Lecture 13:
Model predictive control

Mikael Johansson
School of Electrical Engineering
KTH, Stockholm, Sweden
Learning aims

After this lecture you should be able to

• express finite-horizon constrained LQR problems as quadratic programs
• explain the basic idea of model predictive control
  – apply constrained optimal control in receding-horizon fashion
• enforce integral action in an MPC controller
• explain the issue of infeasibility and know how to circumvent it
• limit computational requirements of MPC by limiting control horizon
Last lecture: finite-horizon LQR

Find control sequence

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

that minimizes the quadratic cost function

$$J(U) = \sum_{k=0}^{N-1} (x_k^T Q_1 x_k + u_k^T Q_2 u_k) + x_N^T Q_f x_N$$

for given state cost, control cost, and final cost matrices

$$Q = Q^T \geq 0, \quad R = R^T > 0, \quad Q_f = Q_f^T \geq 0$$

N is called the **horizon** of the problem. Note the final state cost.

Optimal solution via quadratic minimization or dynamic programming.
Last lecture: finite-horizon LQR

Dynamic programming solution

1. set $P_N = Q_f$

2. for $t = N, N - 1, \ldots, 1$
   
   $$P_{t-1} := Q_1 + A^T P_t A - A^T P_t B (Q_2 + B^T P_t B)^{-1} B^T P_t A$$

3. for $t = 0, 1, \ldots, N - 1$
   
   $$L_t := (Q_2 + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$$
   $$u_t^* = -L_t x_t$$

Note: optimal control is a linear function of the state
Example

Same system as earlier. Investigate how elements of P and L converge

Rapid convergence to stationarity as t drops below horizon N!
Consider the cost function

\[ J_k(u_k, \ldots, u_{k+K-1}) = \sum_{t=k}^{k+K-1} (x_t^T Q_1 x_t + u_t^T Q_2 u_t) + x_{k+K}^T Q_f x_{k+K} \]

Here, K is called the \textbf{horizon}, and if

\[(u_k^*, \ldots, u_{k+K-1}^*)\]

minimizes \(J_k\), then \(u_k^*\) is called \textbf{K-step optimal receding horizon control}

**Receding-horizon control:**
- at time \(k\), find input sequence that minimizes K-step ahead LQR cost (starting at time \(k\))
- then apply only the first element of the input sequence
Last lecture: receding horizon LQR

Two ways to ensure closed-loop stability:

1. Use terminal cost matrix \( Q_f = P \) where
   \[
P = Q_1 + A^T P A - A^T P B (Q_2 + B^T P B)^{-1} B^T P A
   \]
   (i.e. \( P \) solves the discrete-time ARE) ensures stability.

   Why? Receding horizon-control is then (independent of \( t \))
   \[
u_t = -L x_t \quad L = (Q_2 + B^T P B)^{-1} B^T P A
   \]
   and the associated closed-loop system is stable
   (if basic observability and controllability conditions are met)

2. Use longer horizon, so that control approaches stationary optimal
Today: constrained predictive control

Finite-horizon LQG with hard constraints on $u$ and $y$:

\[
\begin{align*}
\text{minimize} & \quad J(U) = \sum_{k=0}^{N-1} (x_k^T Q_1 x_k + u_k^T Q_2 u_k) + x_N^T Q_f x_N \\
\text{subject to} & \quad u_{\text{min}} \leq u_k \leq u_{\text{max}}, \quad k = 0, \ldots, N - 1 \\
& \quad y_{\text{min}} \leq C x_k \leq y_{\text{max}}, \quad k = 1, \ldots, N \\
& \quad x_{k+1} = A x_k + B u_k
\end{align*}
\]

Can be simplified by eliminating $\{x_1, \ldots, x_N\}$ (as in last lecture)

- results in a quadratic programming problem in $\{u_0, \ldots, u_{N-1}\}$
Quadratic programming (QP)

Minimizing a quadratic objective function subject to linear constraints

\[
\begin{align*}
\text{minimize} & \quad u^T P u + 2q^T u + r \\
\text{subject to} & \quad A u \leq b
\end{align*}
\]

Any \(u\) satisfying \(A u \leq b\) is said to be \textbf{feasible}.
- clearly, not all quadratic programs are feasible
  (depends on \(A, b\); more about this later...)

“Easy” to solve when objective function is \textbf{convex} (\(P\) positive semidefinite)
- optimal solution found in polynomial time
- commercial solvers deal with 1000’s of variables in a few seconds
Quadratic programming tricks

Example. The double inequality \( u_{\text{min}} \leq u \leq u_{\text{max}} \) can be written as

\[
\begin{bmatrix} 1 \\ -1 \end{bmatrix} u \leq \begin{bmatrix} u_{\text{max}} \\ -u_{\text{min}} \end{bmatrix}
\]

Example. The equality \( u = u_{\text{tgt}} \) can be written as \( u_{\text{tgt}} \leq u \leq u_{\text{tgt}} \), hence

\[
\begin{bmatrix} 1 \\ -1 \end{bmatrix} u \leq \begin{bmatrix} u_{\text{tgt}} \\ -u_{\text{tgt}} \end{bmatrix}
\]
Constrained control via QP

\[
\begin{align*}
\text{minimize} \quad & J(U) = \sum_{k=0}^{N-1} (x_k^T Q_1 x_k + u_k^T Q_2 u_k) + x_N^T Q_f x_N \\
\text{subject to} \quad & u_{\text{min}} \leq u_k \leq u_{\text{max}}, \quad k = 0, \ldots, N-1 \\
& y_{\text{min}} \leq C x_k \leq y_{\text{max}}, \quad k = 1, \ldots, N \\
& x_{k+1} = A x_k + B u_k
\end{align*}
\]

As in last lecture, introducing \( X = (x_0, \ldots, x_N) \), \( U = (u_0, \ldots, u_{N-1}) \),

\[
X = GU + H x_0
\]

and the objective function can be written as

\[
J(U) = U^T P_{LQ} U + 2 q_{LQ}^T U + r_{LQ}
\]

Convex since \( Q_2 > 0 \) (see last lecture for precise expressions)

What about the constraints?
Predictive control with constraints

Similarly, the constraints \( y_{\text{min}} \leq y_k \leq y_{\text{max}}, \ k = 0, \ldots, N \) can be written as

\[
Y \geq y_{\text{min}} 1, \quad Y \leq y_{\text{max}} 1
\]

where \( Y = (y_0, \ldots, y_N) \). Introducing

\[
\bar{C} = \begin{bmatrix}
C & 0 & \ldots & 0 \\
0 & C & 0 & 0 \\
\vdots & \ddots & \ddots & 0 \\
0 & \ldots & 0 & C
\end{bmatrix}
\]

we can re-write these inequalities in terms of \( U \) via

\[
Y \geq y_{\text{min}} 1 \iff \bar{C}(GU + Hx_0) \geq y_{\text{min}} 1 \iff \overbrace{CGU}^{A_Y} \geq \underbrace{y_{\text{min}} 1 - CHx_0}_{b_Y}
\]

\[
Y \leq y_{\text{max}} 1 \iff \bar{C}(GU + Hx_0) \leq y_{\text{max}} 1 \iff \underbrace{CGU}^{A_Y} \leq \underbrace{y_{\text{max}} 1 - CHx_0}_{b_Y}
\]
Predictive control with constraints

Hence, the constrained predictive control problem can be cast as a QP

\[
\text{minimize} \quad U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ}
\]

subject to

\[
\begin{bmatrix}
  A_Y \\
  -A_Y \\
  I \\
  -I
\end{bmatrix} U \leq \begin{bmatrix}
  b_Y \\
  -b_Y \\
  u_{\max} 1 \\
  -u_{\min} 1
\end{bmatrix}
\]

Solution gives optimal finite-horizon control subject to constraints

Model predictive control:
- apply constrained optimal control in receding horizon fashion
Model predictive control algorithm

1. Given state at time $t$ compute ("predict") future states
   
   $x_{t+k}, \quad k = 0, 1, \ldots, N$
   
   as function of future control inputs
   
   $u_{t+k}, \quad k = 0, 1, \ldots, N - 1$

2. Find "optimal" input by minimizing constrained cost function
   
   - a quadratic program, efficiently solved

3. Implement $u(t)$

4. A next sample $(t+1)$, return to 1.
MPC in pictures
Example: the DC servo

Discrete-time model (sampling time 0.05 sec)

\[ A = \begin{bmatrix} 1 & 0.139 \\ 0 & 0.861 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0107 \\ 0.139 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \]

Constrained input voltage

\[ -1 \leq u \leq 1 \]

Constrained position

\[ y_{\text{min}} \leq y_k \leq y_{\text{max}} \]
Impact of state and control weights

Prediction horizon $N=10$. 

\begin{figure}[h]
\centering
\includegraphics[width=\linewidth]{figures}
\end{figure}
Impact of horizon

Too short horizon $\Rightarrow$ inaccurate predictions $\Rightarrow$ poor performance
Reference tracking

Would like $z$ to track a reference sequence $\{r_1, \ldots, r_N\}$, i.e. to keep

$$
\sum_{k=0}^{N-1} ( (z_k - r_k)^T Q_1 (z_k - r_k) + u_k^T Q_2 u_k ) + (z_N - r_N)^T Q_f (z_N - r_N)
$$

small.

Problem: making $z_k = r_k$ typically requires $u_k \neq 0$

- a trade-off between zero tracking errors and using zero control
- often results in steady-state tracking error
Including integral action

Integral action often included by a change in free variables
- Use $\Delta u_i = u_i - u_{i-1}$ as variables in the optimization
- Actual input obtained by summing up MPC outputs

$$\{\Delta u_k, \ldots, \Delta u_{k+N-1}\}$$

$$\sum \Delta u_k$$

$$u_k = u_{k-1} + \Delta u_k$$

Extended model for MPC computations
Including integral action cont’d

Form augmented model with state

\[ \bar{x}_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \]

and input \( \Delta u_k \):

\[
\begin{bmatrix}
  x_{k+1} \\
  u_k
\end{bmatrix} =
\begin{bmatrix}
  A & B \\
  0 & I
\end{bmatrix}
\begin{bmatrix}
  x_k \\
  u_{k-1}
\end{bmatrix}
+ 
\begin{bmatrix}
  0 \\
  I
\end{bmatrix}
\Delta u_k
\]

\[ y_k = \begin{bmatrix} C & 0 \end{bmatrix}
\begin{bmatrix}
  x_k \\
  u_{k-1}
\end{bmatrix} \]

Consider finite-horizon cost

\[
\sum_{k=0}^{N-1} ((z_k - r_k)^T Q_1 (z_k - r_k) + \Delta u_k^T Q_2 \Delta u_k) + (z_N - r_N)^T Q_f (z_N - r_N)
\]

(easy to add penalty also to \( u_k \)...)

EL2520 Control Theory and Practice

Mikael Johansson mikaelj@ee.kth.se
Tracking with output constraints
Infeasibility

What happens when there is no solution to the QP?

Not clear what control to apply!
Ensuring feasibility

One way to ensure feasibility:
- introduce slack variables \( s_{ck} \geq 0 \)
- “soften” constraints
  \[ u_k \leq u_{\text{max}} \Rightarrow u_k \leq u_{\text{max}} + s_{ck} \]
- add term in quadratic programming objective to minimize slacks

\[
\begin{align*}
\text{minimize} & \quad U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ} \\
\downarrow
\quad \text{minimize} & \quad U^T P_{LQ} U + 2q_{LQ}^T U + r_{LQ} + \kappa S^T S
\end{align*}
\]

Notes:
- still QP, but more variables; can also use penalty \( \kappa S \) (also QP)
- better to soften already soft constraints (e.g. output constraints)
Limited horizon control

The longer horizon, the more variables in the QP.

Idea: limit the number of variables by
   - predicting (and constraining) the system over horizon of length $N$
   - only optimizing the first $N_u \leq N$ control actions
(called input and output horizons, or control and prediction horizons)

Solutions depend on what we assume about $u$ beyond control horizon.
Limited horizon control I

Simple solution: control is held constant beyond control horizon

\[ \{u_0, \ldots, u_{N_u-1}\} \text{ are optimized, } u_{N_u} = \cdots = u_{N-1} = u_{N_u-1} \]

Can use the same QP formulation as before and add linear equalities

\[ u_k = u_{N_u-1} \quad k = N_u, N_{u+1}, \ldots, N \]

Can re-write as a QP on standard form (using techniques form earlier)

Modern solvers automatically remove these variables from QP.
Limited horizon control II

Alternative approach: use inner-loop feedback

\[ u_k = -Lx_k, \quad k = N_u, N_u+1, \ldots, N \]

Prediction model becomes

\[ x_{k+1} = Ax_k + Bu_k \quad k = 0, 1, \ldots, N_u - 1 \]
\[ x_{N_u+t} = (A - BL)^t x_{N_u} \quad t = 0, 1, \ldots, N - N_u - 1 \]

With \( X = (x_0, \ldots, x_N) \), \( \bar{U} = (u_0, \ldots, u_{N_u-1}) \), we can write

\[ X = GU + Hx_0 \]

So finite-horizon constrained optimal control can be found via QP.

**Note:** beyond control horizon, \( u_{\text{min}} \leq u_k \leq u_{\text{max}} \Leftrightarrow u_{\text{min}} \leq -Lx_k \leq u_{\text{max}} \) (constraints on control translates into constraints on predicted states)
MPC controller tuning

MPC has a large number of “tuning” parameters.

The prediction model:
- we need to decide sampling interval
  (rule of thumb: sample 10 times desired closed-loop bandwidth)
- obtain discrete-time state-space model

Finite-horizon optimal control:
- set prediction horizon
  (rule of thumb: equal to closed-loop rise time; could be smaller)
- decide weight matrices (as for continuous-time LQG)
- decide final state penalty
  (guideline: stationary Riccati solution for given weight matrices)
MPC controller tuning

Finite-horizon optimal control, advanced:
- control horizon (try to set small, rule-of-thumb: use 1-10)
- inner-loop control
  (guideline: stationary LQR controller for given weight matrices)

Constraints and feasibility
- specify control and state constraints (problem dependent)
- introduce slacks to “soften” constraints
- choose constraint penalty (large value on kappa)

Integral action (almost always a good idea to include).
Advanced issues: stability

Receding horizon control might yield unstable closed-loop

Stability can be guaranteed:
- for infinite-horizon unconstrained case (this is LQR)
- for finite-horizon unconstrained case
  - if final state is penalized correctly
  - if final state is enforced to lie in a given set
- for constrained finite-horizon
  - if final state enforced to lie in a sufficiently small set and
  - initial QP (solved at time zero) is feasible

Hard to verify for sure in advance...
Advanced issues: robustness

Consider the unconstrained quadratic program

\[
\text{minimize } \quad u^T Q u + 2q^T u
\]

has optimal solution \( u = -Q^{-1} q \)

In the MPC setting, \( Q \) and \( q \) depend on the system model (matrices \( A, B, C \)), weights \( Q_1, Q_2 \), and also horizons.

Solution is sensitive to uncertainties if \( Q \) is ill-conditioned

- Try scaling inputs and outputs in the model
- Modify weight matrices \( Q_1 \) and \( Q_2 \)
- Almost always a good idea to include integral action
Advanced issues: observers

MPC, as presented here, assumes full state feedback.

In many cases, we will need to use an observer,
  – to reconstruct states, and/or
  – to filter out noise

Limited theory, but separation principle holds in some cases.

Suggests guideline
  – design observer as for (unconstrained, infinite-horizon) LQG
  – use estimated state in MPC calculations as if it was true state
Get a feel for MPC!

Files for setting up and simulating a problem is on home page.

Do Computer Exercise 5 (download from home page)!
Summary

Model predictive control (MPC)
- Can handle state and control constraints
- Predictive control computed via quadratic programming

Many parameters and their influence on the control
- System model, weights, horizons, constraints, ...

Advanced issues:
- The need for a state observer
- Different prediction and control horizons
- Feasibility and slacks to “soften” constraints
- Stability and the terminal weight
- Integral action

Do computer exercise 5 to get a feel for MPC!