Module 09 — Optimization, Optimal Control, and Model Predictive Control

Ahmad F. Taha

EE 5143: Linear Systems and Controls

Email: ahmad.taha@utsa.edu

Webpage: http://engineering.utsa.edu/ataha

November 21, 2017

Constrained Control of Dynamic Systems

- A summary of what we learned so far
- The control methods we discussed do not consider strict constraints on states, control inputs, etc...
- [–] Examples: Constraints on maximum speed, minimum/maximum room temperature, minimum fuel level
- State feedback control, Gramian-based control, observer-based control in general do not respect these constraints—-they're not designed to do so anyway
- This module: an introduction to the idea of optimization, optimal control, and model predictive control

Solving Unconstrained Optimization Problems

Objective:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\ f(x)$$

Necessary & Sufficinet Conditions for Optimality

x^* is a local minimum of $f(x)$ iff:

1. Zero gradient at x^*:

$$\nabla_x f(x^*) = 0$$

2. Hessian at x^* is positive semi-definite:

$$\nabla_x^2 f(x^*) \succeq 0$$

- For maximization, Hessian is negative semi-definite
- The idea of local/global minima
- Convexity in optimization

Solving Constrained OPs

- **Main objective:** find/compute minimum or a maximum of an objective function subject to equality and inequality constraints
- Formally, problem defined as finding the optimal x^*:

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

  - $x \in \mathbb{R}^n$
  - $f(x)$ is scalar function, possibly nonlinear
  - $g(x) \in \mathbb{R}^m, h(x) \in \mathbb{R}^l$ are vectors of constraints

Main Principle

To solve constrained optimization problems: transform constrained problems to unconstrained ones. How?
Augment the constraints to the cost function.

General Optimization Problems and KKT Conditions

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

- Define the Lagrangian: $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$

Optimality Conditions

The constrained optimization problem (above) has a local minimizer x^* iff there exists a unique μ^* such that:

1. $\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = \nabla_x f(x) + \lambda^{*T} \nabla_x h(x^*) + \mu^{*T} \nabla_x g(x^*) = 0$
2. $\mu_j^* \geq 0$ for $j = 1, \ldots, m$
3. $\mu_j^* g_j(x^*) = 0$ for $j = 1, \ldots, m$
4. $g_j(x^*) \leq 0$ for $j = 1, \ldots, m$
5. $h_i(x^*) = 0$ for $i = 1, \ldots, l$ (if x^*, μ^*, λ^* satisfy 1–5, they are candidates)
6. Second order necessary conditions (SONC): $\nabla_x^2 \mathcal{L}(x^*, \lambda^*, \mu^*) \succeq 0$

KKT Conditions — Example

Find the minimizer of the following optimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) = (x_1 - 1)^2 + x_2 - 2 \\ \text{subject to} \quad & g(x) = x_1 + x_2 - 2 \leq 0 \\ & h(x) = x_2 - x_1 - 1 = 0 \end{aligned}$$

- First, find the Lagrangian function:

$$\mathcal{L}(x, \lambda, \mu) = (x_1 - 1)^2 + x_2 - 2 + \lambda(x_2 - x_1 - 1) + \mu(x_1 + x_2 - 2)$$

- Second, find the conditions of optimality (from previous slide):

1. $\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = \begin{bmatrix} 2x_1^* - 2 - \lambda^* + \mu^* & 1 + \lambda^* + \mu^* \end{bmatrix}^\top = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top$
2. $\mu^*(x_1^* + x_2^* - 2) = 0$
3. $\mu^* \geq 0$
4. $x_1^* + x_2^* - 2 \leq 0$
5. $x_2^* - x_1^* - 1 = 0$
6. $\nabla_x^2 \mathcal{L}(x^*, \lambda^*, \mu^*) = \nabla_x^2 f(x^*) + \lambda^* \nabla_x^2 h(x^*) + \mu^* \nabla_x^2 g(x^*) \succeq 0$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + \lambda^* \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \mu^* \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \succeq 0$$

Example — Cont'd

- To solve the system equations for the optimal x^*, λ^*, μ^*, we first try $\mu^* > 0$.
- Given that, we solve the following set of equations:
  1. $2x_1^* - 2 - \lambda^* + \mu^* = 0$
  2. $1 + \lambda^* + \mu^* = 0$
  3. $x_1^* + x_2^* - 2 = 0$
  4. $x_2^* - x_1^* - 1 = 0$
  5. $[\Rightarrow]\ x_1^* = 0.5, x_2^* = 1.5, \lambda^* = -1, \mu^* = 0$
- But this solution contradicts the assumption that $\mu^* > 0$
- **Alternative:** assume $\mu^* = 0 \Rightarrow x_1^* = 0.5, x_2^* = 1.5, \lambda^* = -1, \mu^* = 0$
- This solution satisfies $g(x^*) \leq 0$ constraint, hence it's a candidate for being a minimizer
- We now verify the SONC: $L(x^*, \lambda^*, \mu^*) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \succeq 0$
- Thus, $x^* = \begin{bmatrix} 0.5 & 1.5 \end{bmatrix}^\top$ is a strict local minimizer

Optimization Solvers and Taxonomy

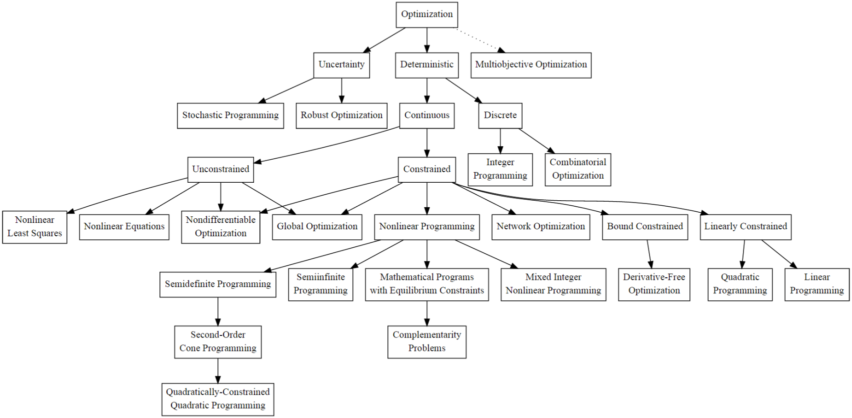


Figure from:
http://www.neos-guide.org/content/optimization-introduction

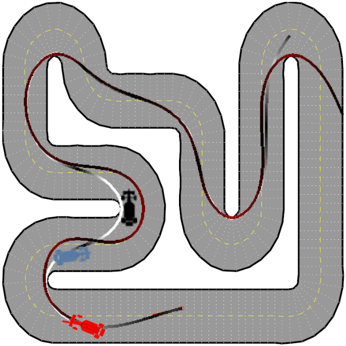
Solvers

- Solving optimization problems require few things
  1. Modeling the problem
  2. Translating the problem model (constraints and objectives) into a modeling language (`AMPL`, `GAMS`, `MATLAB`, `YALMIP`, `CVX`)
  3. Choosing optimization algorithms solvers (`Simplex`, `Interior-Point`, `Brand & Bound`, `Cutting Planes`,...)
  4. Specifying tolerance, exit flags, flexible constraints, bounds, ...
- Convex optimization problems: use `cvx` (super easy to install and code)
- MATLAB's `fmincon` is always handy too (too much overhead, often fails to converge for nonlinear optimization problems)
- Visit http://www.neos-server.org/neos/solvers/index.html
- Check http://www.neos-guide.org/ to learn more

Introduction to MPC — Example[1]

What is Model-Predictive Control?

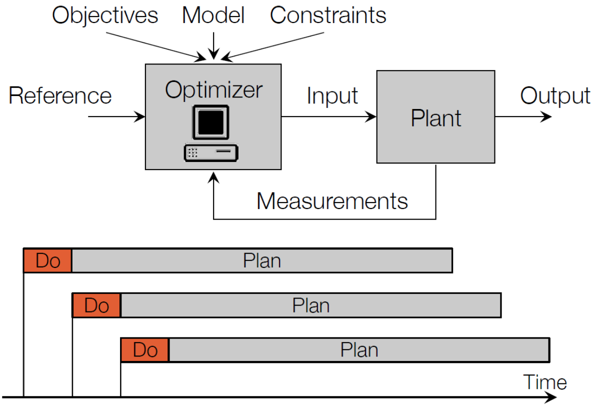
- Compute first control action (for a prediction horizon)
- Apply first control action
- Repeat given updated constraints
- Essentially, solving optimization problems sequentially
- Use static-optimization techniques for optimal control problems
- **Example:** minimizing `LapTime`, while `NotKillingPeople`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



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

MPC Schematic

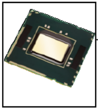
MPC leverages constrained static-optimization for optimal control problems



MPC: real-time, sequential optimization with constraints on states and inputs[2]

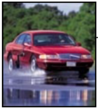
[2]Some figures are borrowed from the references; see the end of the presentation file.

MPC Applications + Time Horizons



Computer control — ns

μs — Power systems



Traction control — ms

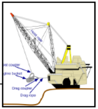
Seconds — Buildings

Refineries — Minutes

Hours — Nurse rostering

Train scheduling — Days

Weeks — Production planning

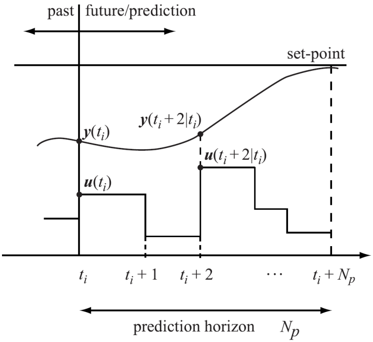


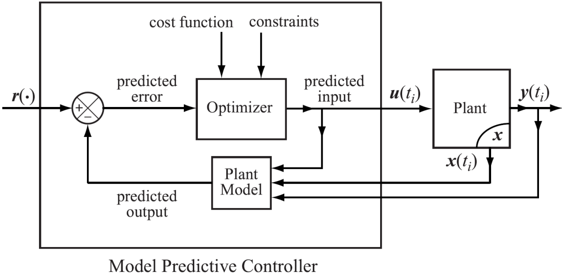
MPC Constraints

- 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 optimal control problems

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 online[3] (explicit MPC, EMPC, is solved offline)





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

Discrete LMPC Formulation

Linear MPC Problem

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

- At each time-instant:
  1. Measure or estimate $x(t)$
  2. Find optimal input sequence the `PredictionHorizon` (N_p)

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

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

Linear Discrete-Time MPC

Objective is to apply MPC for this LTI DT system:

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

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

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

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

MPC Problem Construction

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

- 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

$$\Delta u(k), \Delta u(k+1), \ldots, \Delta u(k+N_p-1), \;\; N_p = \texttt{PredictionHorizon}$$

- This sequence will give us the predicted state vectors

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

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**:

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

- Also, we can write the predicted outputs as:

$$\underbrace{C_a \begin{bmatrix} x_a(k+1|k) \\ x_a(k+2|k) \\ \vdots \\ x_a(k+N_p|k) \end{bmatrix}}_{Y} = \underbrace{C_a \begin{bmatrix} \Phi_a \\ \Phi_a^2 \\ \vdots \\ \Phi_a^{N_p} \end{bmatrix}}_{W} x_a(k) + \underbrace{C_a \begin{bmatrix} \Gamma_a & & & \\ \Phi_a \Gamma_a & \Gamma_a & & \\ \vdots & & \ddots & \\ \Phi_a^{N_p-1}\Gamma_a & \ldots & \Phi_a \Gamma_a & \Gamma_a \end{bmatrix}}_{Z} \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix}}_{\Delta U}$$

- Hence, we obtain:

$$\boxed{Y = \begin{bmatrix} y^\top(k+1|k) & y^\top(k+2|k) & \ldots & y^\top(k+N_p|k) \end{bmatrix}^\top = W x_a(k) + Z \Delta U}$$

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

Optimal MPC Construction

$$Y = \begin{bmatrix} y^\top(k+1|k) & y^\top(k+2|k) & \dots & y^\top(k+N_p|k) \end{bmatrix}^\top = W x_a(k) + Z\Delta U$$

- 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:

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

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

Optimal MPC Construction — 2

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

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

- Above equation can be written as:

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

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

$$\begin{aligned}\Delta u(k) &= K_r r - K_{mpc} \Delta x(k) - K_y y(k) \\ \Delta u(k) &= \underbrace{K_r r - K_y y(k)}_{\text{reference signals}} - \underbrace{K_{mpc} \Delta x(k)}_{\text{state-feedback gain}}, \text{ where:} \\ K_r &= \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} (R + Z^\top Q Z)^{-1} Z^\top Q \\ K_{mpc} &= K_r W \begin{bmatrix} I_n \\ O \end{bmatrix}, \; K_y = K_r W \begin{bmatrix} O \\ I_p \end{bmatrix}\end{aligned}$$

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:

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

- Apply the algorithm:

1. Augmented dynamics:

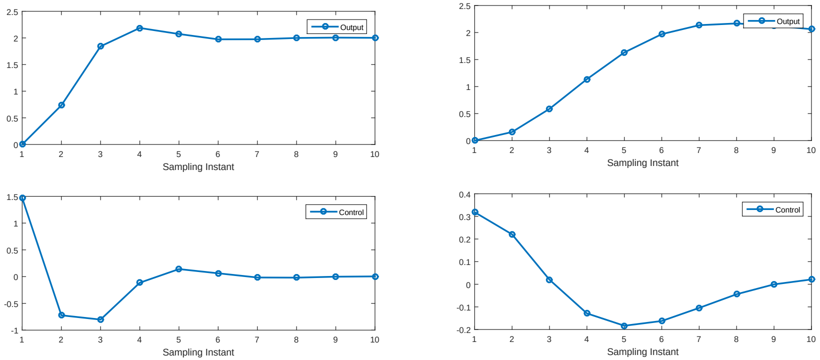
$$\Phi_a=\begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 0\\ 1 & 1 & 1\end{bmatrix}, \Gamma_a=\begin{bmatrix}0.5\\ 1\\ 0\end{bmatrix}, C_a=\begin{bmatrix}0 & 0 & 1\end{bmatrix} \Rightarrow$$

2. 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 know, 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:

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

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:

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

Constrained MPC as an Optimization Problem

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

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

- 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 (< 1msec)
- **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