point.
- The first input u_0 is a function of the current state, which is known. Therefore it is not a function, but a single value.

The problem : We can't optimize over arbitrary functions!

Solution : We assume some structure on the functions μ_i :

Pre-Stabilization $\mu_i(x) = Kx + v_i$
- Fixed K, such that $A + BK$ is stable
- Simple, often conservative

Linear Feedback $\mu_i(x) = K_i x + v_i$
- Optimize over K_i and v_i
- Non convex. Extremely difficult to solve...

Disturbance Feedback $\mu_i(x) = \sum_{j=0}^{i-1} M w_j + v_i$
- Optimize over M and v_i
- Equivalent to linear feedback, but convex!
- Can be very effective, but computationally intense

Tube-MPC $\mu_i(x) = v_i + K(x - \bar{x}_i)$
- Fixed K, such that $A + BK$ is stable
- Optimize over $\bar{x}_i$ and v_i
- Simple, and be effeective

We will focus on tube-MPC in this course.

9.5 Tube-MPC

The idea : We want to separate the available control authority into two parts :

1. A portion that steers the noise-free "nominal" system to the origin

$$z(k+1) = Az(k) + Bv(k)$$

2. A portion that compensates for deviations from this system, i.e. a "tracking" controller, to keep the real trajectory close to the nominal

$$u_i = K(x_i - z_i) + v_i$$

for some linear controller K, which stabilizes the nominal system.

$\hookrightarrow$ We fix the linear feedback controller K offline, and optimize over the nominal trajectory $\{v_0, \dots, v_{N-1}\}$, which results in a convex problem.

Error Dynamics

We define the error $e_i = x_i - z_i$, which give the error dynamics :

$$\begin{aligned} e_{i+1} &= x_{i+1} - z_{i+1} \\ &= \underbrace{Ax_i + Bu_i + w_i}_{x_{i+1}} - \underbrace{Az_i - Bv_i}_{z_{i+1}} \\ &= Ax_i + BK(x_i - z_i) + Bv_i + w_i - Az_i - Bv_i \\ &= (A + BK)(x_i - z_i) + w_i \\ &= (A + BK)e_i + w_i \end{aligned}$$

The bound maximum error, or how far the "real" trajectory is from the nominal

$$e_{i+1} = (A + BK)e_i + w_i \qquad w_i \in \mathcal{W}$$

The dynamics $A + BK$ are stable, and the set $\mathcal{W}$ is bounded, so there is some set $\mathcal{E}$ that e will stay inside for all time. We want the smallest such set (the "minimal invariant set").

9.5.1 Tube-MPC - The Idea

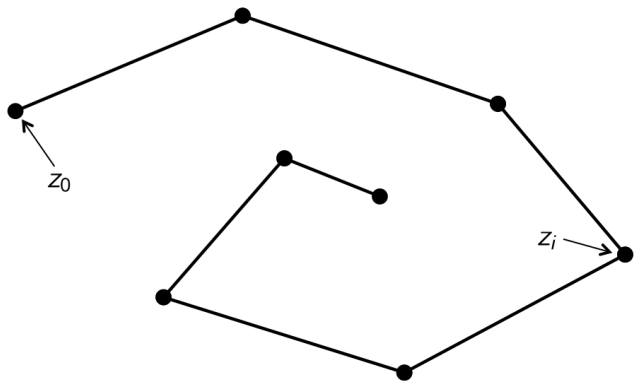


Figure 9.12

We ignore the noise and plan the **nominal trajectory**.

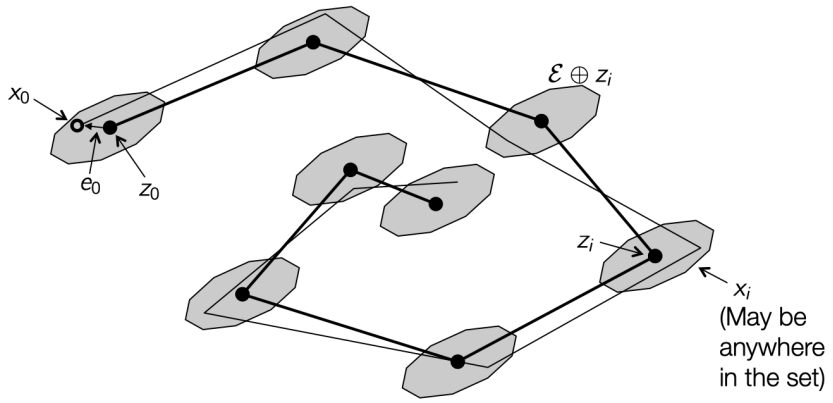


Figure 9.13

We know that the real trajectory stay "nearby" the nominal one : $x_i \in z_i \oplus \mathcal{E}$ **because we plan to apply the controller** $u_i = K(x_i - z_i) + v_i$ **in the future** (we won't actually do this, but it's a valid sub-optimal plan).

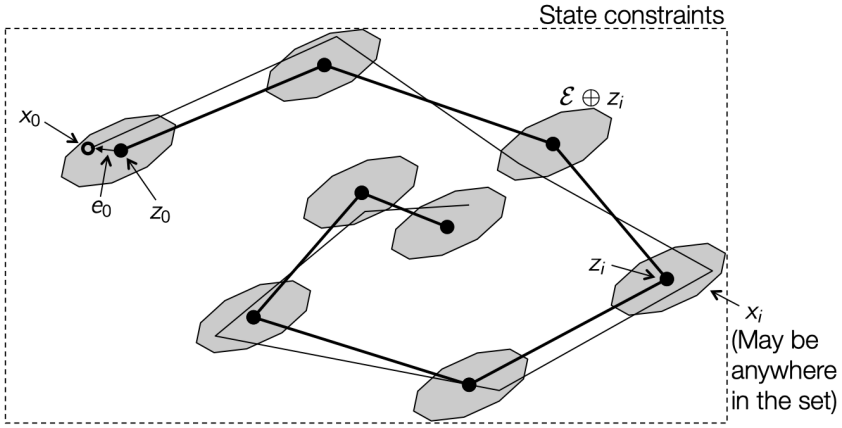


Figure 9.14

We must ensure that all possible state trajectories satisfy the constraints. This is now equivalent to ensuring that $z_i \oplus \mathcal{E} \subset \mathcal{X}$ (we address the input constraints later).

What do we need to make this work?

- Compute the set $\mathcal{E}$ that the error will remain inside
- Modify the constraints on the nominal trajectory $\{z_i\}$ so that $z_i \oplus \mathcal{E} \subset \mathcal{X}$ and $v_i \in \mathcal{U} \ominus K\mathcal{E}$
- Formulate this as a convex optimization problem

. . . and then prove that

- The constraints are robustly satisfied
- The closed-loop system is robustly stable

Recall : Robust Invariant Set

Robust constraint satisfaction, for an autonomous system is $x(k+1) = g(x(k), w(k))$, or **closed-loop** system $x(k+1) = g(x(k), \kappa(x(k)), w(k))$ for a **given** controller κ.

Definition 9.5.1. Robust Positive Invariant Set

A set $\mathcal{O}^{\mathcal{W}}$ is said to be a **robust invariant set** for the autonomous system $x(k+1) = g(x(k), w(k))$ if

$$x \in \mathcal{O}^{\mathcal{W}} \quad \rightarrow \quad g(x, w) \in \mathcal{O}^{\mathcal{W}} \quad \text{for all } w \in \mathcal{W}$$

Previously we wanted the **maximum robust invariant set**, or the largest set in which our terminal control law works.

We now want the **minimum robust invariant set**, or the smallest set that the state will remain inside despite the noise.

Uncertain State Evolution

If we consider the system $x_{i+1} = Ax_i + w_i$ and assume that $x_0 = 0$. Where can the state evolve to? (i.e., how close can we stay to the origin?)

$$\begin{aligned} x_1 &= w_0 \\ x_2 &= Ax_1 + w_1 = Aw_0 + w_1 \\ &\vdots \\ x_i &= \sum_{k=0}^{i-1} A^k w_{i-1-k} \end{aligned}$$

We assume that $w_i \in \mathcal{W}$ for all i. What is the set F_i that contains all possible states x_i?

$$F_i = \mathcal{W} \oplus A\mathcal{W} \oplus \cdots \oplus A^{i-1}\mathcal{W} = \bigoplus_{j=0}^{i-1} A^j \mathcal{W} \quad F_0 \coloneqq \{0\}$$

where $P \oplus Q \coloneqq \{x + y | x \in P, y \in Q\}$

Minimum Robust Invariant Set

As the sum goes to infinity, we arrive at the **minimum robust invariant set** mRPI

$$F_\infty = \bigoplus_{j=0}^{\infty} A^j \mathcal{W} \quad F_0 \coloneqq \{0\}$$

If there exists an n such that $F_n = F_{n+1}$, then $F_n = F_\infty$.

- A finite n does note always exist, but a "large" n is a good approximation.
- If n is not finite, there are other methods of computing small invariant sets, which will be slightly larger than F_∞.

Example

If we have the system :

$$x(k+1) = \left(\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} K \right) x(k) + w(k)$$

$$\mathcal{W} \coloneqq \{w | \, |w_1| \leq 0.01, |w_2| \leq 0.1\}$$

where K is the LQR controller for $Q = I$ and $R = 10$.

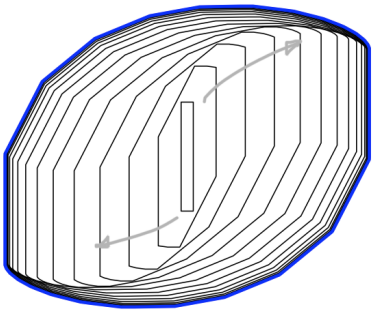


Figure 9.15: Sets $A^j\mathcal{W}$ converging to the minimal robust invariant set F_∞ at the limit.

Figure 9.16: The state trajectory will stay in the set F_∞ for all time.

Noisy System Trajectory

Given the nominal trajectory z_i, what can the noisy system trajectory do?

$$x_i = z_i + e_i$$

We don't know what the error will be at time i, but it will be in the set $\mathcal{E}$. Therefore. x_i can only be up to $\mathcal{E}$ far from z_i :

$$x_i \in z_i \oplus \mathcal{E} = \{z_i + e_i | e_i \in \mathcal{E}\}$$

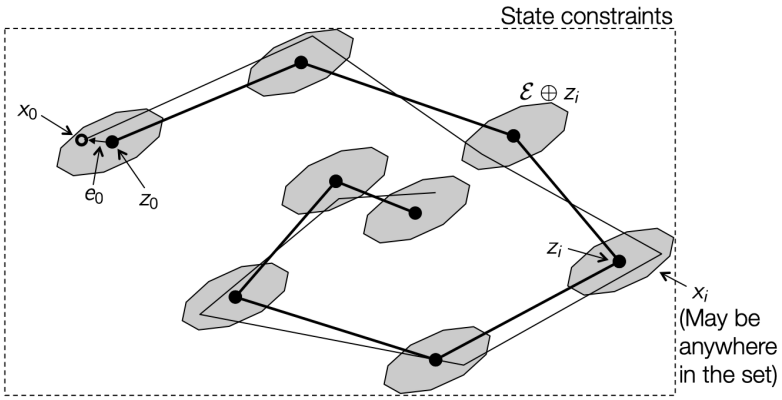


Figure 9.17

9.5.2 Constraint Tightening

Goal : $x_i, u_i \in \mathcal{X} \times \mathcal{U}$ for all $\{w_0, \ldots, w_{i-1}\} \in \mathcal{W}^i$.

We want to work with the nominal system $z(k+1) = Az(k) + Bv(k)$ but ensure that the noisy system $x(k+1) = Ax(k) + Bu(k) + w(k)$ satisfies the constraints.

Sufficient condition :

$$z_i \oplus \mathcal{E} \subseteq \mathcal{X} \qquad \longleftrightarrow \qquad z_i \in \mathcal{X} \ominus \mathcal{E}$$

The set $\mathcal{E}$ is known offline - we can compute the constraints $\mathcal{X} \ominus \mathcal{E}$ offline!

A similar condition holds for the inputs :

$$z_i \in K\mathcal{E} \oplus v_i \subset \mathcal{U} \qquad \longleftrightarrow \qquad v_i \in \mathcal{U} \ominus K\mathcal{E}$$

9.5.3 Tube-MPC Problem Formulation

$$\begin{aligned}
\text{Feasible Set :} \quad & \mathcal{Z}(x_0) \coloneqq \left\{ Z, V \;\middle|\; \begin{array}{ll} z_{i+1} = Az_i + Bv_i & i \in [0, \dots, N-1] \\ z_i \in \mathcal{X} \ominus \mathcal{E} & i \in [0, \dots, N-1] \\ v_i \in \mathcal{U} \ominus K\mathcal{E} & i \in [0, \dots, N-1] \\ z_N \in \mathcal{X}_f & \\ x_0 \in z_0 \oplus \mathcal{E} & \end{array} \right\} \\
\text{Cost Function :} \quad & J(Z, V) \coloneqq \sum_{i=0}^{N-1} l(z_i, v_i) + l_f(z_N) \\
\text{Optimization :} \quad & (V^*(x_0), Z^*(x_0)) = \arg\min_{V,Z} \{J(Z, V) | (Z, V) \in \mathcal{Z}(x_0)\} \\
\text{Control Law :} \quad & \mu_{\text{tube}}(x) \coloneqq K(x - z_0^*(x)) + v_0^*(x)
\end{aligned}$$

- Optimizing the nominal system, when the state and input constraints are tightened
- The first tube center is an optimization variable $\rightarrow$ has to be within $\mathcal{E}$ of x_0
- The cost is with respect to the tube centers (nominal system)
- The terminal set is with respect to the tightened constraints

Tube-MPC Assumptions

These are much the same as for nominal MPC :

1. The stage cost is a positive definite function, i.e. it is strictly positive and only zero at the origin.
2. The terminal set is invariant **for the nominal system** under the local control law $\kappa_f(z)$:

   $$Az + B\kappa_f(z) \in \mathcal{X}_f \qquad \text{for all } z \in \mathcal{X}_f$$

   All **tightened state and input constraints** are satisfied in $\mathcal{X}_f$:

   $$\mathcal{X}_f \subseteq \mathcal{X} \ominus \mathcal{E} \quad \kappa_f(z) \in \mathcal{U} \ominus K\mathcal{E} \text{ for all } z \in \mathcal{X}_f$$

3. The terminal cost is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$:

   $$l_f(Az + B\kappa_f(z)) - l_f(z) \leq -l(z, \kappa_f(z)) \text{ for all } z \in \mathcal{X}_f$$

Robust Invariance

Theorem 9.5.1. ***Robust Invariance of Tube-MPC***
The set $\mathcal{Z} \coloneqq \{x | \mathcal{Z}(x) \neq \emptyset\}$ is a robust invariant set of the system $x(k+1) = Ax(k) + B\mu_{tube}(x(k)) + w(k)$ subject to the constraints $x, u \in \mathcal{X} \times \mathcal{U}$.

Let $\left(\left\{v_0^*, \ldots, v_{N-1}^*\right\}, \{z_0^*, \ldots, z_N^*\}\right)$ be the optimal solution for $x(k)$. At the next point in time, the state is :

$$x(k+1) = Ax(k) + BK\left(x(k) - z_0^*\right) + Bv_0^* + w \quad \text{for some } w \in \mathcal{W}$$

i.e., the state $x(k+1)$ may have many possible values. We need to show that there exists a feasible solution for **all of them**.

By construction, the state $x(k+1)$ is in the set $z_1 \oplus \mathcal{E}$ for all $\mathcal{W}$. Therefore (as in standard MPC), the sequence

$$\left(\left\{v_0^*, \ldots, v_{N-1}^*, \kappa_f(z_N^*)\right\}, \{z_0^*, \ldots, z_N^*, Az_N^* + B\kappa_f(z_n^*)\}\right)$$

is feasible for all $x(k+1)$.

Robust Stability

Theorem 9.5.2. ***Robust Stability of Tube-MPC***
The state $x(k)$ of the system $x(k+1) = Ax(k) + B\mu_{tube}(x(k)) + w(k)$ converges in the limit to the set $\mathcal{E}$.

As in standard MPC, we have the relationship :

$$\begin{aligned} J^*\left(x(k)\right) &= \sum_{i=0}^{N-1} l\left(z_i^*, v_i^*\right) + l_f(z_N^*) \\ J^*\left(x(k+1)\right) &\leq \sum_{i=1}^{N} l\left(z_i^*, v_i^*\right) + l_f(z_{N+1}) \\ &= J^*(x(k)) - \underbrace{l\left(z_0^*, v_0^*\right)}_{\geq 0} \underbrace{-l_f(z_N^*) + l_f(z_{N+1}) + l\left(z_N^*, \kappa_f\left(z_N^*\right)\right)}_{\leq 0 \ (l_f \text{ is a Lyapunov function in } \mathcal{X}_f)} \end{aligned}$$

This shows that $\lim_{k\to\infty} J\left(z_0^*(x(k))\right) = 0$, and therefore $\lim_{k\to\infty} z_0^*(x(k)) = 0$.

However, $x(k)$ does not tend to zero! It only stays within a robust invariant set centered at $z^*(x(k))$: $\lim_{k\to\infty} \operatorname{dist}\left(x(k), \mathcal{E}\right)$, where dist is any distance function.

Tube-MPC - Putting it All Together

To implement tube MPC :

Offline
1. Choose a stabilize controller K so that $\|A + BK\| < 1$
2. Compute the minimal robust invariant set $\mathcal{E} = F_\infty$ for the system
$$x(k+1) = \left(A + BK\right)x(k) + w(k) \qquad w \in \mathcal{W}^1$$
3. Compute the tightened constraints $\tilde{\mathcal{X}} \coloneqq \mathcal{X} \ominus \mathcal{E}, \tilde{\mathcal{U}} \coloneqq \mathcal{U} \ominus K\mathcal{E}$
4. Choose the terminal weight function l_f and constraint $\mathcal{X}_f$ satisfying the previously mentioned assumptions.

Online
1. Measure / estimate the state x
2. Solve the problem $\left(V^*(x), Z^*(x)\right) = \arg\min_{V,Z}\left\{J(Z,V) | (Z,V) \in \mathcal{Z}(x)\right\}$
3. Set the input to $u = K\left(x - z_0^*(x)\right) + v_0^*(x)$

[1] Note that it is not often possible to compute the minimal robust invariant set, as it may have an infinite number of facts. Therefore, we often take an invariant outer approximation.

Offline Design - Compute Minimal Invariant Set

1. Choose a stabilizing controller K so that $\|A+BK\|<1$
2. Compute the minimal robust invariant set $\mathcal{E}=F_\infty$ for the system $x(k+1)=(A+BK)\,x(k)+w(k)$, $w\in\mathcal{W}$ We take the LQR controller for $Q=I$ and $R=1$:

$$K:=\begin{pmatrix}-0.5198 & -0.94\end{pmatrix}$$

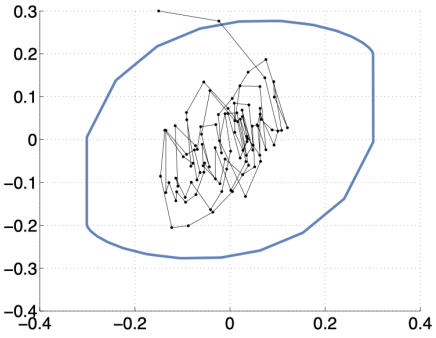


Figure 9.18

Evolution of the system :

$$x(k+1)=(A+BK)\,x(k)+w(k)$$

$$\text{for } x(0)=\begin{pmatrix}-0.15 & 0.3\end{pmatrix}^T$$

3. Compute the tightened constraints $\tilde{\mathcal{X}}:=\mathcal{X}\ominus\mathcal{E}$, $\tilde{\mathcal{U}}:=\mathcal{U}\ominus K\mathcal{E}$

$$\mathcal{X}=\{x\,|\,\|x\|_\infty\le 1\}=\left\{x\,\middle|\,\begin{pmatrix}I\\-I\end{pmatrix}x\le\begin{pmatrix}\mathbf{1}\\\mathbf{1}\end{pmatrix}\right\}$$

If $\mathcal{E}=\{x|Fx\le f\}$, then the tightened constraint sets are :

$$\mathcal{X}\ominus\mathcal{E}=\{x|x+e\in\mathcal{X}\forall e\in\mathcal{E}\}=\left\{x\,\middle|\,\begin{pmatrix}I\\-I\end{pmatrix}x+\begin{pmatrix}I\\-I\end{pmatrix}e\le\begin{pmatrix}\mathbf{1}\\\mathbf{1}\end{pmatrix}\forall e\in\mathcal{E}\right\}$$

$$=\left\{x\,\middle|\,\begin{pmatrix}I\\-I\end{pmatrix}x\le\begin{pmatrix}1-\max\left\{\begin{pmatrix}1 & 0\end{pmatrix}e|e\in\mathcal{E}\right\}\\1-\max\left\{\begin{pmatrix}0 & 1\end{pmatrix}e|e\in\mathcal{E}\right\}\\1-\max\left\{\begin{pmatrix}-1 & 0\end{pmatrix}e|e\in\mathcal{E}\right\}\\1-\max\left\{\begin{pmatrix}0 & -1\end{pmatrix}e|e\in\mathcal{E}\right\}\end{pmatrix}\right\}$$

These maximizations are all linear programs and can be computed offline.

The result is a polytope with a smaller right hand side.

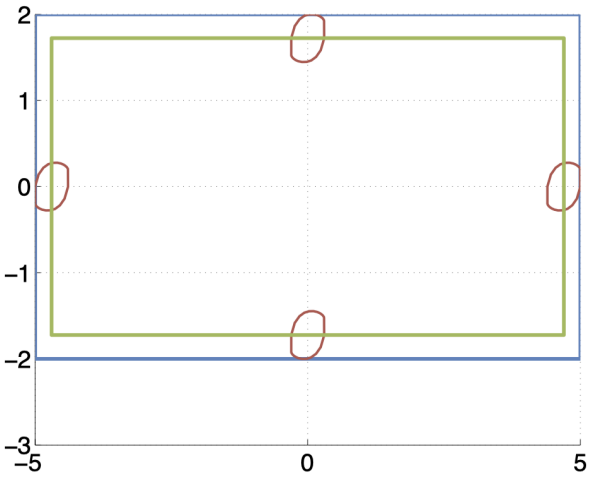


Figure 9.19: Blue : Original constraint set $\mathcal{X}$, Red : Error set $\mathcal{E}$, Green : Tightened constraints $\mathcal{X}\ominus\mathcal{E}$

We compute $\mathcal{U} \ominus K\mathcal{E}$ in the same manner :

$$\begin{aligned}\mathcal{U} \ominus K\mathcal{E} &= \left\{ u \,\middle|\, \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \leq \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\} \ominus \{Ke | Fe \leq f\} \\ &= \left\{ u \,\middle|\, \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \leq \begin{pmatrix} 1 - \max\{Ke | Fe \leq f\} \\ 1 + \max\{Ke | Fe \leq f\} \end{pmatrix} \right\}\end{aligned}$$

Offline Design - Terminal Weights & Constraints

We need to find a function l_f and a set $\mathcal{X}$ that satisfy the conditions listed at the beginning of this chapter :

1. The terminal set is invariant **for the nominal system** under the local control law $\kappa_f(z)$:

   $$Az + B\kappa_f(z) \in \mathcal{X}_f \quad \text{for all } z \in \mathcal{X}_f$$

   All **tightened state and input constraints** are satisfied in $\mathcal{X}_f$:

   $$\mathcal{X}_f \subseteq \mathcal{X} \ominus \quad \kappa_f(z) \in \mathcal{U} \ominus K\mathcal{E} \text{ for all } z \in \mathcal{X}_f$$

2. The terminal cost is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$:

   $$l_f\left(Az + B\kappa_f(z)\right) - l_f(z) \leq -l\left(z, \kappa_f(z)\right) \quad \text{for all } z \in \mathcal{X}_f$$

Offline Design - Terminal Constraint

We base our terminal weights and constraints on the LQR controller (many other choices are possible).

- Choose the terminal control law to the LQR control law : $\kappa_f(x) = Kx$ where the weights Q and R are taken as the same for our MPC problem.
- We need a set $\mathcal{X}_f$ that is invariant under this controller and contained in the tightened constraints. It can be obtained similarly as in the nominal MPC case.
- Recall the optimal cost of the LQR control law :

  $$J^*(z_0) = \sum_{i=0}^{\infty} z_i^T \left(Q + K^T R K\right) z_i = z_0^T P z_0$$

  where P is the solution to a discrete-time Riccati equation.
  We know that $J^*(z)$ is a Lyapunov function for $z(k+1) = (A + BK)\, z(k)$ and we can take $l_f = z^T P z$.

9.5.4 Example - Tube MPC

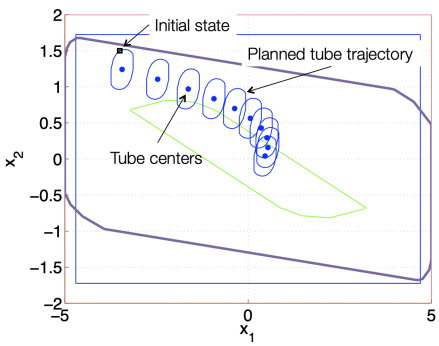


Figure 9.20

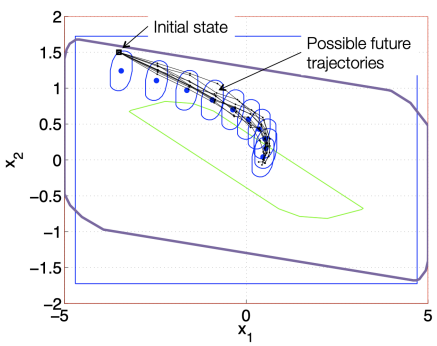


Figure 9.21

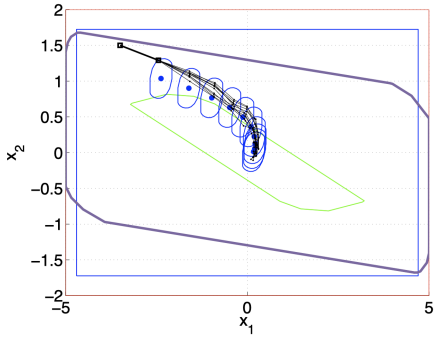


Figure 9.22

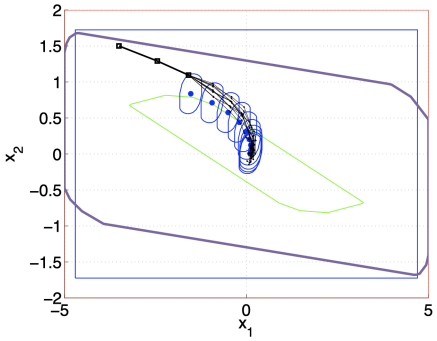


Figure 9.23

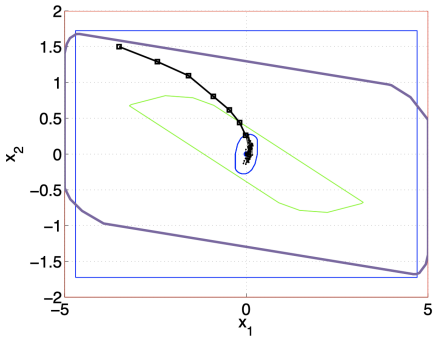


Figure 9.24

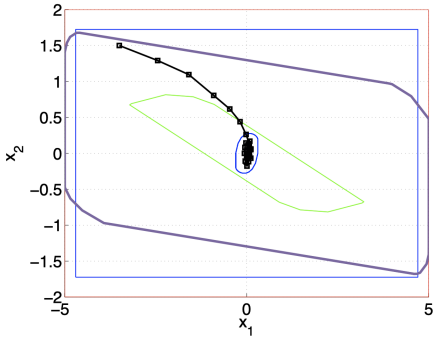


Figure 9.25

Summary - Robust MPC for Uncertain Systems

Idea

- Compensate for noise in prediction to ensure all constraints will be met.

Cons

- Complex (some schemes are simple to implement, like tubes, but complex to understand)
- Must know the largest noise $\mathcal{W}$
- Often conservative
- Feasible set may be small

Benefits

- Feasible set is invariant - We know exactly when the controller will work
- Easier to tune - Knobs to trade-off robustness against performance

Chapter 10

Robustness of Nominal MPC

10.1 Nominal MPC with Noise

We want to control the noisy system :

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

But what happens if we just ignore the noise and hope for the best?

Setup and solve a standard MPC problem :

$$\begin{aligned} J^*(x_0) = \min_U & \sum_{i=0}^{N-1} l\left(x_i, u_i\right) + l_f\left(x_N\right) \\ \text{subject to} \quad & x_{i+1} = Ax_i + Bu_i \\ & x_i, u_i \in \mathcal{X} \times \mathcal{U} \\ & x_N \in \mathcal{X}_f \end{aligned}$$

Our closed-loop system is now :

$$x(k+1) = Ax(k) + Bu_0^*\left(x(k)\right) + w(k)$$

$\hookrightarrow$ We can prove if there is convergence to a neighborhood of the origin (for linear systems)!

10.1.1 Example

If we consider the same example again, with the same noise, but now we just pretend it's not there in the controller.

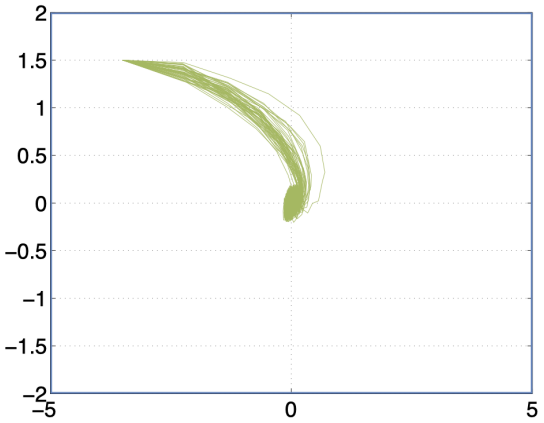


Figure 10.1

In Figure 10.1, we plot 100 trajectories with different noise realisations. But everything seems to work fine.

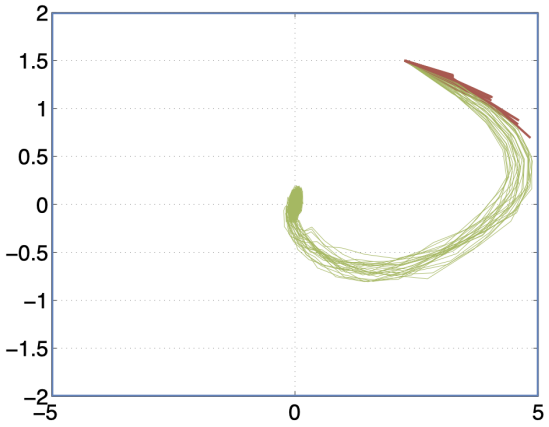


Figure 10.2

However, as we see in Figure 10.2, we can no longer be certain that it will work, the truth is, for some states it will work, sometimes. But how can we formalize this?

What happens to our Lyapunov function?

Recall : The optimal cost $J^*(x)$ is a Lyapunov function for the nominal system

$$J^*\left(Ax + Bu^*(x)\right) - J^*(x) \leq -l\left(x, u^*(x)\right)$$

However, our state at the next point in time is now

$$x(k+1) = Ax(k) + Bu^*(x(k)) + w(k)$$

Do we still have a Lyapunov decrease?

If we assume that our optimal cost J^* is continuous[1] :

$$|J^*\left(Ax + Bu^*(x) + w\right) - J^*\left(Ax + Bu^*(x)\right)| \leq \gamma \|Ax + Bu^*(x) + w - \left(Ax + Bu^*(x)\right)\| = \gamma\|w\|$$

Our Lyapunov decrease can be bounded as :

$$\begin{aligned}
&J^*\left(Ax + Bu^*(x) + w\right) - J^*(x) \\
&= J^*\left(Ax + Bu^*(x) + w\right) - J^*(x) - J^*\left(Ax + Bu^*(x)\right) + J^*\left(Ax + Bu^*(x)\right) \\
&\leq J^*\left(Ax + Bu^*(x)\right) - J^*(x) + \gamma\|w\| \\
&\leq -l\left(x, u^*(x)\right) + \gamma\|w\|
\end{aligned}$$

- The decrease grows with $\|x\|$
- The increase growth is upper-bounded by $\max\{\|w\| | w \in \mathcal{W}\}$

Therefore we will move towards the origin until there is a balance between the size of x and the size of w.

[1] This is true for linear systems, convex constraints and continuous stage costs.

10.2 Input-to-State Stability

What we have shown is that our system in **input-to-state stable**.

Asymptotic stability :

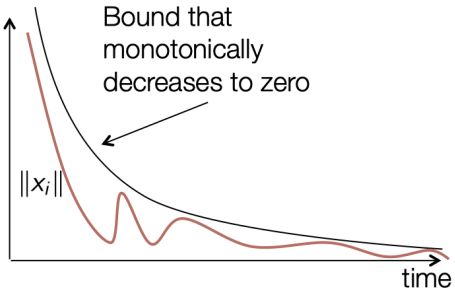


Figure 10.3

The system converges to zero.

ISS stability :

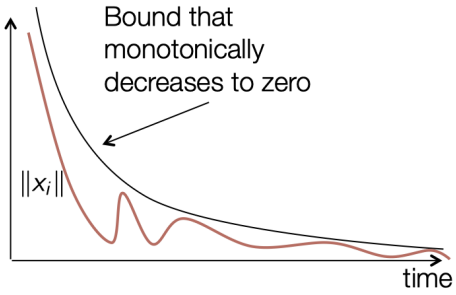


Figure 10.4

The system converges to a set around zero, who's size is determined by the size of the noise.

Summary - Nominal MPC for Uncertain Systems

Idea

- Ignore the noise and hope it works

Benefits

- Simple
- No knowledge of the noise set $\mathcal{W}$ required
- Often very effective in practice (this is what most practitioners say)
- Feasible set is large (we can find a solution, but there is no guarantee that it will work)
- The region of attraction may be larger than other approaches

Cons

- Very difficult to determine the region of attraction (set of states in which the controller works)
- Hard to tune - no obvious way to trade-off robustness against performance
- Works for linear systems, for non-linear systems only under continuity assumptions.