SSY280
Model Predictive Control

Bo Egardt

Department of Signals and Systems
Chalmers University of Technology

January, 2017

Lecture 1: Introduction and course overview

Goals for today:

- To know about the background of MPC and the ideas behind the course
- To get motivated by being reminded about the limitations of linear design
- To understand the receding horizon control (RHC) idea
- To understand how the course is organized
- To understand the learning objectives of the course

Background of MPC

- The origin:
  - Process control in petrochemical industry (1970ies)
  - Dynamic Matrix Control (DMC)
- Early drivers:
  - Operation close to limits $\Rightarrow$ linear control shortcomings
  - Large returns on small improvements in e.g. product quality or energy/material consumption
  - Slow time scale allows time-consuming computations
- Limitations of linear control design:
  - Saturation on control and control rates
  - Process output limitations
  - Buffer control (zone objectives)
- Important attributes of MPC:
  - MPC handles actuator limitations and process constraints
  - MPC handles multivariable systems
  - MPC is model based (step responses, state-space models etc.)
- Current driving factors:
  - Development of theory and new application areas
  - Improvements in numerical computations

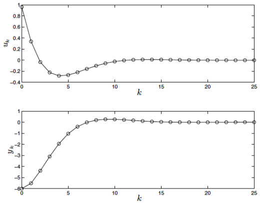


Figure 1.2. u_k and y_k for the cautious design $u_k = -Kx_k$ with weights $Q = C^{\mathsf{T}}C$ and $R = 20$.

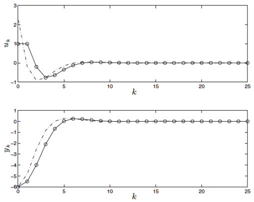


Figure 1.3. u_k and y_k for the unconstrained LQR design $u_k = -Kx_k$ (dashed line), and for the serendipitous strategy $u_k = -\mathrm{sat}(Kx_k)$ (circle-solid line), with weights $Q = C^{\mathsf{T}}C$ and $R = 2$.

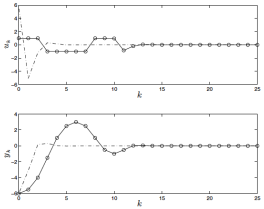


Figure 1.4. u_k and y_k for the unconstrained LQR design $u_k = -Kx_k$ (dashed line), and for the serendipitous strategy $u_k = -\mathrm{sat}(Kx_k)$ (circle-solid line), with weights $Q = C^{\mathsf{T}}C$ and $R = 0.1$.

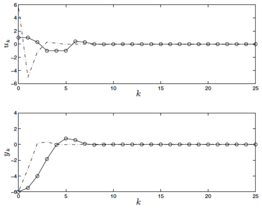


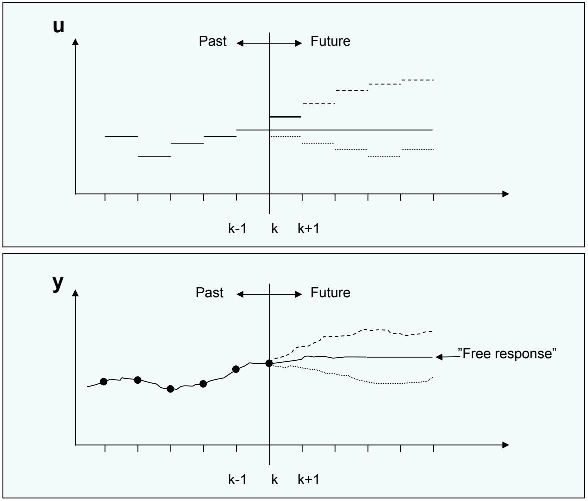
Figure 1.6. u_k and y_k for the unconstrained LQR design $u_k = -Kx_k$ (dashed line), and for the receding horizon design (circle-solid line), with weights $Q = C^{\mathsf{T}}C$ and $R = 0.1$.

The receding horizon idea (1)

The core of the MPC approach, the receding horizon idea:

1. At time instant k, *predict* the process response over a finite *prediction horizon* N; this response depends on the sequence of future control inputs over the *control horizon* M.
2. Pick the control sequence which gives the best performance in terms of a specified *objective*, *cost function* or *criterion*.
3. Apply the first element in the control sequence to the process, discard the rest of the sequence, and return to step 1.

The receding horizon idea (2)



Summary

The MPC recipe for the example:

1. At time k, *predict* the output N samples ahead:
$$\hat{y}(k+1|k), \ldots, \hat{y}(k+N|k)$$
2. The predictions depend on future control inputs
$$u(k|k), u(k+1|k), \ldots, u(k+M-1|k)$$
(Normally, $M < N$, and we assume that u is either 0 or unchanged after this.)
3. Minimize a criterion (now adopting the index notation : as in Matlab)
$$V(k) = V(\hat{y}(k+1\!:\!k+N|k), u(k\!:\!k+M-1|k))$$
with respect to the control sequence $u(k\!:\!k+M-1|k)$
4. Apply the first control signal in the sequence to the process:
$$u(k) = u(k|k)$$
5. Increment time $k := k+1$ and go to 1.

MPC ingredients

- An *internal model* describing process and disturbances
- An *estimator/predictor* to determine the evolution of the state
- An *objective/criterion* to express the desired system behaviour
- An *online optimization algorithm* to determine future control actions
- The *receding horizon* principle
- Our focus: linear models, quadratic criteria with linear constraints

Outline of the course

- What is MPC? Introduction, motivation and review
- MPC basics and relations to dynamic programming
- How to cope with incomplete state information
- Solving predictive control problems
- User aspects, case study, and company visit
- MPC theory
- Review and outlook

Learning objectives

After completion of the course, you should be able to:

- Understand and explain the basic principles of model predictive control, its pros and cons, and the challenges met in implementation and applications
- Correctly state, in mathematical form, MPC formulations based on descriptions of control problems expressed in application terms
- Describe and construct MPC controllers based on a linear model, quadratic costs and linear constraints
- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples
- Understand and explain basic properties of the optimization problem as an ingredient of MPC, in particular concepts like linear, quadratic and convex optimization, optimality conditions, and feasibility
- Use software tools for analysis and synthesis of MPC controllers

Lecture 2: Review and preliminaries

Goals for today:

- To master different discrete-time state space models used for MPC
- To know how to use quadratic forms for stability investigations of linear systems
- To refresh basic concepts used to solve systems of linear equations
- To refresh and learn a new way to test for controllability and observability
- To understand conditions for setpoint tracking and disturbance rejection
- To refresh how to handle LTI systems in Matlab

Learning objectives:

- Understand and explain the basic principles of model predictive control, its pros and cons, and the challenges met in implementation and applications
- Use software tools for analysis and synthesis of MPC controllers

State space model

Continuous state space model:

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t) \\ y(t) &= C_y x(t) \\ z(t) &= C_z x(t) \end{aligned} \tag{1}$$

$x = (x_1, \dots, x_n)$ is the state vector

$u = (u_1, \dots, u_m)$ is the control input vector

$y = (y_1, \dots, y_{p_y})$ is the vector of *measured* outputs

$z = (z_1, \dots, z_{p_z})$ is the vector of *controlled* outputs

Often, $z = y = (y_1, \dots, y_p)$, and then we simply write $y(t) = Cx(t)$.

Discrete state space model

The solution of (1) with initial condition $x(t_0)$ is

$$x(t) = e^{A_c(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A_c(t-s)}B_c u(s)ds \tag{2}$$

Assume that the control signal u is piecewise constant (h is the sampling interval):

$$u(t) = u(kh), \quad kh \leq t < (k+1)h$$

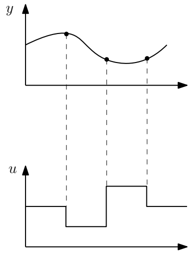
By using this in (2) with $t = (k+1)h$ and $t_0 = kh$, we get the discrete time state equation

$$x(k+1) = e^{A_c h}x(k) + \Big(\int_0^h e^{A_c s}B_c\, ds\Big)u(k) = Ax(k) + Bu(k),$$

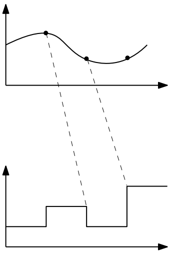
where, for simplicity of notation, h has been omitted from the time argument. The output equation of (1) is the same (with t replaced by k). A compact version of the discrete state-space model in the case $z = y$ is

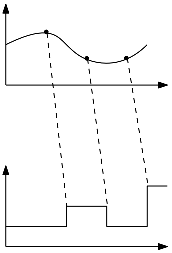
$$x^+ = Ax + Bu$$
$$y = Cx$$

Sampling and computational delay



(a) Instantaneous computation


(b) Full sampling interval delay


(c) Fractional delay

Computational delay (1)

Allowing for a computational delay τ, the control signal is still piecewise constant but now given by

$$u(t) = \begin{cases} u(k-1), & kh \le t < kh + \tau \\ u(k), & kh + \tau \le t < (k+1)h \end{cases}$$

The solution (2) is now obtained in two steps (corresponding to the two different input signal levels) as

$$\begin{aligned} x(k+1) &= e^{A_c(h-\tau)} x(kh+\tau) + \int_0^{h-\tau} e^{A_c s} B_c \, ds \cdot u(k) \\ &= e^{A_c(h-\tau)} \big(e^{A_c \tau} x(kh) + \int_0^{\tau} e^{A_c s} B_c \, ds \cdot u(k-1) \big) + \int_0^{h-\tau} e^{A_c s} B_c \, ds \cdot u(k) \\ &= e^{A_c h} x(kh) + e^{A_c(h-\tau)} \int_0^{\tau} e^{A_c s} B_c \, ds \cdot u(k-1) + \int_0^{h-\tau} e^{A_c s} B_c \, ds \cdot u(k) \\ &= Ax(k) + B_1 u(k-1) + B_2 u(k) \end{aligned}$$

Computational delay (2)

This can be put in standard form by introducing the augmented state vector

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}$$

The new state space model becomes

$$\xi(k+1) = \begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix} = \begin{bmatrix} A & B_1 \\ 0 & 0 \end{bmatrix} \xi(k) + \begin{bmatrix} B_2 \\ I \end{bmatrix} u(k)$$

State space model with incremental control

Consider the state space model

$$\begin{aligned} x^+ &= Ax + Bu \\ y &= Cx \end{aligned}$$

Introduce the incremental control (or *control move*)

$$\Delta u(k) = u(k) - u(k-1)$$

and the augmented state vector

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}$$

This leads to the new model

$$\begin{aligned} \xi^+ &= \mathcal{A}\xi + \mathcal{B}\Delta u \\ y &= \mathcal{C}x \end{aligned}$$

with

$$\mathcal{A} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \quad \mathcal{B} = \begin{bmatrix} B \\ I \end{bmatrix} \quad \mathcal{C} = \begin{bmatrix} C & 0 \end{bmatrix}$$

Quadratic forms

A *quadratic form* is given by the expression

$$x^T S x = |x|_S^2$$

with $x \in \mathbb{R}^n$ and S a real, symmetric matrix.
The matrix S is positive definite ($S \succ 0$) if and only if

$$x^T S x > 0, \quad \forall \text{ nonzero } x \in \mathbb{R}^n$$

and the matrix S is positive semidefinite ($S \succeq 0$) if and only if

$$x^T S x \geq 0, \quad \forall x \in \mathbb{R}^n$$

Stability

The linear, discrete time system

$$x^+ = Ax \tag{3}$$

is asymptotically stable (solutions converge to the origin) if and only if the magnitudes of the eigenvalues of A are strictly less than 1 (A is *stable*).

The evolution of the state trajectory can be studied via the quadratic form

$$V(x) = x^T S x = |x|_S^2, \quad S \succeq 0$$

which can be interpreted as a generalization of the Euclidean norm $|x|$ (squared). Evaluated along solutions to (3), V changes according to

$$V(x^+) - V(x) = x^T(A^T S A - S)x \mathrel{\widehat{=}} -x^T Q x$$

Hence, the matrix Q determines whether V will decay or not. In fact, the following conditions are equivalent:

(a) The matrix A is stable

(b) For each $Q \succ 0$ there is a unique solution $S \succ 0$ of the *discrete Lyapunov equation*

$$A^T S A - S = -Q \tag{4}$$

Systems of linear equations (1)

For any $m \times n$ matrix A ($A \in \mathbb{R}^{m\times n}$) we have:

Rank of A :	$\operatorname{rank}(A) = \operatorname{rank} A^T = r$	
Range of A :	$\mathcal{R}(A) = \left\{Ax \mid x \in \mathbb{R}^n\right\}$	$\dim \mathcal{R}(A) = r$
Nullspace of A :	$\mathcal{N}(A) = \left\{x \mid Ax = 0\right\}$	$\dim \mathcal{N}(A) = n - r$
Orthogonal subspaces:	$\mathcal{R}(A^T) \perp \mathcal{N}(A)$	$\mathcal{R}(A) \perp \mathcal{N}(A^T)$

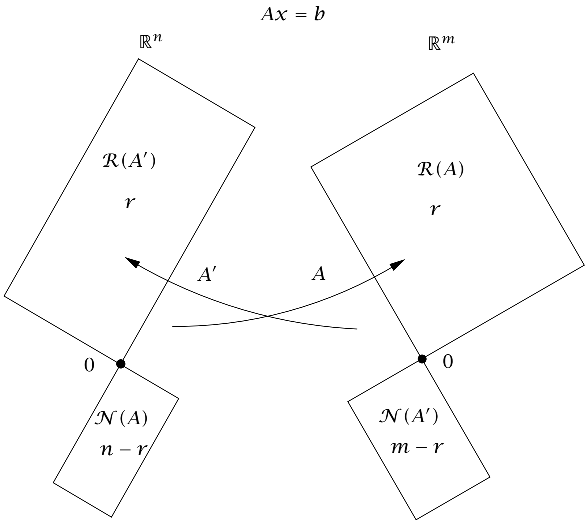
For the system of linear equations

$$Ax = b, \quad A \in \mathbb{R}^{m\times n}$$

we have:

- There exists a solution x for every b if and only if $r = m$, i.e. $\mathcal{R}(A) = \mathbb{R}^m$
- The solution is unique if and only if $r = n$, i.e. $\mathcal{N}(A) = \{0\}$

Systems of linear equations (2)



Systems of linear equations (3)

Consider the overdetermined system of linear equations

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}$$

with $m > n$. Assume that A has maximal rank $r = n$. Then the solution to the minimization problem

$$\min_{x \in \mathbb{R}^n} |Ax - b|$$

is given by

$$x^* = A^\dagger b$$

where the *pseudo-inverse* $A^\dagger$ is defined by

$$A^\dagger = (A^T A)^{-1} A^T$$

Remark. $Ax^* = AA^\dagger b$ is the orthogonal projection of b onto $\mathcal{R}(A)$, i.e. x^* is mapped to the vector in $\mathcal{R}(A)$ that is closest to b.
In Matlab notation, $x^* = A \backslash b$.

Controllability

A linear, discrete time system

$$x^+ = Ax + Bu$$

is controllable if it is possible to steer the system from any state x_0 to any state x_f in finite time.

The system is controllable if and only if any of the following, equivalent, conditions hold:

- The matrix $\begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$ has full rank n
- The matrix $\begin{bmatrix} \lambda I - A & B \end{bmatrix}$ has rank n for all $\lambda \in \mathbb{C}$

Note. A weaker condition is *stabilizability* which requires that any uncontrollable modes are strictly stable.

Observability

A linear, discrete time system

$$x^+ = Ax$$
$$y = Cx$$

is observable if for some N, any $x(0)$ can be determined from $\{y(0), y(1), \ldots, y(N-1)\}$.

The system is observable if and only if any of the following, equivalent, conditions hold:

- The matrix $\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$ has full rank n
- The matrix $\begin{bmatrix} \lambda I - A \\ C \end{bmatrix}$ has rank n for all $\lambda \in \mathbb{C}$

Note. A weaker condition is *detectability*, which requires that any unobservable modes are strictly stable.

Setpoint tracking

Consider the system

$$x^+ = Ax + Bu \tag{5}$$

$$y = Cx \tag{6}$$

A steady state (x_s, u_s) of the system satisfies the equation

$$\begin{bmatrix} I - A & -B \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = 0$$

The task to bring the system output y to a desired, constant setpoint y_{sp} is termed *setpoint tracking*. At steady state, this requires $Cx_s = y_{sp}$ and the condition for setpoint tracking becomes

$$\begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ y_{sp} \end{bmatrix} \tag{7}$$

This is a system of $n + p$ equations with $n + m$ unknowns.

Regulation problem for deviation variables

Assume that equation (7) holds. Then the deviation variables

$$\delta x(k) = x(k) - x_s$$
$$\delta u(k) = u(k) - u_s$$

satisfy the following dynamics

$$\delta x(k+1) = Ax(k) + Bu(k) - x_s = Ax(k) + Bu(k) - (Ax_s + Bu_s) = A\delta x(k) + B\delta u$$
$$\delta y(k) = y(k) - y_{sp} = Cx(k) - Cx_s = C\delta x(k)$$

Thus, the deviation variables satisfy the same state equation as the original variables.

Constant disturbance modeling (1)

A constant disturbance d of dimension n_d can be modeled by the state equation

$$d^+ = d$$

Augmenting the state model (5) with this disturbance gives the model

$$\begin{aligned} \begin{bmatrix} x \\ d \end{bmatrix}^+ &= \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u \\ y &= \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} \end{aligned} \tag{8}$$

where the disturbance affects both state and output equations through the matrices B_d and C_d.

Constant disturbance modeling (2)

Proposition (Detectability of the augmented system)
The augmented system (8) is detectable if and only if the original system is detectable and the following condition holds:

$$\text{rank} \begin{bmatrix} I - A & -B_d \\ C & C_d \end{bmatrix} = n + n_d \tag{9}$$

Remark. In order for the rank condition to be satisfied, we must have

$$n_d \le p$$

i.e. we must have at least as many measured outputs as the dimension of the disturbance vector.

LTI systems in Matlab

Example (Paper machine headbox – Ex 2.4 in [?])

```
% Create continuous time LTI object
Ac = [-1.93   0       0      0;
       0.394 -0.426   0      0;
       0      0      -0.63   0;
       0.82  -0.784   0.413 -0.426 ];
Bc = [ 1.274  1.274;
       0      0;
       1.34  -0.65;
       0      0 ];
Cc = [ 0      1       0      0;
       0      0       1      0;
       0      0       0      1 ];
Dc = zeros(3,2);
csys = ss(Ac,Bc,Cc,Dc)  %create state space model
```

LTI systems in Matlab, cont'd

```
% Assign variable names
set(csys,'InputName',{'Stock flowrate';'WW flowrate'},...
        'OutputName',{'Headbox level';'Feed tank conc';'Headbox conc
        'StateName',{'Feed tank level';'Headbox level';...
                    'Feed tank conc';'Headbox conc'},...
        'TimeUnit','minutes');
get(csys)

% Compute eigenvalues
eig(Ac)

% Create discrete time model
Ts=2;
dsys=c2d(csys,Ts)
```

LTI systems in Matlab, cont'd

```
% Compute eigenvalues
eig(dsys.a)
exp(eig(Ac)*Ts)

% Controllability
rank(ctrb(dsys))
rank(ctrb(dsys.a,dsys.b(:,1)))
rank(ctrb(dsys.a,dsys.b(:,2)))

% Observability
rank(obsv(dsys))
rank(obsv(dsys.a,dsys.c(1,:)))
rank(obsv(dsys.a,dsys.c(2,:)))
rank(obsv(dsys.a,dsys.c(3,:)))

% Step response
step(dsys)
```

Lecture 3: Unconstrained receding horizon control

Goals for today:

- To master the formulation of linear quadratic control (LQ)
- To understand how dynamic programming can be used to solve the LQ problem
- To formulate and solve the finite-time LQ problem using the "batch approach"
- To formulate an unconstrained receding horizon control based on LQ
- To understand the objectives of Assignment 1

Learning objectives:

- Correctly state, in mathematical form, MPC formulations based on descriptions of control problems expressed in application terms
- Describe and construct MPC controllers based on a linear model, quadratic costs and linear constraints
- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples

The LQ problem

System:

$$S: \quad x^+ = Ax + Bu \tag{10}$$

Optimization problem:

$$P: \quad \min_{u(0:N-1)} V_N(x(0), u(0:N-1))$$

where the minimization is with respect to the sequence of control inputs

$$u(0:N-1) = \{u(0), u(1), \ldots, u(N-1)\}$$

and subject to the system model (10).

The *objective* or *criterion* or *cost function* V_N is given by

$$\begin{aligned} V_N(x(0), u(0:N-1)) &= \sum_{i=0}^{N-1} (x^T(i)Qx(i) + u^T(i)Ru(i)) + x^T(N)P_f x(N) \\ &= \sum_{i=0}^{N-1} l(x(i), u(i)) + l_f(x(N)) \end{aligned} \tag{11}$$

Remark 1: All $x(i)$ are functions of $x(0)$ and $u(0:N-1)$ via the model (10)!

Remark 2: The first term $x^T(0)Qx(0)$ in the objective is really redundant but is kept for notational convenience.

Solution to the LQ problem

Sequence of optimal control laws (*control policy*):

$$u^0(k;x) = K(k)x, \quad k = 0, \dots, N-1$$
$$K(k) = -(R + B^T P(k+1)B)^{-1} B^T P(k+1)A$$

Riccati equation:

$$P(k-1) = Q + A^T P(k)A - A^T P(k)B(R + B^T P(k)B)^{-1} B^T P(k)A, \quad P(N) = P_f \tag{12}$$

Optimal cost-to-go (from time k to time N):

$$V^0_{k \to N}(x) = x^T P(k)x$$

Batch solution (1)

Repeated use of the system equation (10) gives

$$\begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix} \tag{13}$$

or, with a more compact notation,

$$\boldsymbol{x} = \Omega x(0) + \Gamma \boldsymbol{u} \tag{14}$$

The LQ criterion (11) can now be written

$$\begin{aligned} V_N(x(0), \boldsymbol{u}) &= x^T(0)Qx(0) + \boldsymbol{x}^T \bar{Q} \boldsymbol{x} + \boldsymbol{u}^T \bar{R} \boldsymbol{u} \\ &= x^T(0)Qx(0) + (\Omega x(0) + \Gamma \boldsymbol{u})^T \bar{Q} (\Omega x(0) + \Gamma \boldsymbol{u}) + \boldsymbol{u}^T \bar{R} \boldsymbol{u} \\ &= \boldsymbol{u}^T (\Gamma^T \bar{Q} \Gamma + \bar{R}) \boldsymbol{u} + 2x^T(0) \Omega^T \bar{Q} \Gamma \boldsymbol{u} + x^T(0)(Q + \Omega^T \bar{Q} \Omega) x(0) \end{aligned} \tag{15}$$

where $\bar{Q} = \operatorname{diag}(Q, \ldots, Q, P_f)$ and $\bar{R} = \operatorname{diag}(R, \ldots, R)$.

Batch solution (2)

Since $V_N(x(0), \boldsymbol{u})$ is quadratic in $\boldsymbol{u}$, we can solve for the optimal control vector (either by differentiation or by completing the squares):

$$\boldsymbol{u}^0 = -(\Gamma^T \bar{Q} \Gamma + \bar{R})^{-1} \Gamma^T \bar{Q} \Omega \, x(0)$$

and the optimal cost-to-go

$$V_N^0(x(0), \boldsymbol{u}) = x^T(0) \big(Q + \Omega^T \bar{Q} \Omega - \Omega^T \bar{Q} \Gamma (\Gamma^T \bar{Q} \Gamma + \bar{R})^{-1} \Gamma^T \bar{Q} \Omega \big) x(0)$$

Note that $\boldsymbol{u}^0$ is linear in $x(0)$ and that V_N^0 is quadratic in $x(0)$!

Infinite horizon LQ control

System:

$$S: \quad x^+ = Ax + Bu \tag{16}$$

Optimization criterion:

$$V_\infty(x(0), u(0\colon\infty)) = \sum_{i=0}^{\infty}(x^T(i)Qx(i) + u^T(i)Ru(i)) \tag{17}$$

If the pair (A, B) is controllable and $Q, R \succ 0$, then the solution to this infinite horizon LQ control problem gives a stable closed-loop system

$$x^+ = Ax + BKx$$

with a time invariant feedback gain K, given by

$$K = -(B^TPB + R)^{-1}B^TPA \tag{18}$$

$$P = Q + A^TPA - A^TPB(B^TPB + R)^{-1}B^TPA \tag{19}$$

The latter equation is called the *algebraic Riccati equation*. The optimal cost is given by

$$V_\infty^0(x(0)) = x^T(0)Px(0)$$

Lecture 4: Constrained receding horizon control

Goals for today:

- To understand the principles behind and to formulate a constrained receding horizon controller (RHC)
- To formulate an MPC based on linear models and quadratic criteria

Learning objectives:

- Describe and construct MPC controllers based on a linear model, quadratic costs and linear constraints
- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples

Infinite horizon optimal control

Cost function:

$$V_\infty(x_0, u(0\!:\!\infty)) = \sum_{i=0}^{\infty} l(x(i), u(i)),$$

with $l(\cdot,\cdot) \geq 0$ and $l(0,0) = 0$.

Optimization problem:

$$\begin{aligned} \min_{u(0:\infty)} \quad & V_\infty(x_0, u(0\!:\!\infty)) && (20)\\ \text{subject to} \quad & x^+ = f(x,u), \quad x(0) = x_0 && (21)\\ & x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad \text{for all } k \in (0,\infty) && (22) \end{aligned}$$

Finite horizon optimal control

Cost function:

$$V_N(x_0, u(0\colon N-1)) = \sum_{i=0}^{N-1} l(x(i), u(i)) + V_f(x(N)),$$

with the final cost $V_f(\cdot) \geq 0$ and $V_f(0) = 0$.

Optimization problem:

$$\begin{aligned}
\min_{u(0:N-1)} \quad & V_N(x_0, u(0\colon N-1)) && (23)\\
\text{subject to} \quad & x^+ = f(x,u), \quad x(0) = x_0 && (24)\\
& x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad \text{for all } k \in (0, N-1) && (25)\\
& x(N) \in \mathbb{X}_f \subseteq \mathbb{X} && (26)
\end{aligned}$$

Note that we have added a *terminal constraint* $x(N) \in \mathbb{X}_f$!
It is assumed that $\mathbb{U}$, $\mathbb{X}$, and $\mathbb{X}_f$ all contain the origin.

Solution to the finite time optimization problem

The following notation for the optimal solution will be used:

Optimal cost-to-go:

$$V_N^0(x_0) = \min_{u(0:N-1)} \{V_N(x_0, u(0:N-1)) \mid u(0:N-1) \in \mathcal{U}_N(x_0)\}$$

Optimal control and state sequences:

$$u^0(0:N-1;x_0) = \{u^0(0;x_0), u^0(1;x_0), \ldots, u^0(N-1;x_0)\}$$
$$x^0(0:N;x_0) = \{x^0(0;x_0), x^0(1;x_0), \ldots, x^0(N;x_0)\}$$

Receding horizon control

1. At sampling instant k, when we have access to the state $x(k) = x$ (assumed to be within he feasible set), solve the optimization problem

$$V_N^0(x) = \min_{u(0:N-1)} \{ V_N(x, u(0:N-1)) \mid u(0:N-1) \in \mathcal{U}_N(x) \}$$

for the optimal control sequence

$$u^0(0:N-1; x) = \{ u^0(0; x), u^0(1; x), \ldots, u^0(N-1; x) \}$$

2. Apply the first control input

$$u(k) = u^0(0; x)$$

3. Let $k := k + 1$ and go to (1).

Note that the receding horizon controller described this way *implicitly* defines a feedback control law for all x belonging to the feasible set:

$$u(x) = \kappa_N(x) \mathrel{\widehat{=}} u^0(0; x), \quad x \in \mathcal{X}_N$$

Linear quadratic MPC (1)

1. At sampling instant k, when we have access to the state $x(k) = x$, solve the optimization problem

$$V_N^0(x) = \min_{u(0:N-1)} \{V_N(x, u(0:N-1)) \mid u(0:N-1) \in \mathcal{U}_N(x)\},$$

$$V_N(x, u(0:N-1)) = x^T(N)P_f x(N) + \sum_{i=0}^{N-1} \left(x^T(i)Qx(i) + u^T(i)Ru(i)\right);\ x(0) = x$$

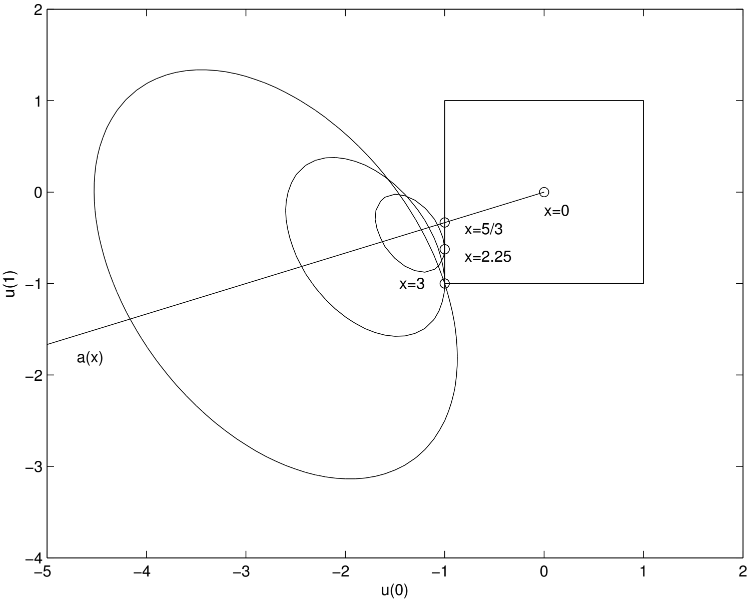
   for the optimal control sequence

$$u^0(0:N-1; x) = \{u^0(0; x), u^0(1; x), \ldots, u^0(N-1; x)\}$$

2. Apply the first control input

$$u(k) = u^0(0; x)$$

3. Let $k := k + 1$ and go to (1).



x=0
x=5/3
x=2.25
x=3
a(x)
u(0)
u(1)
2
1
0
−1
−2
−3
−4
−5
−4
−3
−2
−1
0
1
2

Linear quadratic MPC (2)

1. At sampling instant k, when we have access to the state $x(k) = x$, solve the optimization problem

$$V_N^0(x) = \min_{u(0:N-1),x(0:N)} \{V_N(x, u(0:N-1), x(0:N))\},$$

$$V_N(x, u(0:N-1), x(0:N)) = x^T(N)P_f x(N) + \sum_{i=0}^{N-1} \left(x^T(i)Qx(i) + u^T(i)Ru(i)\right),$$

   for the optimal control and state sequences

$$u^0(0:N-1; x) = \{u^0(0; x), u^0(1; x), \dots, u^0(N-1; x)\}$$
$$x^0(0:N; x) = \{x^0(0; x), x^0(1; x), \dots, x^0(N; x)\}$$

   and subject to

$$x(k+1) = Ax(k) + Bu(k), \quad x(0) = x \tag{27}$$
$$x(k) \in \mathbb{X} \quad u(k) \in \mathbb{U} \quad \text{for all } k \in (0, N-1) \tag{28}$$
$$x(N) \in \mathbb{X}_f \subseteq \mathbb{X} \tag{29}$$

2. Apply the first control input $u(k) = u^0(0; x)$
3. Let $k := k + 1$ and go to (1).

Linear quadratic MPC (3)

1. At sampling instant k, when we have access to the state $x(k) = x$, solve the optimization problem

$$\begin{aligned} \underset{\boldsymbol{u},\boldsymbol{x}}{\text{minimize}} \quad & V_N(x, \boldsymbol{u}, \boldsymbol{x}) = x^T Q x + \boldsymbol{x}^T \bar{Q} \boldsymbol{x} + \boldsymbol{u}^T \bar{R} \boldsymbol{u} \\ \text{subject to} \quad & \boldsymbol{x} = \Omega x + \Gamma \boldsymbol{u} \\ & F\boldsymbol{u} + G\boldsymbol{x} \leq h \end{aligned}$$

2. Apply the first control input $u(k) = \boldsymbol{u}(1)$
3. Let $k := k + 1$ and go to (1).

Lecture 5: Setpoints, disturbances and observers

Goals for today:

- To understand how the state model is used for open loop prediction
- To formulate the DMC scheme with a constant output disturbance
- To interpret the DMC scheme in terms of state observers
- To formulate an MPC controller including setpoints and steady state targets
- To state conditions for offset-free control

Learning objectives:

- Correctly state, in mathematical form, MPC formulations based on descriptions of control problems expressed in application terms
- Describe and construct MPC controllers based on a linear model, quadratic costs and linear constraints
- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples

State predictions

The state model

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

gives the state predictions

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{x}(k+2|k) \\ \vdots \\ \hat{x}(k+N|k) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \hat{x}(k|k) + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N-1|k) \end{bmatrix} \tag{30}$$

Output predictions

The state model

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

gives the output predictions

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \hat{x}(k|k) + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N-1| \end{bmatrix} \tag{31}$$

Output predictions using control moves

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix} = \underbrace{\begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \hat{x}(k|k) + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \begin{bmatrix} I \\ \vdots \\ \vdots \\ I \end{bmatrix} u(k-1)}_{\text{free response}}$$

$$+ \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB + CB & CB & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \sum_{i=0}^{N-1} CA^i B & \sum_{i=0}^{N-2} CA^i B & \cdots & CB \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+N-1|k) \end{bmatrix} \qquad (32)$$

or

$$\boldsymbol{y}(k) = \boldsymbol{y_f}(k) + \Theta \boldsymbol{\Delta u}(k) \qquad (33)$$

State observer

System model:

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

Observer update:

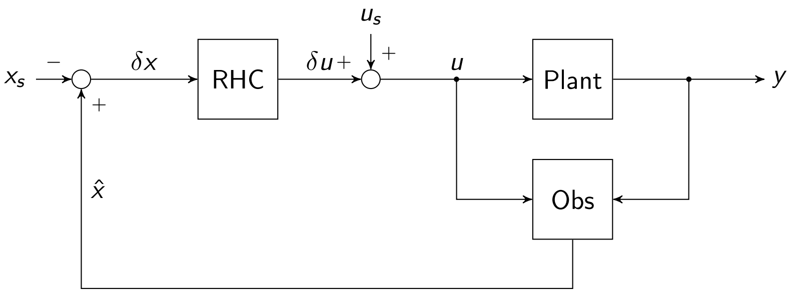
$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + L(y(k) - C\hat{x}(k|k-1))$$

Error dynamics:

$$\tilde{x}(k) = \hat{x}(k) - x(k)$$
$$\tilde{x}(k+1) = (A - LC)\tilde{x}(k)$$

If (A, C) is observable, then the eigenvalues of the error dynamics matrix $(A - LC)$ can be assigned arbitrarily.

MPC overall structure



- RHC = Receding horizon controller, working on deviation variables
- Obs = Observer (state estimator)

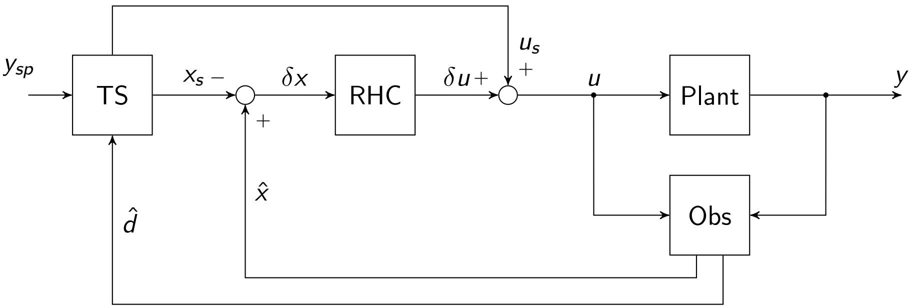
Steady-state target problem ($p = m$)

If the system is square ($p = m$):

1. Define desired setpoints for the outputs, y_{sp}.
2. Solve the following system of linear equations to find the steady-state targets:

$$\begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B_d \hat{d} \\ y_{sp} - C_d \hat{d} \end{bmatrix}, \tag{34}$$

MPC block diagram



- RHC = Receding horizon controller, working on deviation variables
- Obs = Observer (state estimator)
- TS = Target selector

Steady-state target problem ($p > m$)

If there are more outputs than inputs ($p > m$):

1. Define desired setpoints for the outputs, y_{sp}.
2. Solve the following optimization problem to find the best steady-state targets:

$$\min_{x_s, u_s} \left(|Cx_s - y_{sp}|_Q^2 \right), \quad Q \succeq 0$$

subject to

$$\begin{bmatrix} I - A & -B \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = 0$$

$$Eu_s \leq e$$

$$FCx_s \leq f$$

Steady-state target problem ($p_z < m$)

If there are more inputs than controlled outputs ($p_z < m$):

1. Define setpoints for controlled outputs, z_{sp}, and desired values for the control input, u_{sp}, and non-controlled outputs y_{sp}.
2. Solve the following optimization problem to find feasible steady-state targets:

$$\min_{x_s, u_s} \left(|u_s - u_{sp}|^2_{R_s} + |C_y x_s - y_{sp}|^2_{Q_s} \right), \quad R_s \succ 0$$

subject to

$$\begin{bmatrix} I - A & -B \\ C_z & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ z_{sp} \end{bmatrix}$$

$$E u_s \leq e$$

$$F C_z x_s \leq f$$

Steady-state target problem with disturbance

Optimization problem to find feasible steady-state target:

$$\min_{x_s, u_s} \left(|u_s - u_{sp}|^2_{R_s} + |Cx_s + C_d \hat{d} - y_{sp}|^2_{Q_s} \right), \quad R_s \succ 0$$

subject to

$$\begin{bmatrix} I - A & -B \\ HC & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B_d \hat{d} \\ z_{sp} - HC_d \hat{d} \end{bmatrix}$$

$$Eu_s \leq e$$

$$FHCx_s \leq f - FHC_d \hat{d}$$

Off-set free control

Proposition (Off-set free control)

Assume that the steady-state target problem is feasible and that an MPC with the following augmented model is used:

$$\begin{bmatrix} x \\ d \end{bmatrix}^{+} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$
$$y = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} \tag{35}$$

Further assume that $C_z = HC$, $n_d = p$ and that B_d, C_d are chosen such that

$$rank \begin{bmatrix} I - A & -B_d \\ C & C_d \end{bmatrix} = n + p \tag{36}$$

Assume that the closed-loop converges to a steady-state with constraints inactive. Then there is zero off-set in the controlled outputs, i.e.

$$z_s = z_{sp}$$

Lecture 6: The Kalman filter and moving horizon estimation

Goals for today:

- To refresh the Kalman filter
- To formulate a state estimator based on least-squares (LS)
- To formulate a moving horizon estimator based on LS
- To formulate a moving horizon estimator with constraints

Learning objectives:

- Understand and explain the basic principles of model predictive control, its pros and cons, and the challenges met in implementation and applications
- Correctly state, in mathematical form, MPC formulations based on descriptions of control problems expressed in application terms
- Describe and construct MPC controllers based on a linear model, quadratic costs and linear constraints

The Kalman filter

System model:

$$
\begin{aligned}
x(k+1) &= Ax(k) + Bu(k) + w(k), \quad x(0) \sim \mathcal{N}(x_0, P_0), w \sim \mathcal{N}(0, Q) \\
y(k) &= Cx(k) + v(k), \quad v \sim \mathcal{N}(0, R)
\end{aligned}
$$

State estimator:

Correction: $\hat{x}(k|k) = \hat{x}(k|k-1) + L(k)[y(k) - C\hat{x}(k|k-1)],\ \hat{x}(0|0) = x_0$

Prediction: $\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k)$

Kalman filter gain:

$$L(k) = P(k)C^T[CP(k)C^T + R]^{-1}$$

State estimation error covariance update (note that $P(k) \widehat{=} P(k|k-1)$):

Estimation error: $P(k|k) = P(k) - P(k)C^T[CP(k)C^T + R]^{-1}CP(k),\ P(0|0) = P_0$

Prediction error: $P(k+1) = AP(k|k)A^T + Q$

Stationary Kalman filter

System:

$$\begin{aligned} x^+ &= Ax + Bu + w, \quad w \sim \mathcal{N}(0, Q) \\ y &= Cx + v, \quad v \sim \mathcal{N}(0, R) \end{aligned}$$

If the pair (C, A) is observable and $Q, R \succ 0$, then the Kalman filter gain $L(k)$ and the prediction error covariance $P(k)$ converge to the solution of the (filtering) algebraic Riccati equation

$$\begin{aligned} L &= PC^T[CPC^T + R]^{-1} \\ P &= APA^T - APC^T[CPC^T + R]^{-1}CPA^T + Q \end{aligned}$$

Duality of LQR and Kalman filter

LQ regulator:

$$\begin{aligned} u(k) &= K(k)x(k), \quad k = 0, \dots, N-1 \\ K(k) &= -(R + B^T P(k+1)B)^{-1} B^T P(k+1)A \end{aligned}$$

Riccati equation:

$$P(k-1) = Q + A^T P(k)A - A^T P(k)B[R + B^T P(k)B]^{-1} B^T P(k)A, \quad P(N) = P_f$$

Kalman filter:

$$\begin{aligned} \hat{x}(k+1|k) &= A\hat{x}(k|k-1) + Bu(k) + L(k)[y(k) - C\hat{x}(k|k-1)] \\ L(k) &= AP(k)C^T[CP(k)C^T + R]^{-1} \end{aligned}$$

Riccati equation:

$$P(k+1) = AP(k)A^T + Q - AP(k)C^T[CP(k)C^T + R]^{-1}CP(k)A^T, \quad P(0) = P_0$$

Least-squares estimation

System model:

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

Optimization problem:

$$\min_{x(0:T)} V_T(x(0:T))$$

where the minimization is with respect to the sequence of state estimates

$$x(0:T) = \{x(0), x(1), \ldots, x(T)\}$$

The objective function V_T is given by

$$\begin{aligned} V_T(x(0:T)) = {} & (x(0) - x_0)^T P_0^{-1} (x(0) - x_0) \\ & + \sum_{i=0}^{T-1} (x(i+1) - Ax(i) - Bu(i))^T Q^{-1} (x(i+1) - Ax(i) - Bu(i)) \\ & + \sum_{i=0}^{T} (y(i) - Cx(i))^T R^{-1} (y(i) - Cx(i)) \end{aligned} \tag{37}$$

Least squares estimator

System model:

$$x(k+1) = Ax(k) + Bu(k)), \quad y(k) = Cx(k)$$

Objective function:

$$V_T(x(0\colon T)) = (x(0) - x_0)^T P_0^{-1}(x(0) - x_0) + \sum_{i=0}^{T}(y(i) - Cx(i))^T R^{-1}(y(i) - Cx(i))$$
$$+ \sum_{i=0}^{T-1}(x(i+1) - Ax(i) - Bu(i))^T Q^{-1}(x(i+1) - Ax(i) - Bu(i))$$

State estimator:

$$\begin{aligned}
P_t(k) &= AP_m(k-1)A^T + Q, \quad P_t(0) = P_0 \\
L(k) &= P_t(k)C^T[CP_t(k)C^T + R]^{-1} \\
P_m(k) &= P_t(k) - P_t(k)C^T[CP_t(k)C^T + R]^{-1}CP_t(k) \\
\hat{x}(k) &= A\hat{x}(k-1) + Bu(k-1) + L(k)\big(y(k) - C(A\hat{x}(k-1) + Bu(k-1))\big);\ \hat{x}(-1) \\
x^0(k) &= \hat{x}(k) + P_m(k)A^T[AP_m(k)A^T + Q]^{-1}\big(x^0(k+1) - A\hat{x}(k)\big);\ x^0(T) = \hat{x}(T)
\end{aligned} \tag{38}$$

Least squares estimator convergence

Assume that the LS estimator is applied to the system given by

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

If the pair (C, A) is observable and $Q, R \succ 0$, then the LS state estimate converges to the true system state:

$$x^0(T|T) \to x(T), \quad T \to \infty$$

Moving horizon estimation

System model:

$$x(i+1) = Ax(i) + Bu(i)$$
$$y(i) = Cx(i)$$

Optimization problem:

$$\min_{x(k-T:k)} \hat{V}_T(x(k-T:k))$$

where the minimization is with respect to the sequence of state estimates

$$x(k-T:k) = \{x(k-T), x(k-T+1), \dots, x(k)\}$$

The objective function V_T is given by

$$\hat{V}_T(x(k-T:k)) = \sum_{i=k-T}^{k-1} \big(x(i+1) - Ax(i) - Bu(i)\big)^T Q^{-1} \big(x(i+1) - Ax(i) - Bu(i$$
$$+ \sum_{i=k-T}^{k} \big(y(i) - Cx(i)\big)^T R^{-1} \big(y(i) - Cx(i)\big) \qquad (39)$$

Constrained moving horizon estimator

Objective function:

$$\hat{V}_T(x(k-T:k)) = \sum_{i=k-T}^{k-1} \omega(i)^T Q^{-1} \omega(i) + \sum_{i=k-T}^{k} \nu(i)^T R^{-1} \nu(i)$$

Optimization problem:

$$\min \hat{V}_T(x(k-T:k))$$

with respect to

$$\{x(k-T:k), \omega(k-T:k-1), \nu(k-T:k)\}$$

and subject to

$$\begin{gathered} \omega(i) = x(i+1) - (Ax(i) + Bu(i)), \quad x(0) = x_0 \\ \nu(i) = y(i) - Cx(i) \\ x(i) \in \mathbb{X}, \quad \omega(i) \in \mathbb{W}, \quad \nu(i) \in \mathbb{V}, \quad \text{for all } i \in (k-T, k) \end{gathered}$$

Lecture 7: Optimization basics and convexity

Goals for today:

- To formulate a general constrained optimization problem
- To formulate necessary conditions for optimality
- To master the basics of convex sets and convex functions
- To formulate a standard convex optimization problem
- To characterize a standard quadratic program (QP)
- To understand the objectives of Assignment 2

Learning objectives:

- Understand and explain basic properties of the optimization problem as an ingredient of MPC, in particular concepts like linear, quadratic and convex optimization, optimality conditions, and feasibility

Constrained optimization problem

A basic optimization problem is formulated as

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \tag{40}$$

$x = \{x_1, \dots, x_n\}$ are the optimization or decision variables

$f : \mathbb{R}^n \to \mathbb{R}$ is the objective or cost function

$g_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \dots, m$ are inequality constraint functions

$h_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \dots, p$ are equality constraint functions

The *optimal solution* x^0 has the smallest value of $f(\cdot)$ among all vectors x that belong to dom f (the *domain* of f, i.e. the subset of $\mathbb{R}^n$ where f is defined) and satisfy the constraints. The *optimal value* p^0 is always defined:

$$p^0 = \inf\{f(x) \mid g_i(x) \leq 0,\ i = 1, \dots, m;\ h_j(x) = 0,\ j = 1, \dots, p\}$$

$$p^0 = \infty, \quad \text{if problem is infeasible}$$

$$p^0 = -\infty, \quad \text{if problem is unbounded below}$$

Conditions for local optimality – unconstrained case

- **First order necessary condition**

$$x^* \text{ is a local optimum} \quad \Rightarrow \quad \nabla f(x^*) = 0 \tag{41}$$

- **Second order sufficient conditions**

$$\nabla f(x^*) = 0 \text{ and } \nabla^2 f(x^*) \succ 0 \quad \Rightarrow \quad x^* \text{ is a strict local minimum} \tag{42}$$

First order necessary conditions – equality constraints

Consider the optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & h(x) = 0 \end{aligned}$$

Assume x^* is a local minimum and that x^* is regular.

Then there is a unique vector λ^* such that

$$\begin{aligned} \nabla f(x^*) + \nabla h(x^*)\lambda^* &= 0 \\ h(x^*) &= 0 \end{aligned}$$

This is a system of non-linear equations having $n + p$ equations for the $n + p$ unknowns (x and λ).
The vector λ contains the *Lagrange multipliers* λ_i, $i = 1, \ldots, p$.

First order necessary conditions – the KKT conditions

Consider the optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \tag{43}$$

Assume x^* is a local minimum and that x^* is regular.

Then there are unique vectors μ^* and λ^* such that

$$\nabla f(x^*) + \nabla g(x^*)\mu^* + \nabla h(x^*)\lambda^* = 0 \tag{44}$$

$$\mu^* \geq 0 \tag{45}$$

$$g(x^*) \leq 0, \quad h(x^*) = 0 \tag{46}$$

$$\mu_i^* g_i(x^*) = 0, \quad i = 1, \ldots, m \tag{47}$$

These conditions are referred to as the *KKT (Karush-Kuhn-Tucker) conditions*.

Second order sufficient conditions

Consider the optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \tag{48}$$

Assume x^* is regular and that x^*, μ^*, λ^* satisfy the KKT conditions with all active constraints being strictly active. Further assume that

$$d^T \nabla_x^2 \mathcal{L}(x^*, \mu^*, \lambda^*) d > 0, \quad \text{for all } d \text{ such that } d^T \begin{bmatrix} \nabla g_{\mathbb{A}} & \nabla h \end{bmatrix} = 0,$$

where $\nabla_x^2 \mathcal{L}$ is the Hessian of the Lagrangian.

Then x^* is a local minimum.

Convex optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_i(x) = 0, \quad i = 1, \dots, p \end{aligned}$$

where the objective function f and constraint functions $\{g_i\}$ are convex, i.e.

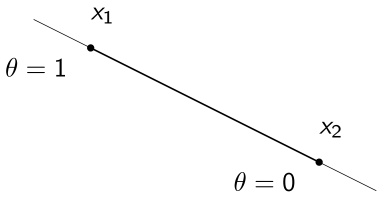
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y), \quad 0 \leq \theta \leq 1$$

and the functions $\{h_i\}$ are *affine* (linear).

Affine sets

Line through x_1 and x_2 are all points x,

$$x = \theta x_1 + (1-\theta)x_2, \quad \theta \in \mathbb{R}$$



Affine set contains the line through any two distinct points in the set.

Example: solution set of linear equations $\{x \mid Ax = b\}$.
(all affine sets can be described as solutions to a system of linear equations)

Convex sets

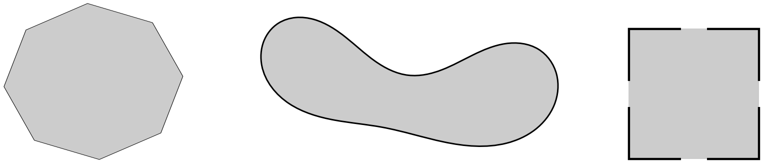
Line segment between x_1 and x_2 are all points x,

$$x = \theta x_1 + (1-\theta)x_2, \quad 0 \leq \theta \leq 1$$

Convex set contains line segment between every two points in the set

$$x_1, x_2 \in \mathcal{S} \;\Rightarrow\; \theta x_1 + (1-\theta)x_2 \in \mathcal{S}, \quad 0 \leq \theta \leq 1$$

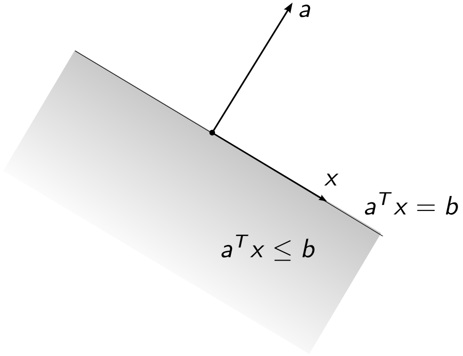
Examples (one convex and two non-convex sets):



Hyperplanes and halfspaces

Hyperplane is a set of the form $\{x \mid a^T x = b\}$

Halfspace is a set of the form $\{x \mid a^T x \leq b\}$



Hyperplanes are affine and convex; halfspaces are convex.

Operations that preserve convexity

- The *intersection* of convex sets is convex.
- If f is *affine* ($f(x) = Ax + b$), then the image of a convex set under f is convex:

$$\mathcal{S} \text{ convex } \Rightarrow f(\mathcal{S}) \text{ convex}$$

- If f is affine, then the inverse image of f is convex:

$$\mathcal{S} \text{ convex } \Rightarrow f^{-1}(\mathcal{S}) = \{x \mid f(x) \in \mathcal{S}\} \text{ convex}$$

Examples: Scaling, translation, projection

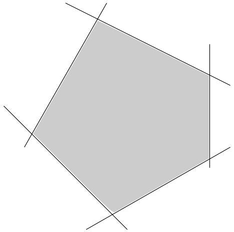
Polyhedra

A *polyhedron* is the intersection of a finite number of halfspaces and hyperplanes or, equivalently, the solution set of a finite number of linear inequalities and equalities:

$$\begin{aligned} Ax &\leq b \\ Cx &= d \end{aligned}$$

Polyhedra are convex sets.

Example:



Convex functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if $\operatorname{dom} f$ is convex and

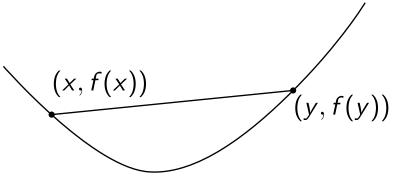
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

for all $x, y \in \operatorname{dom} f$ and $0 \leq \theta \leq 1$.

- f is *concave* of $-f$ is convex
- f is *strictly convex* if

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y)$$

  for all $x, y \in \operatorname{dom} f$ and $0 < \theta < 1$



Examples of convex and concave functions

Convex functions:

- affine: $a^T x + b$
- exponential: e^{ax}
- powers: $x^\alpha, \quad x > 0$, for $\alpha \geq 1$ or $\alpha \leq 0$

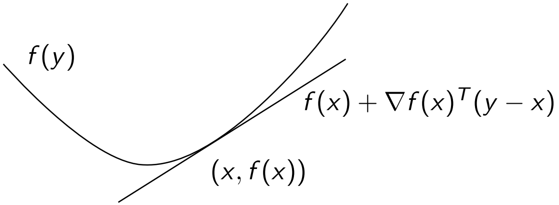
Concave functions:

- affine: $a^T x + b$
- logarithm: $\log x,\ x > 0$
- powers: $x^\alpha, \quad x > 0$, for $0 \leq \alpha \leq 1$

First and second order conditions

- Differentiable f with convex domain is convex if and only if

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y \in \operatorname{dom} f$$



- Twice differentiable f with convex domain is convex if and only if

$$\nabla^2 f(x) \geq 0 \quad \text{for all } x \in \operatorname{dom} f$$

Operations preserving convexity

- Sub-level sets $\mathcal{S}_\alpha$ of a convex function f are convex

$$\mathcal{S}_\alpha = \{x \in \operatorname{dom} f \mid f(x) \leq \alpha\}$$

- Nonnegative weighted sum of convex functions is convex:

$$f_1, \ldots, f_N \text{ convex} \quad \Rightarrow \sum_{i=1}^{N} \alpha_i f_i \text{ convex, for all } \alpha_i \geq 0$$

- The composition with an affine function is convex:

$$f \text{ convex} \quad \Rightarrow \quad f(Ax+b) \text{ convex}$$

Convex optimization problem

Standard form convex optimization problem:

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \dots, m \\ & Ax = b, \quad \text{(affine equality constraints)} \end{array}$$

where f and $\{g_i\}$ are convex.

Note: The *feasible set* of a convex optimization problem is convex (why is this so?).

Optimality conditions for convex problems

Consider the convex optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \tag{49}$$

where f and $\{g_i\}$ are convex and h is affine.

Assume x^* is regular. Then x^* is globally optimal if and only if the KKT conditions are fulfilled for some $\mu^* \geq 0$, λ^*.

Examples of convex optimization problems

Linear programming (LP):

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

Quadratic programming (QP):

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} x^T Q x + p^T x, \quad Q \succeq 0 \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

In both cases, the feasible set is a polyhedron.

The Lagrangian

Consider the standard form problem (not necessarily convex)

$$\begin{array}{rll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array}$$

where $x \in \mathcal{D} \subseteq \mathbb{R}^n$.

Lagrangian $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, with $\operatorname{dom} L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$,

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{i=1}^{p} \lambda_i h_i(x) \,\widehat{=}\, f(x) + \mu^T g(x) + \lambda^T h(x)$$

where

- μ_i is Lagrange multiplier associated with the constraint $g_i(x) \leq 0$
- λ_i is Lagrange multiplier associated with the constraint $h_i(x) = 0$

Note that for $\mu \geq 0$ and any feasible x, we have $\mathcal{L}(x, \mu, \lambda) \leq f(x)$.

Lagrange dual function

Lagrange dual function $q : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$,

$$q(\mu, \lambda) = \inf_{x \in \mathcal{D}} \mathcal{L}(x, \mu, \lambda) = \inf_{x \in \mathcal{D}} \left\{ f(x) + \mu^T g(x) + \lambda^T h(x) \right\}$$

Properties:

- q is concave but may be $-\infty$ for some μ, λ
- $q(\mu, \lambda) \leq p^0$ if $\mu \geq 0$ (p^0 is the optimal value of the original problem)

The dual problem

The *Lagrange dual problem*:

$$\begin{array}{rl} \text{maximize} & q(\mu, \lambda) \\ \text{subject to} & \mu \geq 0 \end{array}$$

- Finds best lower bound on p^0 from Lagrange dual function
- Always a convex, unconstrained problem; denote optimal value d^0
- μ, λ are *dual feasible* if $\mu \geq 0$ and $(\mu, \lambda) \in \operatorname{dom} q$
- We always have $d^0 \leq p^0$ (*weak duality*)

Weak and strong duality

Weak duality: $d^0 \leq p^0$

- always holds (even for non-convex problems)
- gives lower bound for the original (*primal*) problem

Strong duality: $d^0 = p^0$

- does not hold in general
- often holds for convex problems
- conditions that guarantee this are called *constraint qualifications*

Constraint qualifications

Strong duality holds for a convex problem

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

if any of the following conditions are fulfilled:

1. The gradients of the equality constraints and the active inequality constraints are linearly independent (LICQ)
2. The problem is strictly feasible, i.e. there exists some $\tilde{x} \in \operatorname{int} \mathcal{D}$ (the interior of $\mathcal{D}$) such that (Slater CQ)

$$g_i(\tilde{x}) < 0, \quad i = 1, \ldots, m; \quad A\tilde{x} = b$$

Convex problems with strong duality

Consider the standard convex optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{aligned}$$

where f and $\{g_i\}$ are convex.

Assume x^* is regular. Then the following statements are equivalent:

- x^* is a global optimum
- there are μ^*, λ^* such that the KKT conditions hold

Lecture 8 : Solving QP problems

Goals for today:

- To formulate Newton's method to solve the KKT conditions in simple cases
- To understand the principles of active set and interior point methods for QP:s

Learning objectives:

- Understand and explain basic properties of the optimization problem as an ingredient of MPC, in particular concepts like linear, quadratic and convex optimization, optimality conditions, and feasibility

Quadratic programming (QP)

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2}x^T Qx + p^T x, \quad Q \succ 0 && (50) \\ \text{subject to} \quad & Ax = b, \quad A \in \mathbb{R}^{p \times n} && (51) \end{aligned}$$

The Lagrangian:

$$\mathcal{L}(x, \lambda) = \frac{1}{2}x^T Qx + p^T x + \lambda^T (Ax - b)$$

Newton's method:

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} Qx + p \\ Ax - b \end{bmatrix} \quad \Leftrightarrow$$

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^+ \\ \lambda^+ \end{bmatrix} = \begin{bmatrix} -p \\ b \end{bmatrix} \qquad (52)$$

Solutions to QP special cases

Unconstrained case:

$$\text{minimize} \quad f(x) = \frac{1}{2}x^T Qx + p^T x, \quad Q \succ 0$$

Solution:

$$Qx^0 + p = 0$$

QP with equality constraint:

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2}x^T Qx + p^T x, \quad Q \succ 0 \\ \text{subject to} \quad & Ax = b \end{aligned}$$

Solution:

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^0 \\ \lambda^0 \end{bmatrix} = \begin{bmatrix} -p \\ b \end{bmatrix}$$

QP with inequality constraints

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T Q x + p^T x, \quad Q \succeq 0 \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned} \tag{53}$$

For easy reference, we repeat the KKT conditions (44)-(47) for this special case:

$$Qx^* + p + G^T\mu^* + A^T\lambda^* = 0 \tag{54}$$

$$\mu^* \geq 0 \tag{55}$$

$$Gx^* - h \leq 0, \quad Ax^* - b = 0 \tag{56}$$

$$\mu_i^*(g_i^T x^* - h_i) = 0, \quad i = 1, \ldots, m \tag{57}$$

Logarithmic barrier formulation

The QP problem

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2}x^T Q x + p^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

can be approximated by the following problem:

$$\begin{aligned} \text{minimize} \quad & f_\tau(x) = f(x) - \tau \sum_{i=1}^{m} \log(h_i - g_i^T x) \quad (\tau > 0) \\ \text{subject to} \quad & Ax = b \end{aligned}$$

where g_i^T is the ith row of G, h_i is the ith element of h.

The *convex* function

$$\phi_\tau(x) = -\tau \sum_{i=1}^{m} \log(h_i - g_i^T x)$$

is called the *logarithmic barrier* for the original QP problem.

A primal-dual interior point method

The QP problem

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2}x^T Q x + p^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

is characterized by the approximated (smoothed) KKT conditions

$$\begin{gathered} Qx + p + G^T\mu + A^T\lambda = 0 \\ Ax - b = 0 \\ Gx - h + s = 0 \\ \mu_i s_i = \tau \\ s > 0, \quad \mu > 0 \end{gathered}$$

Applying Newton's method on the equalities gives the Newton step

$$\begin{bmatrix} Q & A^T & G^T & 0 \\ A & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & \operatorname{diag}(s) & \operatorname{diag}(\mu) \end{bmatrix} \begin{bmatrix} x^+ \\ \lambda^+ \\ \mu^+ \\ s^+ \end{bmatrix} = \begin{bmatrix} -p \\ b \\ h \\ \operatorname{diag}(s)\mu + \tau \end{bmatrix}$$

Backtracking to secure $s > 0$ and $\mu > 0$ is simple!

Lecture 9 : MPC practice – implementation and tuning

Goals for today:

- To understand the issue of recursive feasibility and how to obtain it
- To understand the role of constraint management and back-up strategies
- To know some measures to improve computational efficiency
- To have an overview of important user choices for MPC

Learning objectives:

- Understand and explain the basic principles of model predictive control, its pros and cons, and the challenges met in implementation and applications
- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples
- Understand and explain basic properties of the optimization problem as an ingredient of MPC, in particular concepts like linear, quadratic and convex optimization, optimality conditions, and feasibility

Recursive feasibility – definition

Definition (Invariant set)

The set $\mathcal{S}$ is *positively invariant* for the autonomous system $x^+ = f_a(x)$ if

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

Definition (Recursive feasibility)

The receding horizon controller is recursively feasible if the feasible set $\mathcal{X}_N$ is positively invariant for the closed-loop system $x^+ = f(x, \kappa_N(x))$, i.e.

$$x(0) \in \mathcal{X}_N \quad \Rightarrow \quad x(k+1) = f(x(k), \kappa_N(x(k))) \in \mathcal{X}_N, \quad \forall k \in \mathbb{N}$$

The implication of these definitions is that if the RHC is recursively feasible, then it is guaranteed that the optimization problem to be solved at every time instant is feasible, provided the initial state is feasible.

Condition for recursive feasibility

Definition (Control invariant set)

A set $\mathcal{C} \subseteq \mathbb{X}$ is a *control invariant set* of the system $x^+ = f(x, u)$ if

$$x \in \mathcal{C} \quad \Rightarrow \quad \exists u \in \mathbb{U} \text{ such that } x^+ = f(x, u) \in \mathcal{C}$$

The *maximal control invariant set* contained in $\mathbb{X}$ is denoted $\mathcal{C}_\infty$ and contains all control invariant sets in $\mathbb{X}$.

Theorem (Sufficient condition for recursive feasibility)

The receding horizon controller based on the finite horizon optimal control problem (23)-(26) is recursively feasible if the terminal constraint set $\mathbb{X}_f$ is control invariant.

Constraint management

- Soft constraints can be introduced:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & V_N(\boldsymbol{u}) + \rho\|\varepsilon\|^2 && (58) \\ \text{subject to} \quad & F\boldsymbol{u} + G\boldsymbol{x} \leq e + \varepsilon && (59) \\ & \varepsilon \geq 0 && (60) \end{aligned}$$

- Reduce window in which constraints are enforced
- Prioritise constraints $\rightarrow$ mixed-integer quadratic program
- Don't forget time limitations – control output *must* be delivered!

Feasibility – summary

- It may be impossible, for some initial states, to solve the RHC optimization problem while respecting constraints on states and/or outputs. The problem is in these cases *infeasible*.
- In the nominal case, without disturbances and with a perfect model, *recursive feasibility* may be insured by choosing the terminal constraint set $\mathbb{X}_f$ in a proper way.
- In practice, infeasibility may still occur, possibly caused by too strict performance requirements, a large disturbance, model uncertainty, or by an unstable system.
- Ad hoc solutions are for example to keep the control as is (from the previous sampling instant), or to switch in a backup controller.
- A more systematic approach is to relax the constraints in some way; this is often referred to as *constraint management*.

Computational efficiency measures

- Problem size depends on M and N
- Representations and ordering of variables
- *Blocking*, i.e. requiring a piecewise constant u, gives fewer decision variables
- Using the previous result as initial guess may speed up the optimization
- Tailor-made code is typically much faster than Matlab

Design or tuning parameters

- Cost function weights
- Prediction and control horizons
- Constraints
- Disturbance models
- Observer dynamics
- Reference trajectories
- Steady state target calculations

Weights and horizons

- Cost function weights
  - which states and outputs are most important? which are the targets/setpoints?
  - penalty within system delay is not meaningful (parameter H_w)
  - penalty on control or control moves?
  - only relations between Q and R important
- Horizons
  - important that predicted trajectories resemble what is obtained in closed-loop
  - general rule: choose M/N as large as possible (trade-off between stability and performance vs. computations)
  - rule-of-thumb: $N \approx$ settling time and M should cover transient (usually small).
- Some special cases:
  - 'Mean-level control': $M = 1$, N large, $R = 0$ and r constant.
    One control action, steady-state important; slow response (roughly as open-loop).
  - 'Dead-beat control': $M = H_w = n$, $N \geq 2n$, $R = 0$, r constant.
    Enough control actions to settle the states before the output is observed.
  - 'Myopic control': $M = N = 1$, $R = 0$.
    Resembles inverting the plant dynamics (minimum phase!)

Constraints, disturbances and references

- Constraints
  - constraints on u and/or Δu, compatible with physical constraints OK!
  - tighter constraints on u or Δu may be used to achieve smoother control (warning: unstable plant)
  - constraints on states and/or outputs is at the core of MPC, but too strict constraints may be impossible to fulfil!
- Disturbance models and observers
  - standard: step disturbances and off-set free control
  - stochastic disturbances: modelling $\rightarrow$ estimator/observer design
  - rule-of-thumb: observer dynamics should be faster than closed-loop settling time
- Reference trajectories and steady-state targets
  - piecewise constant setpoints most common
  - targets/setpoints are important, but some outputs may have no targets
  - more careful design to avoid hitting constraints ('reference governor')
  - look-ahead of references/set-points sometimes possible!

Lecture 10: Stability

Goals for today:

- To understand and be able to use Lyapunov functions to study simple stability problems
- To understand the fundamental ingredients in establishing stability for receding horizon controllers
- To formulate and verify the stability conditions of constrained linear quadratic MPC

Learning objectives:

- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples

Stability definitions

Consider the system

$$x^+ = f(x), \quad f(0) = 0 \tag{61}$$

Definition (Local stability)

The origin is *locally stable* for the system (61) if, for any $\epsilon > 0$, there exists a $\delta > 0$ such that $|x(0)| < \delta$ implies $|x(k)| < \epsilon$ for all $k \geq 0$.

Definition (Global attraction)

The origin is *globally attractive* for the system (61) if $x(k) \to 0,\ k \to \infty$ for any initial $x(0)$.

Definition (Global asymptotic stability)

The origin is *globally asymptotically stable* for the system (61) if it is locally stable and globally attractive.

Remark. If the origin is attractive only for initial states within a certain set, the definitions can be modified and the properties then hold with a *region of attraction* that is not any longer the entire $\mathbb{R}^n$.

Lyapunov function

Definition (Lyapunov function)

A function $V : \mathbb{R}^n \to \mathbb{R}_+$ is said to be a Lyapunov function for the system (61) if there are functions $\alpha_i \in \mathcal{K}_\infty, i = 1, 2$ and a positive definite function α_3 such that for any $x \in \mathbb{R}^n$,

$$\begin{aligned} V(x) &\geq \alpha_1(|x|) \\ V(x) &\leq \alpha_2(|x|) \\ V(f(x)) - V(x) &\leq -\alpha_3(|x|) \end{aligned}$$

Remark. A function belongs to $\mathcal{K}_\infty$ if it is nonnegative, continuous, zero at zero, strictly increasing and unbounded. A positive definite function is continuous and positive everywhere except at the origin.

Lyapunov's theorem

Proposition (Lyapunov's theorem)

Suppose $V(\cdot)$ is a Lyapunov function for the system

$$x^+ = f(x), \quad f(0) = 0$$

Then the origin is globally asymptotically stable.

Basic MPC equations

Cost function:

$$V_N(x_0, u(0:N-1)) = \sum_{i=0}^{N-1} l(x(i), u(i)) + V_f(x(N)) \tag{62}$$

Optimal cost-to-go:

$$V_N^0(x_0) = \min_{u(0:N-1)} \{V_N(x_0, u(0:N-1)) \mid u(0:N-1) \in \mathcal{U}_N(x_0)\}$$

Constraints:

$$x^+ = f(x,u), \quad x(0) = x_0 \tag{63}$$

$$x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad \text{for all } k \in (0, N) \tag{64}$$

$$x(N) \in \mathbb{X}_f \subseteq \mathbb{X} \tag{65}$$

Feasible control sequences and initial states:

$$u(0:N-1) \in \mathcal{U}_N(x_0) \tag{66}$$

$$\mathcal{X}_N = \{x_0 \in \mathbb{X} \mid \mathcal{U}_N(x_0) \neq \emptyset\} \tag{67}$$

Optimal control and state sequences:

$$\begin{aligned} u^0(0:N-1; x_0) &= \{u^0(0; x_0), u^0(1; x_0), \dots, u^0(N-1; x_0)\} \\ x^0(0:N; x_0) &= \{x^0(0; x_0), x^0(1; x_0), \dots, x^0(N; x_0)\} \end{aligned} \tag{68}$$

Existence of solution

Proposition (Existence of solution)

With the assumptions above, the following holds:

(a) *The function V_N is continuous on $\mathcal{X}_N \times \mathcal{U}_N$.*

(b) *For each $x \in \mathcal{X}_N$, the control constraint set $\mathcal{U}_N(x)$ is closed and bounded.*

(c) *For each $x \in \mathcal{X}_N$, a solution to the optimal control problem exists.*

Remark. Note that nothing is said about the properties of the *value function* $V_N^0(x)$ or the control law $\kappa_N(x) = u^0(0; x)$. In fact, these may both be discontinuous. However, if there are no state constraints (i.e. $\mathbb{X} = \mathbb{X}_f = \mathbb{R}^n$) or if the system is linear and the constraint sets are all polyhedral, then the value function is continuous.

Basic stability assumption

We proceed with the following basic assumption:

Assumption (Basic stability assumption)

The terminal set $\mathbb{X}_f$ is control invariant and the following inequality holds:

$$\min_{u \in \mathbb{U}} \{V_f(f(x,u)) + l(x,u) \mid f(x,u) \in \mathbb{X}_f\} \leq V_f(x), \quad \forall x \in \mathbb{X}_f$$

Remark. Recall that control invariance of $\mathbb{X}_f$ means that there exists a $u \in \mathbb{U}$ such that $f(x,u) \in \mathbb{X}_f$, see Definition 9.3. The implication of this, together with the properties of $\mathbb{U}$, is that the minimum in the assumption above exists.

MPC stability

Theorem (MPC stability)

Assume that Assumption 10.3 holds and that the stage cost $l(\cdot)$ and the terminal cost $V_f(\cdot)$ satisfy

$$\begin{aligned} l(x,u) &\geq \alpha_1(|x|) \quad \forall x \in \mathcal{X}_N, u \in \mathbb{U} \\ V_f(x) &\leq \alpha_2(|x|) \quad \forall x \in \mathbb{X}_f \end{aligned}$$

with $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$. Further, assume that $\mathbb{X}_f$ contains the origin in its interior. Then the origin is asymptotically stable with a region of attraction $\mathcal{X}_N$ for the system $x^+ = f(x, \kappa_N(x))$.

Stability of constrained linear quadratic MPC

Theorem (Stability of constrained linear quadratic MPC)

Consider the linear quadratic MPC with linear constraints applied to the controllable system $x^+ = Ax + Bu$ and with positive definite matrices Q and R. Further assume that the terminal cost V_f is chosen as the value function of the corresponding unconstrained, infinite horizon LQ controller, and that the terminal constraint set $\mathbb{X}_f$ is chosen as described above. Then the origin is asymptotically stable with a region of attraction $\mathcal{X}_N$ for the controlled system $x^+ = Ax + B\kappa_N(x)$.

Lecture 11: Alternative formulations of MPC

Goals for today:

- To get an introductory knowledge about MPC based on step response and transfer function models
- To understand how MPC can be extended to incorporate feedforward from measured disturbances
- To understand concepts like stabilized predictions, zones, funnels, and coincidence points
- To get some insight from an application of MPC to vehicle control

Learning objectives:

- Understand and explain the basic principles of model predictive control, its pros and cons, and the challenges met in implementation and applications
- Correctly state, in mathematical form, MPC formulations based on descriptions of control problems expressed in application terms

Advantages of using state space models in MPC

- Generality – linear and nonlinear
- Often result from modelling from first principles
- Linearization and discretization is straightforward
- Computational delays can be incorporated
- State observers can be designed
- Numerical computations

Step response models for MPC

+ Simple to understand
+ Easy to get from step response experiments
- Emphasis on low frequency behaviour
- Constrained to stable plants
- Inefficient model representation (e.g. first order lag with large T)

Refer to the discussion about myths on step response models in Maciejowski's book!

Impulse response models

An LTI input-output model is given by

$$y(t) = \sum_{k=0}^{t} H(t-k)u(k), \tag{69}$$

where $\{H(0), H(1), \ldots\}$ is the (matrix valued) impulse response. The step response is obtained as

$$S(t) = \sum_{k=0}^{t} H(k)$$

If the step response (or the impulse response) is truncated at N with $S(N+1) \approx S(N)$, the remaining sequence is a model of the system:

- Dynamic matrix: $\{S(0), S(1), \ldots S(N)\}$ (used in DMC)
- FIR model: $\{H(0), H(1), \ldots H(N)\}$

Relations to state space models

The state space model (with $D = 0$)

$$\begin{aligned} x^+ &= Ax + Bu \\ y &= Cx \end{aligned}$$

has an impulse response given by the *Markov parameters* $\{CA^iB\}$, giving the step response

$$S(k) = \sum_{i=0}^{k-1} CA^iB$$

Using this in (32) reveals the close connections to the state space formulation:

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \hat{x}(k|k) + \begin{bmatrix} S(1) \\ S(2) \\ \vdots \\ S(N) \end{bmatrix} u(k-1) + \begin{bmatrix} S(1) & 0 & \cdots & 0 \\ S(2) & S(1) & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ S(N) & S(N-1) & \cdots & S(1) \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+N-1|k) \end{bmatrix}$$

$$\Leftrightarrow$$

$$\boldsymbol{y}(k) = \boldsymbol{y_f}(k) + \Theta \boldsymbol{\Delta u}(k)$$

State space models from step responses

From the relation (assuming $D = 0$)

$$H(k) = CA^{k-1}B$$

the following *Hankel matrix* can be formed:

$$\begin{bmatrix} H(1) & H(2) & H(3) & \cdots \\ H(2) & H(3) & H(4) & \cdots \\ H(3) & H(4) & H(5) & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} CB & CAB & \cdots \\ CAB & CA^2B & \cdots \\ CA^2B & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} B & AB & A^2B & \cdots \end{bmatrix}$$

By collecting step response data, the matrix above can be factorized for a suitable order (dimension of the Hankel matrix), and a state space representation (A, B, C) can be obtained from the factors.

Measured disturbances and feedforward

The state space model can easily be extended with measurable disturbances:

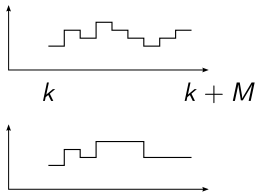
$$x(k+1) = Ax(k) + Bu(k) + B_d d_m(k)$$

- Include the disturbance in the state prediction using $B_d \hat{d}_m(k+i|k)$.
- The control signal computed will try to compensate for the anticipated effect of the disturbance ("look-ahead").

Re-parameterization of the control signal

- Reduction of complexity:

  Limit the number of optimization variables by keeping M low or by using *blocking*, i.e. keeping the control signal constant over several sampling intervals.



- The use of *stabilized predictions* can improve numerics:

$$u(k+i|k) = -K\hat{x}(k+i|k) + \tilde{u}(k+i|k),$$

  implying that the predictor expressions will contain $(A-BK)^i$ instead of A^i.
- Parameterize the control signal using *basis functions*:

$$u(k+i|k) = \sum_{j=1}^{n_u} c_j u_j(i),$$

  where $\{u_j(\cdot)\}$ are basis functions (e.g. polynomials). Once the coefficients $\{c_j\}$ have been determined (in the optimization step of the MPC), the control signal is defined over the whole prediction horizon. This is called *Predictive Functional Control (PFC)*.

Zones, funnels, and coincidence points

As an input to the optimization in the MPC, the desired behavior of the system within the prediction horizon needs to be expressed. There are many ways to do this — the following variants are discussed further in [**?**]:

- A *reference trajectory* $r(\cdot)$ for the output(s) is defined, starting at the current output and ending at the setpoint $s(\cdot)$. Typically, the objective contains a term $(y - r)^T Q(y - r)$.
- If a setpoint is not really relevant, then it is possible to instead define a *zone objective*. In this case, an objective as above is not meaningful, and instead constraints are given for upper and lower bounds of the output(s).
- Combining the two ideas above, it is possible to define constraints that become stricter towards the end of the prediction horizon, thus forming a *funnel* in which the states or outputs should reside.

Lecture 12: Explicit control laws for constrained, linear systems

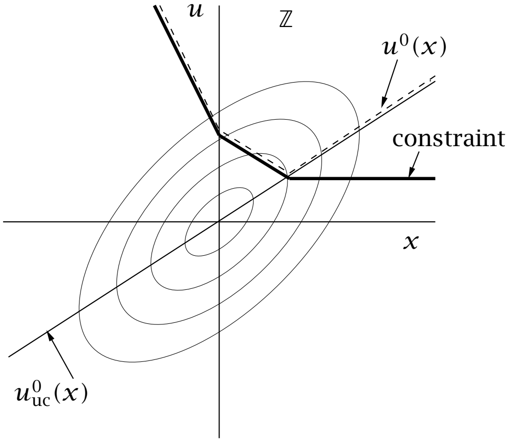
Goals for today:

- To understand how the concept of parametric programming can be applied to RHC
- To understand the principles behind so called explicit MPC

Learning objectives:

- Describe and construct MPC controllers based on a linear model, quadratic costs and linear constraints
- Describe basic properties of MPC controllers and analyze algorithmic details on very simple examples

Parametric quadratic program (from [1])



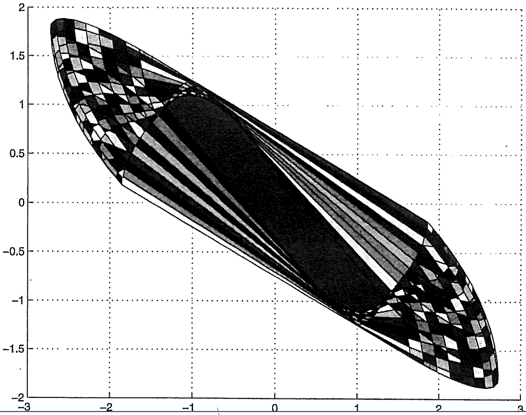
Explicit MPC strategy

1. Find a partition $\{R_{x_i}\}$ of $\mathcal{X}$.
2. Calculate the optimal, affine control law on each of the regions R_{x_i} and store the result, i.e. controller parameters. This and the previous step can be done *off-line*.
3. During *on-line* operation, run the controller by
   - for the measured state x, identify the region to which x belongs
   - look up the controller parameters for the found region
   - compute the next control signal

Explicit MPC example

Example (Second order system)

It was mentioned above that the number of regions R_i may become very large. This is indicated in the figure below, which shows the regions for a simple second order system.



Lecture 13: Beyond linear MPC

Goals for today:

- To understand some of the ideas used to extend MPC to the nonlinear case
- To understand what is meant by robust MPC

Learning objectives:

- Understand and explain the basic principles of model predictive control, its pros and cons, and the challenges met in implementation and applications

Linear time-invariant MPC

At every time instant k, solve the constrained optimization problem

$$\begin{aligned}
\underset{\boldsymbol{\delta u}_k, \boldsymbol{\delta x}_k}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \frac{1}{2} \begin{bmatrix} \delta x_k(i) \\ \delta u_k(i) \end{bmatrix}^T W \begin{bmatrix} \delta x_k(i) \\ \delta u_k(i) \end{bmatrix} && \text{(70a)} \\
\text{subject to} \quad & \delta x_k(0) = \hat{x}(k) - x_k^r(0) && \text{(70b)} \\
& \delta x_k(i+1) = A\,\delta x_k(i) + B\,\delta u_k(i); \quad i = 0, \ldots, N-1 && \text{(70c)} \\
& F\,\delta u_k(i) + G\,\delta x_k(i) \leq h; \quad i = 0, \ldots, N-1 && \text{(70d)}
\end{aligned}$$

The solution is written as

$$(\boldsymbol{\delta u}_k, \boldsymbol{\delta x}_k) = \mathrm{QP}_{\mathrm{MPC}}(\hat{x}(k), \boldsymbol{u}_k^{\boldsymbol{r}}, \boldsymbol{x}_k^{\boldsymbol{r}}) \qquad \text{(70e)}$$

The control output is given by

$$u(k) = u_k^r(0) + \delta u_k(0) \qquad \text{(70f)}$$

Linear time-varying MPC

At every time instant k, solve the constrained optimization problem

$$
\begin{aligned}
\underset{\boldsymbol{\delta u}_k, \boldsymbol{\delta x}_k}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \frac{1}{2} \begin{bmatrix} \delta x_k(i) \\ \delta u_k(i) \end{bmatrix}^T W_{k,i} \begin{bmatrix} \delta x_k(i) \\ \delta u_k(i) \end{bmatrix} && (71a) \\
\text{subject to} \quad & \delta x_k(0) = \hat{x}(k) - x_k^r(0) && (71b) \\
& \delta x_k(i+1) = A_{k,i}\, \delta x_k(i) + B_{k,i}\, \delta u_k(i) + r_k(i); \quad i = 0, \dots, N-1 && (71c) \\
& F_{k,i}\, \delta u_k(i) + G_{k,i}\, \delta x_k(i) \le h_{k,i}; \quad i = 0, \dots, N-1 && (71d)
\end{aligned}
$$

The solution is written as

$$(\boldsymbol{\delta u}_k, \boldsymbol{\delta x}_k) = \mathrm{QP}_{\mathrm{MPC}}(\hat{x}(k), \boldsymbol{u}_k^{\boldsymbol{r}}, \boldsymbol{x}_k^{\boldsymbol{r}}) \tag{71e}$$

The control output is given by

$$u(k) = u_k^r(0) + \delta u_k(0) \tag{71f}$$

Nonlinear MPC

At every time instant k, solve the constrained optimization problem

$$\begin{aligned}
\underset{\boldsymbol{u}_k,\boldsymbol{x}_k}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \frac{1}{2} \begin{bmatrix} x_k(i) - x_k^r(i) \\ u_k(i) - u_k^r(i) \end{bmatrix}^T W_{k,i} \begin{bmatrix} x_k(i) - x_k^r(i) \\ u_k(i) - u_k^r(i) \end{bmatrix} && \text{(72a)} \\
\text{subject to} \quad & x_k(0) = \hat{x}(k) && \text{(72b)} \\
& x_k(i+1) = f(x_k(i), u_k(i)); \quad i = 0, \ldots, N-1 && \text{(72c)} \\
& h(x_k(i), u_k(i)) \leq 0; \quad i = 0, \ldots, N-1 && \text{(72d)}
\end{aligned}$$

The solution is written as

$$(\boldsymbol{u}_k, \boldsymbol{x}_k) = \text{NLP}(\hat{x}(k), \boldsymbol{u}_k^{\boldsymbol{r}}, \boldsymbol{x}_k^{\boldsymbol{r}}) \qquad \text{(72e)}$$

The control output is given by

$$u(k) = u_k(0) \qquad \text{(72f)}$$

SQP for NMPC

At every time instant k, the SQP optimization variables $(\boldsymbol{u}_k, \boldsymbol{x}_k)$ are initialized as

$$(\boldsymbol{u}_k, \boldsymbol{x}_k) = (\boldsymbol{u}_k^{\text{guess}}, \boldsymbol{x}_k^{\text{guess}}) \tag{73a}$$

Then the following QP is solved repeatedly at the current SQP iterate $(\boldsymbol{u}_k, \boldsymbol{x}_k)$ for the Newton correction $(\Delta\boldsymbol{u}_k, \Delta\boldsymbol{x}_k)$, which gives the next iterate by taking a (reduced) Newton step $(\boldsymbol{u}_k, \boldsymbol{x}_k) \leftarrow (\boldsymbol{u}_k, \boldsymbol{x}_k) + t(\Delta\boldsymbol{u}_k, \Delta\boldsymbol{x}_k)$:

$$\underset{\Delta\boldsymbol{u}_k, \Delta\boldsymbol{x}_k}{\text{minimize}} \quad \sum_{i=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta x_k(i) \\ \Delta u_k(i) \end{bmatrix}^T H_{k,i} \begin{bmatrix} \Delta x_k(i) \\ \Delta u_k(i) \end{bmatrix} + J_{k,i}^T \begin{bmatrix} \Delta x_k(i) \\ \Delta u_k(i) \end{bmatrix} \tag{73b}$$

$$\text{subject to} \quad \Delta x_k(0) = \hat{x}(k) - x_k(0) \tag{73c}$$

$$\Delta x_k(i+1) = A_{k,i}\,\Delta x_k(i) + B_{k,i}\,\Delta u_k(i) + r_k(i); \tag{73d}$$

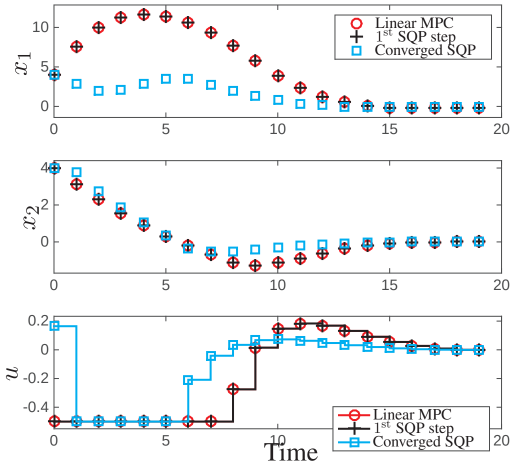
$$F_{k,i}\,\Delta u_k(i) + G_{k,i}\,\Delta x_k(i) + h_{k,i} \leq 0; \quad i = 0, \ldots, N-1 \tag{73e}$$

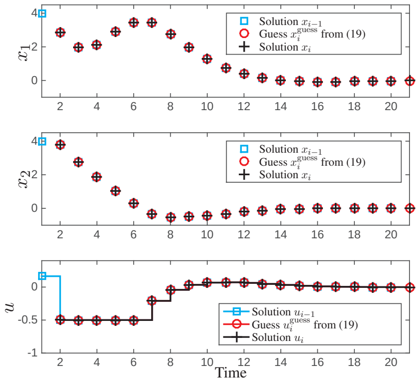
The solution of the SQP is written as

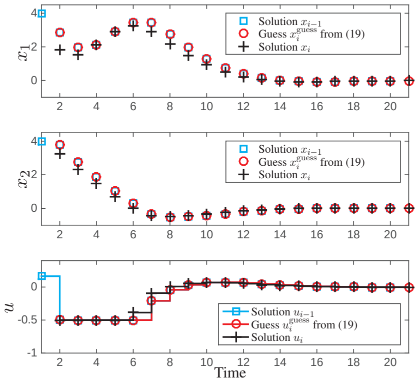
$$(\boldsymbol{u}_k, \boldsymbol{x}_k) = \text{SQP}(\hat{x}(k), \boldsymbol{u}_k^{\text{guess}}, \boldsymbol{x}_k^{\text{guess}}, \boldsymbol{u}_k^{\boldsymbol{r}}, \boldsymbol{x}_k^{\boldsymbol{r}}) \tag{73f}$$

The control output is given by

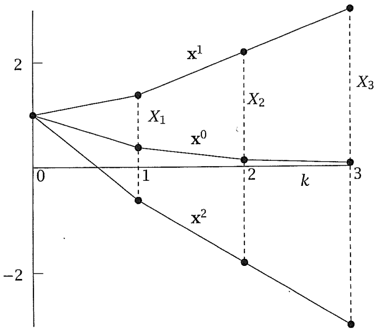
$$u(k) = u_k(0) \tag{73g}$$


Linear MPC
1st SQP step
Converged SQP
x_1
x_2
u
Time
0
5
10
15
20
-0.4
-0.2
0.2
4
2

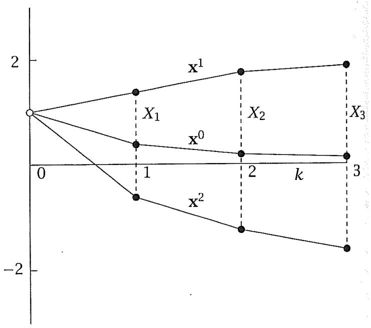

x_1
x_2
u
Time
Solution x_{i-1}
Guess x_i^{guess} from (19)
Solution x_i
Solution u_{i-1}
Guess u_i^{guess} from (19)
Solution u_i


x_1
x_2
u
Time
Solution x_{i-1}
Guess x_i^{guess} from (19)
Solution x_i
Solution u_{i-1}
Guess u_i^{guess} from (19)
Solution u_i

■



(a) Open-loop trajectories.



(b) Feedback trajectories.