

Module 06 — Introduction to Model Predictive Control

Ahmad F. Taha

EE 5243: Introduction to Cyber-Physical Systems

Email: ahmad.taha@utsa.edu

Webpage: <http://engineering.utsa.edu/~taha/index.html>

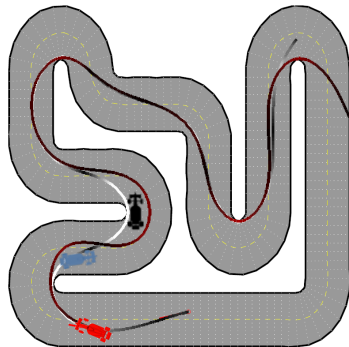


October 5, 2015

Introduction to MPC — Example¹

What is Model-Predictive Control?

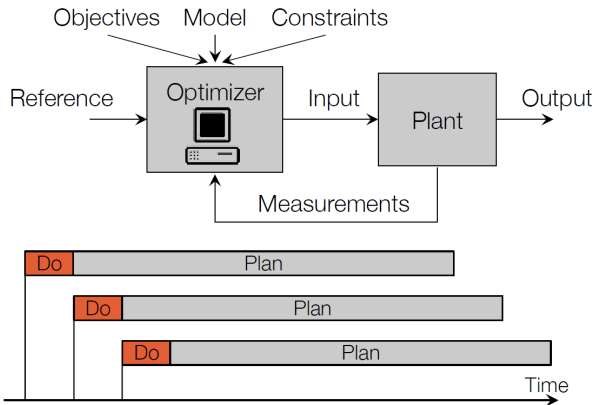
- Compute first control action (for a prediction horizon)
- Apply first control action
- Repeat given updated constraints
- Essentially, solving optimization problems sequentially
- Use static-optimization techniques for optimal control problems
- **Example:** minimizing LapTime, while NotKillingPeople
- **MPC** \equiv Receding Horizon Control



¹Some figures are borrowed from the references; see the end of the presentation file.

MPC Schematic

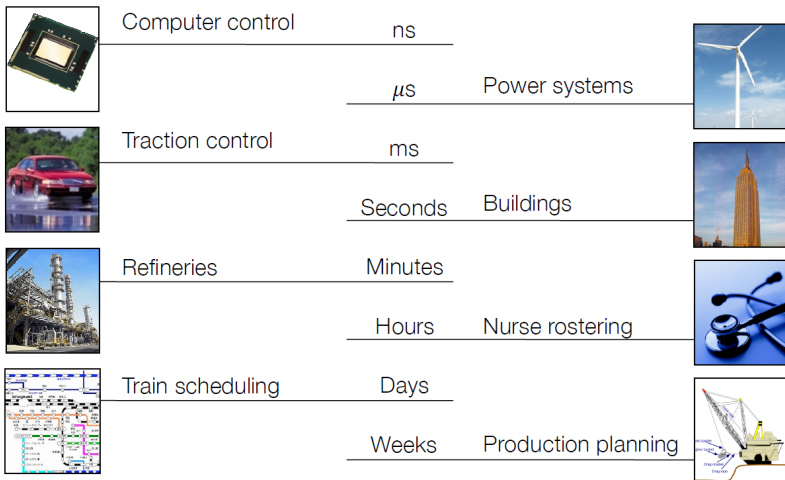
MPC leverages constrained static-optimization for optimal control problems



MPC: real-time, sequential optimization with constraints on states and inputs²

²Some figures are borrowed from the references; see the end of the presentation file.

MPC Applications + Time Horizons

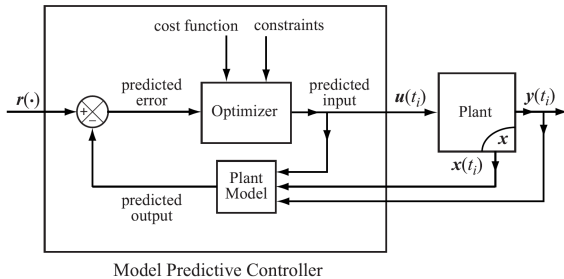
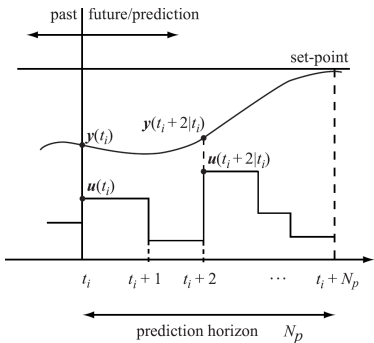


MPC Constraints

- We saw in this module that simple OCPs can be very hard to analyze and solve
- OCPs become harder with constraints on states and controls
- Most physical systems have constraints
 - ① Safety limits (minimum and maximum capacities)
 - ② Actuator limits
 - ③ Overshoot constraints
- MPC provides a great alternative to solving constrained OCPs, in comparison to HJB, PMP

More on MPC

- 1 At each instant, an MPC uses: current inputs, outputs, states
- 2 Using these signals, MPC computes (over a prediction horizon), a future optimal control sequence
- 3 Solved online³ (explicit MPC, EMPC, is solved offline)



³Figures are borrowed from the references; see the end of the presentation file.

Discrete LMPC Formulation

Linear MPC Problem

$$\begin{aligned} & \underset{U_t}{\text{minimize}} && \sum_{k=0}^{N_p-1} J(x_{t+k}, u_{t+k}) \\ & \text{subject to} && x(t+k+1) = Ax(t+k) + Bu(t+k) \\ & && u \in \mathcal{U} \\ & && x \in \mathcal{X} \\ & && U_t = \{u_t, \dots, u_{t+N_p-1}\} \\ & && x(t) = x_t \text{ (fixed)} \end{aligned}$$

- At each time-instant:

- 1 Measure or estimate $x(t)$
- 2 Find optimal input sequence the PredictionHorizon (N_p)

$$U_t^* = \{u_t, \dots, u_{t+N_p-1}^*\}$$

- 3 Implement **first control action**, u_t^*

Linear Discrete-Time MPC

Objective is to apply MPC for this LTI DT system:

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m, y \in \mathbb{R}^p\end{aligned}$$

- Define $\Delta x(k+1) = x(k+1) - x(k) = A\Delta x(k) + B\Delta u(k)$
- $\Delta y(k+1) = y(k+1) - y(k) = C\Delta x(k+1) = CA\Delta x(k) + CB\Delta u(k)$
- Hence: $y(k+1) = y(k) + CA\Delta x(k) + CB\Delta u(k)$
- Combining the boxed equations, we get:

$$\underbrace{\begin{bmatrix} \Delta x(k+1) \\ y(k+1) \end{bmatrix}}_{x_a(k+1)} = \underbrace{\begin{bmatrix} A & 0 \\ CA & I_p \end{bmatrix}}_{\Phi_a} \underbrace{\begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix}}_{x_a(k)} + \underbrace{\begin{bmatrix} B \\ CB \end{bmatrix}}_{\Gamma_a} \Delta u(k) \quad (1)$$

$$y(k) = \underbrace{\begin{bmatrix} O & I_p \end{bmatrix}}_{C_a} \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix} \quad (2)$$

MPC Problem Construction

$$\begin{aligned}x_a(k+1) &= \Phi_a x_a(k) + \Gamma_a \Delta u(k) \\y(k) &= C_a x_a(k), \quad x_a \in \mathbb{R}^{n+p}, \Gamma_a \in \mathbb{R}^{n+p \times m}, C_a \in \mathbb{R}^{p \times n+p}\end{aligned}$$

- Assume $u(k)$ and $x(k)$ are available, we can get $x(k+1)$
- Hence, x_a is known at k
- **Control objective:** construct control sequence

$$\Delta u(k), \Delta u(k+1), \dots, \Delta u(k+N_p-1), \quad N_p = \text{PredictionHorizon}$$

- This sequence will give us the predicted state vectors

$$\{x_a(k+1|k), \dots, x_a(k+N_p|k)\} \Rightarrow \{y(k+1|k), \dots, y(k+N_p|k)\}$$

MPC Construction

- How can we construct $u(k)$ given $x(k)$? Seems like a least-square problem
- We can write the **predicted future state variables** as:

$$\begin{aligned}
 x_a(k+1|k) &= \Phi_a x_a(k) + \Gamma_a \Delta u(k) \\
 x_a(k+2|k) &= \Phi_a x_a(k+1|k) + \Gamma_a \Delta u(k+1) = \Phi_a^2 x_a(k) + \Phi_a \Gamma_a \Delta u(k) + \Gamma_a \Delta u(k+1) \\
 &\dots = \dots \\
 x_a(k+N_p|k) &= \Phi_a^{N_p} x_a(k) + \Phi_a^{N_p-1} \Gamma_a \Delta u(k) + \dots + \Gamma_a \Delta u(k+N_p-1)
 \end{aligned}$$

- Also, we can write the predicted outputs as:

$$\underbrace{C_a \begin{bmatrix} x_a(k+1|k) \\ x_a(k+2|k) \\ \vdots \\ x_a(k+N_p|k) \end{bmatrix}}_Y = \underbrace{C_a \begin{bmatrix} \Phi_a \\ \Phi_a^2 \\ \vdots \\ \Phi_a^{N_p} \end{bmatrix}}_W x_a(k) + \underbrace{C_a \begin{bmatrix} \Gamma_a & & & \\ \Phi_a \Gamma_a & \Gamma_a & & \\ \vdots & & \ddots & \\ \Phi_a^{N_p-1} \Gamma_a & \dots & \Phi_a \Gamma_a & \Gamma_a \end{bmatrix}}_Z \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix}}_{\Delta U}$$

- Hence, we obtain:

$$Y = \left[y^\top(k+1|k) \quad y^\top(k+2|k) \quad \dots \quad y^\top(k+N_p|k) \right]^\top = W x_a(k) + Z \Delta U$$

- Note: all variables written in terms of current state and future control

Optimal MPC Construction

$$Y = [y^\top(k+1|k) \quad y^\top(k+2|k) \quad \dots \quad y^\top(k+N_p|k)]^\top = Wx_a(k) + Z\Delta U$$

- Y, W, Z, x_a all given \Rightarrow **determine** ΔU (or $\Delta u(k), \dots, \Delta u(k+N_p-1)$)
- Assume that we want to minimize this cost function:

$$J(\Delta U) = \frac{1}{2}(r - Y)^\top Q(r - Y) + \frac{1}{2}\Delta U^\top R\Delta U, \quad Q = Q^\top \succ 0, R = R^\top \succ 0$$

- Cost function = *min deviations from output set-points + control actions*
- This is an unconstrained optimization problem \Rightarrow it's easy to find ΔU^*
- Setting $\frac{\partial J}{\partial \Delta U} = 0 \Rightarrow \Delta U^* = (R + Z^\top QZ)^{-1} Z^\top Q(r - Wx_a)$
- Note that SONC are satisfied as $\frac{\partial^2 J}{\partial \Delta U^2} = R + Z^\top QZ \succ 0$

Optimal MPC Construction — 2

- Now, we need to compute $\Delta u(k)$ (recall $\Delta U, \Delta u(k)$):

$$\begin{aligned}\Delta u(k) &= \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} \Delta U \\ &= \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} (R + Z^\top QZ)^{-1} Z^\top Q(r - Wx_a)\end{aligned}$$

- Above equation can be written as:

$$\begin{aligned}\Delta u(k) &= K_r r - K_r W x_a(k), \text{ where:} \\ K_r &= \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} (R + Z^\top QZ)^{-1} Z^\top Q\end{aligned}$$

- Recall that $x_a(k) = \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix} \Rightarrow$ above equation can be written as:

$$\begin{aligned}\Delta u(k) &= K_r r - K_{mpc} \Delta x(k) - K_y y(k) \\ \Delta u(k) &= \underbrace{K_r r - K_y y(k)}_{\text{reference signals}} - \underbrace{K_{mpc} \Delta x(k)}_{\text{state-feedback gain}}, \text{ where:}\end{aligned}$$

$$K_r = \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} (R + Z^\top QZ)^{-1} Z^\top Q$$

$$K_{mpc} = K_r W \begin{bmatrix} I_n \\ O \end{bmatrix}, \quad K_y = K_r W \begin{bmatrix} O \\ I_p \end{bmatrix}$$

Solving Unconstrained MPC Problems, An Algorithm

- 1 Given CT LTI system, discretize your system (on MATLAB: c2d)
- 2 Specify your prediction horizon N_p
- 3 Find augmented dynamics:

$$\begin{aligned}x_a(k+1) &= \Phi_a x_a(k) + \Gamma_a \Delta u(k) \\ y(k) &= C_a x_a(k)\end{aligned}$$

- 4 Compute W, Z and formulate predicted output equation:

$$Y = W x_a(k) + Z \Delta U$$

- 5 Assign reference signals and weights on control action—formulate $J(\Delta U)$
- 6 Compute optimal control ΔU , extract $\Delta u(k)$ and $u(k)$

LMPC Example

- Consider this LTI, DT dynamical system, give by:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}, C = [1 \quad 0], N_p = 10$$

- Apply the algorithm:

- Augmented dynamics:

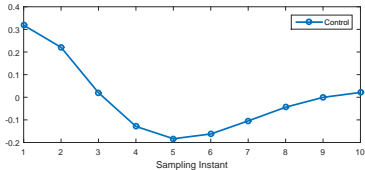
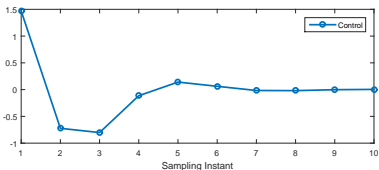
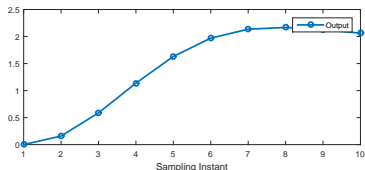
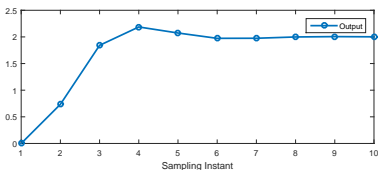
$$\Phi_a = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \Gamma_a = \begin{bmatrix} 0.5 \\ 1 \\ 0 \end{bmatrix}, C_a = [0 \quad 0 \quad 1] \Rightarrow$$

- Find Z, W :

$$Z = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4.5 & 2 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 4.5 & 2 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12.5 & 8 & 4.5 & 2 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 18 & 12.5 & 8 & 4.5 & 2 & 0.5 & 0 & 0 & 0 & 0 \\ 24.5 & 18 & 12.5 & 8 & 4.5 & 2 & 0.5 & 0 & 0 & 0 \\ 32 & 24.5 & 18 & 12.5 & 8 & 4.5 & 2 & 0.5 & 0 & 0 \\ 40.5 & 32 & 24.5 & 18 & 12.5 & 8 & 4.5 & 2 & 0.5 & 0 \\ 50 & 40.5 & 32 & 24.5 & 18 & 12.5 & 8 & 4.5 & 2 & 0.5 \end{bmatrix}, W = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 3 & 6 & 1 \\ 4 & 10 & 1 \\ 5 & 15 & 1 \\ 6 & 21 & 1 \\ 7 & 28 & 1 \\ 8 & 36 & 1 \\ 9 & 45 & 1 \\ 10 & 55 & 1 \end{bmatrix}$$

Example

- Select an output reference signal ($r = 2$) and weight on control ($R = 0.1I$)
- Solve for the optimal ΔU and extract $\Delta u(k), u(k)$
- Apply the first control and generate states and dynamics
- Plots show optimal control with $R = 0.1I$ (left) and $R = 10I$ (right)
- Putting more weight on control action is reflected in the left figure



MPC With Constraints on $\Delta u(k)$

- Previously, we assumed no constraints on states or control
- What if the rate of change of the control, $\Delta u(k)$, is bounded?
- **Solution:** if $\Delta u^{\min} \leq \Delta u(k) \leq \Delta u^{\max}$, then:

$$\begin{bmatrix} -I_m \\ I_m \end{bmatrix} \Delta u(k) \leq \begin{bmatrix} -\Delta u^{\min} \\ \Delta u^{\max} \end{bmatrix}$$

- For a prediction horizon N_p , we have:

$$\begin{bmatrix} -I_m & O & \dots & O & O \\ I_m & O & \dots & O & O \\ O & -I_m & \dots & O & O \\ O & I_m & \dots & O & O \\ \vdots & & & & \vdots \\ O & O & \dots & O & -I_m \\ O & O & \dots & O & I_m \end{bmatrix} \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix}}_{\Delta U} \leq \begin{bmatrix} -\Delta u^{\min} \\ \Delta u^{\max} \\ -\Delta u^{\min} \\ \Delta u^{\max} \\ \vdots \\ -\Delta u^{\min} \\ \Delta u^{\max} \end{bmatrix}$$

MPC With Constraints on $u(k)$

- What if the control, $u(k)$, is bounded?
- **Solution:** We know that:

$$u(k) = u(k-1) + \Delta u(k) = u(k-1) + \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} \Delta U(k)$$

- Similarly:

$$u(k+1) = u(k) + \Delta u(k+1) = u(k-1) + \begin{bmatrix} I_m & I_m & O & \dots & O \end{bmatrix} \Delta U(k)$$

- Or:

$$\begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_p-1) \end{bmatrix} = \begin{bmatrix} I_m \\ I_m \\ \vdots \\ I_m \end{bmatrix} u(k-1) + \begin{bmatrix} I_m & & & \\ I_m & I_m & & \\ \vdots & \vdots & \ddots & \\ I_m & I_m & \dots & I_m \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix}$$

- Therefore, we can write:

$$U(k) = Eu(k-1) + H\Delta U(k)$$

MPC With Control Constraints

- Suppose that we have the following constraints:

$$u^{\min} \leq U(k) \leq u^{\max}$$

- We can represent the above constraints as:

$$\begin{bmatrix} -U(k) \\ U(k) \end{bmatrix} \leq \begin{bmatrix} -u^{\min} \\ u^{\max} \end{bmatrix}$$

- Recall that

$$U(k) = Eu(k-1) + H\Delta U(k)$$

- Since $u(k-1)$ is known, we obtain an $Ax \leq b$ -like inequality:

$$\begin{bmatrix} -H \\ H \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -u^{\min} + Eu(k-1) \\ u^{\max} - Eu(k-1) \end{bmatrix}$$

- Input-Constrained MPC—a quadratic program:

$$\begin{array}{ll} \text{minimize} & J(\Delta U) = \frac{1}{2}(r - Y)^\top Q(r - Y) + \frac{1}{2}\Delta U^\top R\Delta U \\ \text{subject to} & \begin{bmatrix} -H \\ H \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -u^{\min} + Eu(k-1) \\ u^{\max} - Eu(k-1) \end{bmatrix} \end{array}$$

MPC With Output Constraints

- Suppose that we require the output to be bounded:

$$y^{\min} \leq Y(k) \leq y^{\max}$$

- Hence, we can write:

$$\begin{bmatrix} -Y(k) \\ Y(k) \end{bmatrix} \leq \begin{bmatrix} -y^{\min} \\ y^{\max} \end{bmatrix}$$

- Recall that $Y(k) = Wx_a(k) + Z\Delta U(k)$
- Similar to the input-constraints, we obtain:

$$\begin{bmatrix} -Z \\ Z \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -y^{\min} + Wx_a(k) \\ y^{\max} - Wx_a(k) \end{bmatrix}$$

- Output-Constrained MPC—a quadratic program:

$$\begin{array}{ll} \text{minimize} & J(\Delta U) = \frac{1}{2}(r - Y)^\top Q(r - Y) + \frac{1}{2}\Delta U^\top R\Delta U \\ \text{subject to} & \begin{bmatrix} -Z \\ Z \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -y^{\min} + Wx_a(k) \\ y^{\max} - Wx_a(k) \end{bmatrix} \end{array}$$

Constrained MPC as an Optimization Problem

- As we saw in the previous 3–4 slides, MPC problem can be written as:

$$\begin{array}{ll} \text{minimize} & J(\Delta U) \text{ (quadratic function)} \\ \text{subject to} & g(\Delta U) \leq 0 \text{ (linear constraints)} \end{array}$$

- Hence, we solve a constrained optimization problem (possibly convex) for each time-horizon
- Linear constraints can include constraints on: input, output, or rate of change (or their combination)
- Plethora of methods to solve such optimization problems
- How about nonlinear constraints? Can be included too!

MPC Pros and Cons

Pros:

- Easy way of dealing with constraints on controls and states
- **High performance** controllers, accurate
- No need to generate solutions for the whole time-horizon
- **Flexibility:** any model, any objective

Cons:

- Main disadvantage: **Online** computations in real-time
- Solving **constrained optimization** problem might be a daunting task
- Might be *stuck* with an unfeasible solution
- **Robustness and stability**

Explicit MPC

- Solving MPC online might be a problem for applications with fast sampling time ($< 1\text{msec}$)
- **Solution:** Explicit MPC (EMPC) — solving problems offline
- Basic idea: offline computations to determine all operating regions
- EMPC controllers require fewer run-time computations
- To implement explicit MPC, first design a traditional MPC
- Then, use this controller to generate an EMPC for use in real-time control
- Check <http://www.mathworks.com/help/mpc/explicit-mpc-design.html?refresh=true>

Questions And Suggestions?



Thank You!

Please visit

engineering.utsa.edu/~taha

IFF you want to know more 😊

References I

- ① Wang, Liuping. *Model predictive control system design and implementation using MATLAB*. Springer Science & Business Media, 2009.
- ② Course on *Model Predictive Control* — <http://control.ee.ethz.ch/index.cgi?page=lectures;action=details;id=67>
- ③ Żak, Stanislaw H. *Systems and control*. New York: Oxford University Press, 2003.
- ④ Course on Optimal Control, Lecture Notes — Żak, Stanislaw H., Purdue University, 2013.
- ⑤ MATLAB's EMPC page — <http://www.mathworks.com/help/mpc/explicit-mpc-design.html?refresh=true>