Course on Model Predictive Control
Part II – Linear MPC design

Gabriele Pannocchia

Department of Chemical Engineering, **University of Pisa**, Italy
Email: `g.pannocchia@diccism.unipi.it`

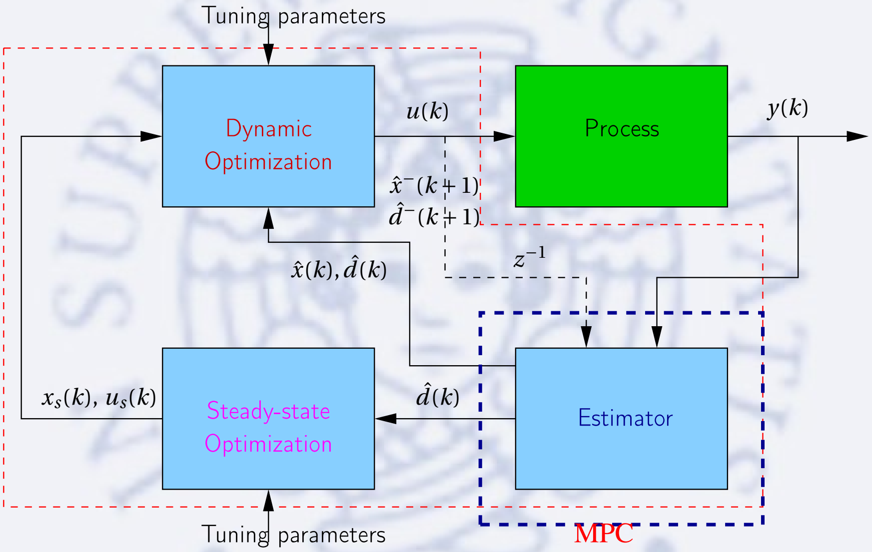
Centro E. Piaggio
bioengineering and robotics research center

Facoltà di Ingegneria, Pisa.
July 16th, 2012, Aula Pacinotti

Outline

1. Estimator module design for offset-free tracking
2. Steady-state optimization module design
3. Dynamic optimization module design
4. Closed-loop implementation and receding horizon principle
5. Quick overview of numerical optimization

Estimator module



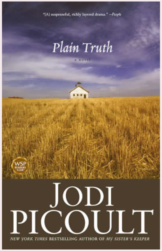
Estimator preliminaries

Basic model, inputs and outputs

Basic model:

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

- Inputs k: **measured output** $y(k)$ and **predicted state** $\hat{x}^-(k)$
- Outputs k: **updated state** estimate $\hat{x}(k)$



State estimator for basic model

- Choose **any** L such that $(A - ALC)$ is strictly Hurwitz
- **Filtering**: $\hat{x}(k) = \hat{x}^-(k) + L(y(k) - C\hat{x}^-(k))$

Key observation

*The estimator is the **only feedback module** in an MPC. Any **discrepancy** between true plant and model **should be corrected** there*

Augmented system: definition

Issue and solution approach

- An MPC based on the previous estimator **does not compensate** for plant/model mismatch and persistent disturbances
- As in [Davison and Smith, 1971, Kwakernaak and Sivan, 1972, Smith and Davison, 1972, Francis and Wonham, 1976], one should **model** and estimate the **disturbance to be rejected**
- For **offset-free control**, an **integrating disturbance** is added

Augmented system [Muske and Badgwell, 2002, Pannocchia and Rawlings, 2003], with $d \in \mathbb{R}^{n_d}$

$$\begin{bmatrix} x \\ d \end{bmatrix}^+ = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} w \\ w_d \end{bmatrix}$$

$$y = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + v$$

Augmented system: observability

Observability of the augmented system

- Assume that (A, C) is **observable**
- **Question**: is $\left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix}, \begin{bmatrix} C & C_d \end{bmatrix}\right)$ observable?
- **Answer**: yes, if and only if (from the Hautus test)

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

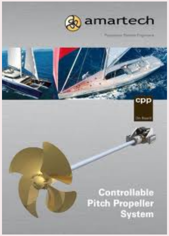
- **Observation**: the previous can be satisfied if and only if

$$n_d \le p$$

Augmented system: controllability

Controllability of the augmented system

- Assume that (A, B) is **controllable**
- **Question**: is $\left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix}, \begin{bmatrix} B \\ 0 \end{bmatrix} \right)$ controllable?
- **Answer**: No
- **Observation**: the disturbance is **not going to be controlled**. Its **effect** is taken into account in Steady-State Optimization and Dynamic Optimization modules



Augmented estimator

General design

- Set $n_d = p$ (see [Pannocchia and Rawlings, 2003, Maeder et al., 2009] for issues on choosing $n_d < p$)
- Choose (B_d, C_d) such that the **augmented system** is **observable**
- Choose $L = \begin{bmatrix} L_x \\ L_d \end{bmatrix}$ such that

$$\left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} - \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right)$$

  is strictly **Hurwitz**

- **Augmented estimator**:

$$\begin{bmatrix} \hat{x}(k) \\ \hat{d}(k) \end{bmatrix} = \begin{bmatrix} \hat{x}^-(k) \\ \hat{d}^-(k) \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(y(k) - \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} \hat{x}^-(k) \\ \hat{d}^-(k) \end{bmatrix} \right)$$

The typical industrial design

Typical industrial design for stable systems (A strictly Hurwitz)

$$B_d = 0, \quad C_d = I, \quad L_x = 0, \quad L_d = I$$

Output disturbance model

- Any error $y(k) - C\hat{x}^-(k)$ is assumed to be caused by a **step (constant) disturbance** acting on the **output**. In fact, the **filtered disturbance** estimate is:

$$\hat{d}(k) = \hat{d}^-(k) + (y(k) - C\hat{x}^-(k) - \hat{d}^-(k)) = y(k) - C\hat{x}^-(k)$$

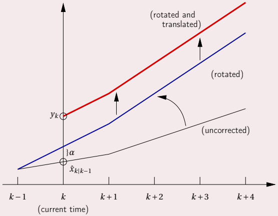
- It is a **deadbeat** Kalman filter

$$Q = 0, \quad Q_d = I, \quad R \to 0$$

- It is **simple** and does the job [Rawlings et al., 1994]

Comments on the industrial design

Rotation factor for integrating systems

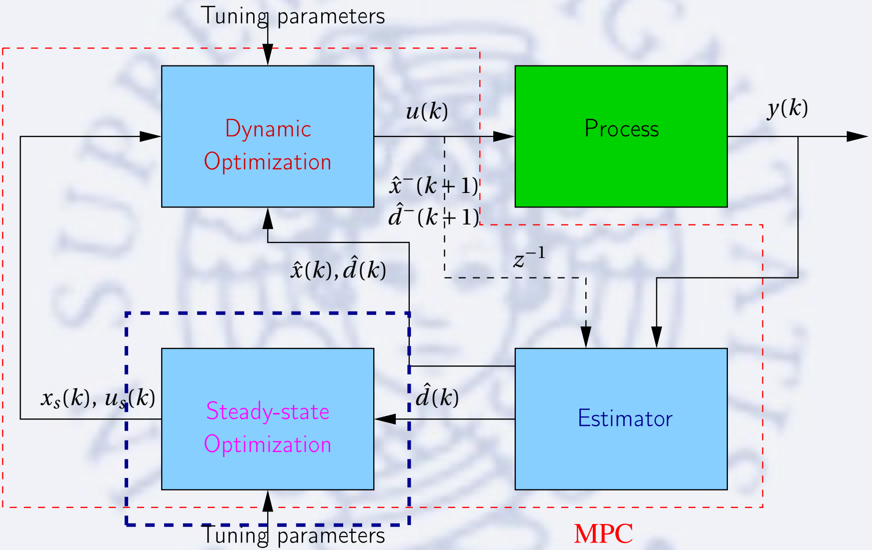


Limitations of the output disturbance model

- The overall **performance** is often **sluggish** [Lundström et al., 1995, Muske and Badgwell, 2002, Pannocchia and Rawlings, 2003, Pannocchia, 2003]
- A suitable estimator design can **improve the closed-loop performance** [Pannocchia, 2003, Pannocchia and Bemporad, 2007, Rajamani et al., 2009]
- **Often**, a deadbeat **input disturbance model** works better

$$B_d = B, \quad C_d = 0, \quad Q = 0, \quad Q_d = I, \quad R \to 0$$

Steady-state optimization module



Steady-state optimization module: introduction

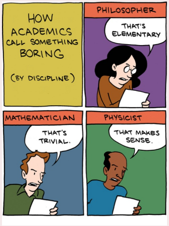
A trivial case

- **Square** system ($m = p$) **without constraints** and **with setpoints** on all CVs
- **Solve** the linear system

$$x_s = Ax_s + Bu_s + B_d\hat{d}$$
$$r_s = Cx_s + C_d\hat{d}$$

- Obtain

$$\begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} B_d\hat{d} \\ r_{sp} - C_d\hat{d} \end{bmatrix}$$



Observation

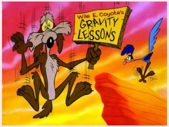
The steady-state target (x_s, u_s) may **change** at **each decision time** because of the **disturbance** estimate $\hat{d}$

Objectives of the steady-state optimization module

Boundary conditions

- In most cases the number of CVs is **different** from the number of MVs: $p \neq m$
- CVs and MVs have **constraints** to meet

$$y_{\min} \leq y(k) \leq y_{\max}, \qquad u_{\min} \leq u(k) \leq u_{\max}$$



- Only a small **subset of CVs** have fixed **setpoints**: $Hy(k) \rightarrow r_{sp}$

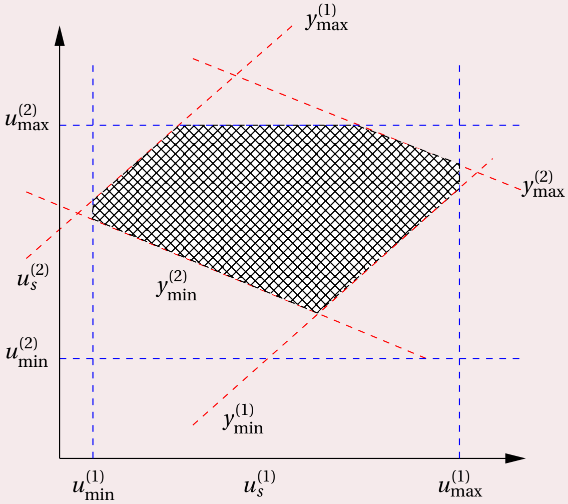
Objectives

- Given the current **disturbance estimate**, $\hat{d}$
- Compute the **equilibrium** (x_s, u_s, y_s):
  $x_s = Ax_s + Bu_s + B_d\hat{d}$, $y_s = Cx_s + C_d\hat{d}$ such that:
  - **constraints** on MVs and CVs are satisfied:
    $y_{\min} \leq y_s \leq y_{\max}$, $u_{\min} \leq u_s \leq u_{\max}$
  - the subset of CVs tracks the **setpoint**: $Hy_s = r_s$

Examples of steady-state optimization feasible regions

Stable system with 2 MVs and 2 CVs (with bounds)

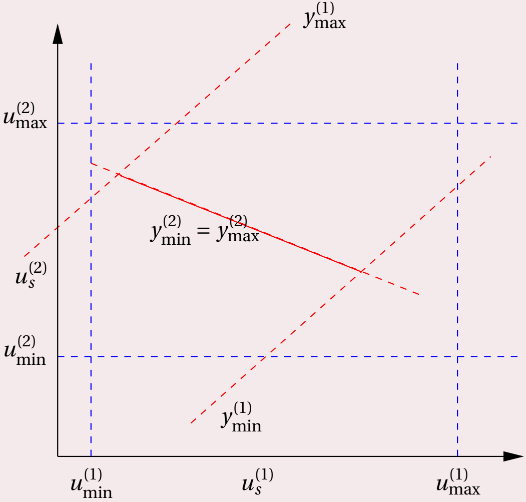
$$y_s = C(I-A)^{-1}Bu_s + (C(I-A)^{-1}B_d + C_d)\hat{d} = Gu_s + G_d\hat{d}$$



Examples of steady-state optimization feasible regions

Stable system with 2 MVs and 2 CVs (one setpoint)

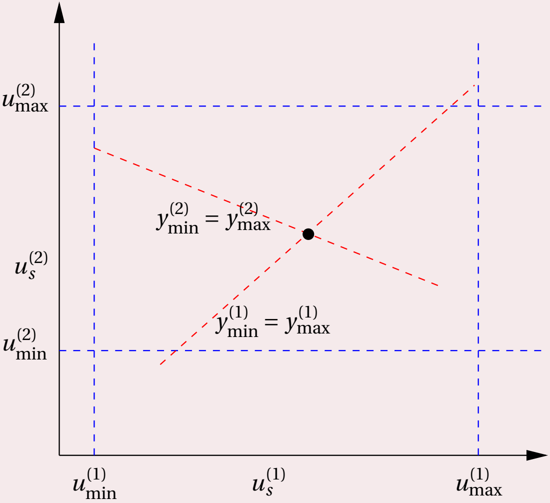
$$y_s = C(I-A)^{-1}Bu_s + (C(I-A)^{-1}B_d + C_d)\hat{d} = Gu_s + G_d\hat{d}$$



Examples of steady-state optimization feasible regions

Stable system with 2 MVs and 2 CVs (two setpoints)

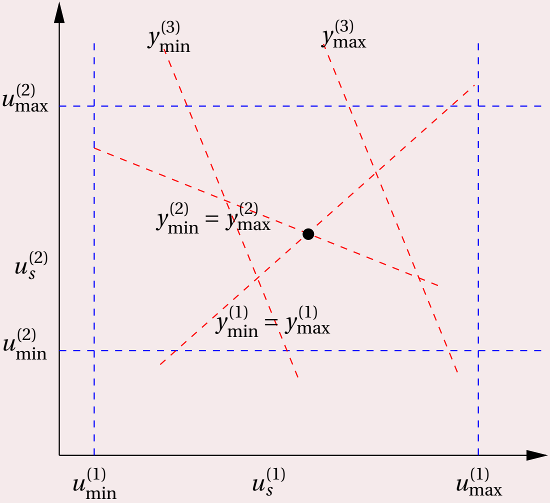
$$y_s = C(I-A)^{-1}Bu_s + (C(I-A)^{-1}B_d + C_d)\hat{d} = Gu_s + G_d\hat{d}$$



Examples of steady-state optimization feasible regions

Stable system with 2 MVs and 3 CVs (two setpoints)

$$y_s = C(I-A)^{-1}Bu_s + (C(I-A)^{-1}B_d + C_d)\hat{d} = Gu_s + G_d\hat{d}$$



Hard and soft constraints

Hard constraints

- In the two optimization modules, constraints on **MVs** are regarded as **hard**, i.e., cannot be violated
- This choice comes from the **possibility** of satisfying them exactly

Soft constraints

- In the two optimization modules, constraints on **CVs** are regarded as **soft**, i.e., can be violated when necessary
- The amount of violation is **penalized** in the objective function
- This choice comes from the **impossibility** of satisfying them exactly

Steady-state optimization module: linear formulation

General formulation (LP)

$$\min_{u_s, x_s, \overline{\epsilon}_s, \underline{\epsilon}_s} \overline{q}'\overline{\epsilon}_s + \underline{q}'\underline{\epsilon}_s + r'u_s \qquad \text{s.t.}$$

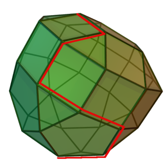
$$x_s = Ax_s + Bu_s + B_d\hat{d}$$

$$u_{\min} \leq u_s \leq u_{\max}$$

$$y_{\min} - \underline{\epsilon}_s \leq Cx_s + C_d\hat{d} \leq y_{\max} + \overline{\epsilon}_s$$

$$\overline{\epsilon}_s \geq 0$$

$$\underline{\epsilon}_s \geq 0$$



Extensions

- **Setpoints** on some CVs can be specified with either an **equality constraint** or by setting **identical** value for **minimum and maximum** value
- Often CVs are grouped by **ranks**

LP steady-state optimization module: tuning

Cost function

- $r \in \mathbb{R}^m$ is the **MV cost** vector: a **positive** (negative) entry in r implies that the corresponding MV should be **minimized** (maximized)
- $\overline{q} \in \mathbb{R}^p$ and $\underline{q} \in \mathbb{R}^p$ are the weights of upper and lower bound **CV violations**. Sometimes they are defined in terms of **equal concern error**:
  $\overline{q}_i = \frac{1}{\text{SSECE}_i^U}, \quad \underline{q}_i = \frac{1}{\text{SSECE}_i^L}, \quad i = 1, \ldots, p$

Constraints

- Bounds $u_{\max}$, $u_{\min}$, $y_{\max}$, $y_{\min}$ **can be modified** by the operators (within ranges defined by the MPC designer)
- Sometimes in order to **avoid large target changes** from time $k-1$ to time k a rate constraint is added

$$-\Delta u_{\max} \leq u_s(k) - u_s(k-1) \leq \Delta u_{\max}$$

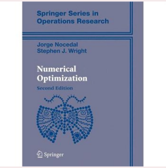
LP steady-state optimization module: numerical solution

Standard LP form

$$\min_z c' z \qquad \text{s.t.}$$
$$Ez \le e, \qquad Fz = f$$

with

$$z = \begin{bmatrix} x_s \\ u_s \\ \bar{\epsilon}_s \\ \underline{\epsilon}_s \end{bmatrix}, E = \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & -I & 0 & 0 \\ C & 0 & -I & 0 \\ -C & 0 & 0 & -I \\ 0 & 0 & -I & 0 \\ 0 & 0 & 0 & -I \end{bmatrix}, e = \begin{bmatrix} u_{\max} \\ -u_{\min} \\ y_{\max} - C_d \hat{d} \\ -(y_{\min} - C_d \hat{d}) \\ 0 \\ 0 \end{bmatrix}, F = \begin{bmatrix} I-A & -B & 0 & 0 \end{bmatrix}, f = B_d \hat{d}$$



Solution algorithms [Nocedal and Wright, 2006]

Solution methods based on **simplex** or **interior point** algorithms

Steady-state optimization module: quadratic formulation

QP formulation

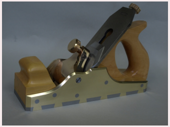
- The **LP** formulation is quite **intuitive** but its outcome may be too "jumpy"
- Sometimes a **QP** formulation may be preferred

$$\min_{u_s, x_s, \overline{\epsilon}_s, \underline{\epsilon}_s} \|\overline{\epsilon}_s\|^2_{\overline{Q}_s} + \|\underline{\epsilon}_s\|^2_{\underline{Q}_s} + \|u_s - u_{sp}\|^2_{R_s} \qquad \text{s.t.}$$

$$x_s = Ax_s + Bu_s + B_d\hat{d}$$

$$u_{\min} \le u_s \le u_{\max}$$

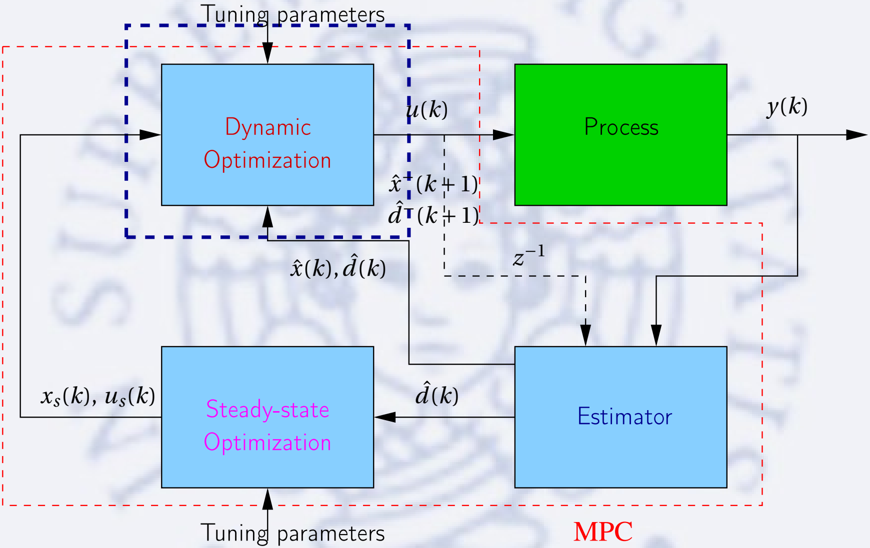
$$y_{\min} - \underline{\epsilon}_s \le Cx_s + C_d\hat{d} \le y_{\max} + \overline{\epsilon}_s$$



where u_{sp} is the desired MV setpoint and $\|x\|^2_Q = x'Qx$

- **Tuning** is slightly more **complicated**, but the outcome is usually "**smoother**"

Dynamic optimization module

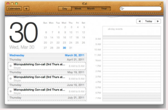


Dynamic optimization module: introduction

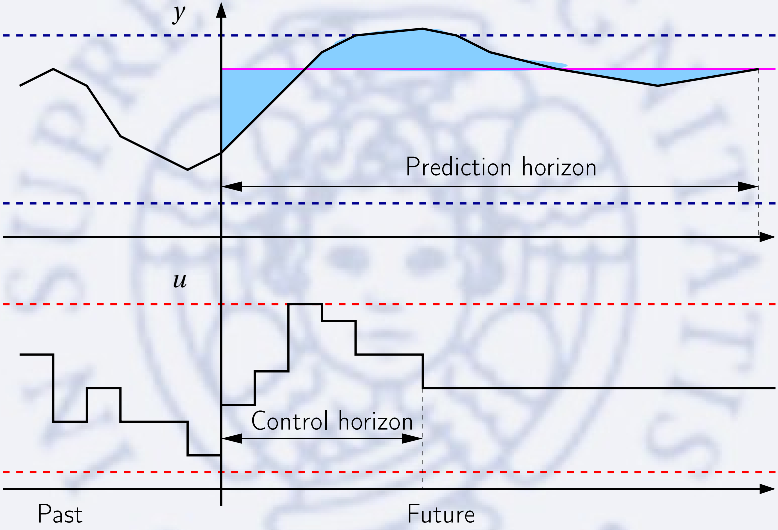
Agenda

- Make a **finite-horizon** prediction of future **CVs evolution** based on a sequence of MVs
- Find the **optimal** MVs sequence, minimizing a **cost function** that comprises:
  - **deviation** of CVs (and MVs) from their **targets**
  - **rate of change** of MVs

  respecting **constraints** on:
  - MVs (**always**)
  - CVs (**possibly**)



Dynamic optimization module: graphical interpretation



Dynamic optimization module: formulation

Cost function: Q, either R or S, $\overline{Q}$, $\underline{Q}$, P positive definite matrices

$$V_N(\hat{x}, \mathbf{u}, \overline{\boldsymbol{\epsilon}}, \underline{\boldsymbol{\epsilon}}) = \sum_{j=0}^{N-1} \Big[\|\hat{y}(j) - y_s\|_Q^2 + \|u(j) - u_s\|_R^2 + \|u(j) - u(j-1)\|_S^2 +$$

$$\|\overline{\epsilon}(j)\|_{\overline{Q}}^2 + \|\underline{\epsilon}(j)\|_{\underline{Q}}^2 \Big] + \|x(N) - x_s\|_P^2 \qquad \text{s.t.}$$

$$x^+ = Ax + Bu + B_d\hat{d} \qquad x(0) = \hat{x}$$

$$\hat{y} = Cx + C_d\hat{d} \qquad y_s = Cx_s + C_d\hat{d}$$

Control problem

$$\min_{\mathbf{u}, \overline{\boldsymbol{\epsilon}}, \underline{\boldsymbol{\epsilon}}} V_N(\hat{x}, \mathbf{u}, \overline{\boldsymbol{\epsilon}}, \underline{\boldsymbol{\epsilon}}) \qquad \text{s.t.}$$

$$u_{\min} \le u(j) \le u_{\max}$$

$$-\Delta u_{\max} \le u(j) - u(j-1) \le \Delta u_{\max}$$

$$y_{\min} - \underline{\epsilon}(j) \le \hat{y}(j) \le y_{\max} + \overline{\epsilon}(j)$$

Dynamic optimization module: formulation

Main tuning parameters

- Q: diagonal matrix of weights for **CVs deviation** from target:

$$q_{ii} = \left(\frac{1}{DECE_i^M} \right)^2$$

- R: diagonal matrix of weights for **MVs deviation** from target
- S: diagonal matrix of weights for MVs rate of change, often called **move suppression factors**

- $\overline{Q}$, $\underline{Q}$: diagonal matrices of weights for **CVs violation** of constraints

$$\overline{q}_{ii} = \left(\frac{1}{DECE_i^U} \right)^2, \qquad \underline{q}_{ii} = \left(\frac{1}{DECE_i^L} \right)^2$$

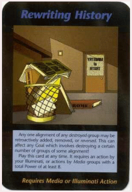
- $u_{\max}$, $u_{\min}$, $y_{\max}$, $y_{\min}$, $\Delta u_{\max}$: constraint vectors
- N: prediction horizon

Deviation variables

- Use the **target** (x_s, u_s):
  $\tilde{x}(j) = x(j) - x_s, \quad \tilde{u}(j) = u(j) - u_s,$
- **Recall** that:
  $x_s = Ax_s + Bu_s + B_d\hat{d}, \quad y_s = Cx_s + C_d\hat{d}$
- Cost function **becomes**:

$$V_N(\hat{x}, \tilde{\mathbf{u}}, \bar{\boldsymbol{\epsilon}}, \underline{\boldsymbol{\epsilon}}) = \sum_{j=0}^{N-1} \Big[\|\tilde{x}(j)\|^2_{C'QC} + \|\tilde{u}(j)\|^2_R + \|\tilde{u}(j) - \tilde{u}(j-1)\|^2_S + \|\bar{\epsilon}(j)\|^2_{\bar{Q}} + \|\underline{\epsilon}(j)\|^2_{\underline{Q}} \Big] + \|\tilde{x}(N)\|^2_P \qquad \text{s.t.}$$

$$\tilde{x}^+ = A\tilde{x} + B\tilde{u} \qquad \tilde{x}(0) = \hat{x} - x_s$$



Dynamic optimization module: constrained LQR

Compact problem formulation

$$\min_{\tilde{\mathbf{u}},\bar{\boldsymbol{\epsilon}},\underline{\boldsymbol{\epsilon}}} V_N(\hat{x},\tilde{\mathbf{u}},\bar{\boldsymbol{\epsilon}},\underline{\boldsymbol{\epsilon}}) \qquad \text{s.t.}$$

$$u_{\min} - u_s \leq \tilde{u}(j) \leq u_{\max} - u_s$$

$$-\Delta u_{\max} \leq \tilde{u}(j) - \tilde{u}(j-1) \leq \Delta u_{\max}$$

$$y_{\min} - y_s - \underline{\epsilon}(j) \leq C\tilde{x}(j) \leq y_{\max} - y_s + \bar{\epsilon}(j)$$

Constrained LQR formulation

With suitable definitions (shown next), we obtain a **constrained LQR** formulation:

$$\min_{\mathbf{u}_a} V_N(x_a,\mathbf{u}_a) = \sum_{j=0}^{N-1}\Big[\|x_a(j)\|^2_{Q_a} + \|u_a(j)\|^2_{R_a} + 2x_a(j)M_a u_a(j)\Big] + \|x_a(N)\|^2_{P_a}$$

$$\text{s.t. } x_a^+ = A_a x_a + B_a u_a, \qquad D_a u_a + E_a x_a \leq e_a$$

Dynamic optimization module: constrained LQR (cont.'d)

Augmented state and input

$$x_a(j) = \begin{bmatrix} \tilde{x}(j) \\ \tilde{u}(j-1) \end{bmatrix}, \qquad u_a(j) = \begin{bmatrix} \tilde{u}(j) \\ \overline{\epsilon}(j) \\ \underline{\epsilon}(j) \end{bmatrix}$$

- The **state is augmented** to write terms $u(j) - u(j-1)$ (in the objective function and/or in the constraints)
- The **input is augmented** to write the soft output constraints

Matrices and vectors

$$A_a = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \quad B_a = \begin{bmatrix} B & 0 & 0 \\ I & 0 & 0 \end{bmatrix} \quad Q_a = \begin{bmatrix} C'QC & 0 \\ 0 & S \end{bmatrix} \quad R_a = \begin{bmatrix} R+S & 0 & 0 \\ 0 & \overline{Q} & 0 \\ 0 & 0 & \underline{Q} \end{bmatrix} \quad M_a = \begin{bmatrix} 0 & 0 & 0 \\ -S & 0 & 0 \end{bmatrix}$$

$$D_a = \begin{bmatrix} I & 0 & 0 \\ -I & 0 & 0 \\ I & 0 & 0 \\ -I & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{bmatrix} \quad E_a = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & -I \\ 0 & I \\ C & 0 \\ -C & 0 \end{bmatrix} \quad e_a = \begin{bmatrix} u_{\max} - u_s \\ u_s - u_{\min} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ y_{\max} - y_s \\ y_s - y_{\min} \end{bmatrix}$$

From constrained LQR to a QP problem

$$\mathbf{x}_a = \begin{bmatrix} x_a(0) \\ x_a(1) \\ x_a(2) \\ \vdots \\ x_a() \end{bmatrix} \quad \mathbf{A}_a = \begin{bmatrix} I \\ A_a \\ A_a^2 \\ \vdots \\ A_a^N \end{bmatrix} \quad \mathbf{B}_a = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_a & \ddots & & \vdots \\ A_aB_a & B_a & \ddots & \\ \vdots & & \ddots & 0 \\ A_a^{N-1}B_a & \cdots & A_aB_a & B_a \end{bmatrix} \Longrightarrow \mathbf{x}_a = \mathbf{A}_a x_a(0) + \mathbf{B}_a \mathbf{u}_a$$

$$\mathbf{Q}_a = \begin{bmatrix} Q_a & & & & \\ & Q_a & & & \\ & & \ddots & & \\ & & & Q_a & \\ & & & & P_a \end{bmatrix} \quad \mathbf{R}_a = \begin{bmatrix} R_a & & & \\ & R_a & & \\ & & \ddots & \\ & & & R_a \end{bmatrix} \quad \mathbf{M}_a = \begin{bmatrix} M_a & & & \\ & M_a & & \\ & & \ddots & \\ & & & M_a \\ 0 & \cdots & & 0 \end{bmatrix}$$

$$\mathbf{D}_a = \begin{bmatrix} D_a & & & \\ & D_a & & \\ & & \ddots & \\ & & & D_a \end{bmatrix} \quad \mathbf{E}_a = \begin{bmatrix} E_a & & & 0 \\ & E_a & & \vdots \\ & & \ddots & \\ & & E_a & 0 \end{bmatrix} \quad \mathbf{e}_a = \begin{bmatrix} e_a \\ e_a \\ \vdots \\ e_a \end{bmatrix}$$

Dynamic optimization module: QP solution (cont.'d)

QP problem

$$\min_{\mathbf{u}_a} \frac{1}{2}\mathbf{u}_a'\mathbf{H}_a\mathbf{u}_a + \mathbf{u}_a'\mathbf{q}_a \qquad \text{s.t.}$$

$$\mathbf{F}_a\mathbf{u}_a \leq \mathbf{e}_a - \mathbf{G}_a x_a(0)$$

where

$$\mathbf{H}_a = \mathbf{B}_a'\mathbf{Q}_a\mathbf{B}_a + \mathbf{R}_a + \mathbf{B}_a'\mathbf{M}_a + \mathbf{M}_a'\mathbf{B}_a \quad \mathbf{q}_a = (\mathbf{B}_a'\mathbf{Q}_a + \mathbf{M}_a')\mathbf{A}_a x_a(0)$$

$$\mathbf{F}_a = \mathbf{D}_a + \mathbf{E}_a\mathbf{B}_a \quad \mathbf{G}_a = \mathbf{E}_a\mathbf{A}_a$$

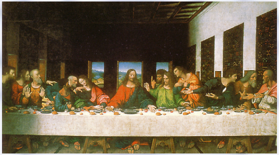
Observations

- Both the linear penalty and constraint RHS **vary linearly** with the current augmented state $x_a(0)$, while all other terms are fixed
- QP solvers are based on **Active-Set Methods** or **Interior Point Methods**

Feedback controllers synthesis from open-loop controllers: the receding horizon principle

A quote from [Lee and Markus, 1967]

One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function.
The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement.
The procedure is then repeated.



Optimal control sequence and closed-loop implementation

The optimal control sequence

- The optimal control sequence $\mathbf{u}_a^0$ is in the form

$$\mathbf{u}_a^0 = \left\{ \underbrace{\tilde{u}^0(0), \tilde{u}^0(1), \ldots, \tilde{u}^0(N-1)}_{\tilde{\mathbf{u}}^0}, \underbrace{\bar{\epsilon}(0), \bar{\epsilon}(1), \ldots, \bar{\epsilon}(N-1)}_{\bar{\boldsymbol{\epsilon}}}, \underbrace{\underline{\epsilon}(0), \underline{\epsilon}(1), \ldots, \underline{\epsilon}(N-1)}_{\underline{\boldsymbol{\epsilon}}} \right\}$$

- The variables $\bar{\boldsymbol{\epsilon}}$ and $\underline{\boldsymbol{\epsilon}}$ are present only when **soft output constraints** are used

Closed-loop implementation

- Only the **first element** of the optimal control sequence is **injected** into the plant: $u(k) = \tilde{u}^0(0) + u_s(k)$
- The **successor state** is **predicted** using the estimator model

$$\hat{x}^-(k+1) = A\hat{x}(k) + Bu(k) + B_d\hat{d}(k)$$
$$\hat{d}^-(k+1) = \hat{d}(k)$$

Linear MPC: summary

Overall algorithm

1. Given predicted state and disturbance $(\hat{x}^{-}(k), \hat{d}^{-}(k))$ and output measurement $y(k)$, compute **filtered estimate**:
   $\hat{x}(k) = \hat{x}^{-}(k) + L_x\left(y(k) - C\hat{x}^{-}(k) - C_d\hat{d}^{-}(k)\right),$
   $\hat{d}(k) = d^{-}(k) + L_d\left(y(k) - C\hat{x}^{-}(k) - C_d\hat{d}^{-}(k)\right)$
2. Solve **Steady-State Optimization** problem and compute targets $(x_s(k), u_s(k))$
3. Define deviation variables: $\tilde{x}(0) = \hat{x}(k) - u_s(k)$,
   $\tilde{u}(-1) = u(k-1) - u_s(k)$, and initial regulator state $x_a(0) = \begin{bmatrix} \tilde{x}(0) \\ \tilde{u}(-1) \end{bmatrix}$.
   Solve **Dynamic Optimization** problem to obtain $\tilde{\mathbf{u}}^0$
4. Inject **control action** $u(k) = \tilde{u}^0(0) + u_s(k)$. Predict **successor** state $\hat{x}^{-}(k+1) = A\hat{x}(k) + Bu(k) + B_d\hat{d}(k)$ and disturbance $\hat{d}^{-}(k+1) = \hat{d}(k)$. Set $k \leftarrow k+1$ and go to 1

General formulation of an optimization problem

The three ingredients

- $x \in \mathbb{R}^n$, vector of variables
- $f : \mathbb{R}^n \to \mathbb{R}$, objective function
- $c : \mathbb{R}^n \to \mathbb{R}^m$, vector function of constraints that the variables must satisfy. m is the number of restrictions applied

The optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{subject to} \begin{cases} c_i(x) = 0 & \forall i \in \mathcal{E} \\ c_i(x) \geq 0 & \forall i \in \mathcal{I} \end{cases}$$

$\mathcal{E}, \mathcal{I}$: sets of indices of equality and inequality constraints, respectively

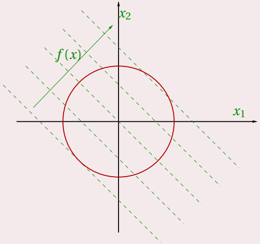
Constrained optimization: example 1

Solve

$$\min\ x_1 + x_2 \qquad \text{s. t. }\ x_1^2 + x_2^2 - 2 = 0$$

Standard notation, feasibility region and solution

- In standard notation: $f(x) = x_1 + x_2$, $\mathcal{I} = \emptyset$, $\mathcal{E} = \{1\}$, $c_1(x) = x_1^2 + x_2^2 - 2$
- Feasibility region: circle of radius $\sqrt{2}$, only the border
- Solution: $x^* = [-1, -1]^T$



Observation

$$\nabla f(x^*) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \nabla c_1(x^*) = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \Longrightarrow \nabla f(x^*) = -\frac{1}{2} \nabla c_1(x^*)$$

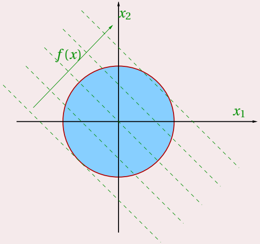
Constrained optimization: example 2

Solve

$$\min\ x_1 + x_2 \qquad \text{s. t.}\ \ 2 - x_1^2 - x_2^2 \geq 0$$

Standard notation, feasibility region and solution

- In standard notation: $f(x) = x_1 + x_2$, $\mathscr{I} = \{1\}$, $\mathscr{E} = \emptyset$, $c_1(x) = 2 - (x_1^2 + x_2^2)$
- Feasibility region: circle of radius $\sqrt{2}$, including the interior
- Solution: $x^* = [-1, -1]^T$



Observation

$$\nabla f(x^*) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \nabla c_1(x^*) = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \Longrightarrow \nabla f(x^*) = \frac{1}{2} \nabla c_1(x^*)$$

Constrained optimality conditions (KKT)

Lagrangian function

$$\mathscr{L}(x,\lambda) = f(x) - \sum_{i \in \mathscr{E} \cup \mathscr{I}} \lambda_i c_i(x)$$

Karush-Kuhn-Tucker optimality conditions

- If x^* is a local solution to the standard problem, there exists a vector $\lambda^* \in \mathbb{R}^m$ such that the following conditions hold:

$$\begin{aligned}
\nabla_x \mathscr{L}(x^*,\lambda^*) &= 0 & & \\
c_i(x^*) &= 0 & &\text{for all } i \in \mathscr{E} \\
c_i(x^*) &\geq 0 & &\text{for all } i \in \mathscr{I} \\
\lambda_i^* &\geq 0 & &\text{for all } i \in \mathscr{I} \\
\lambda_i^* c_i(x^*) &= 0 & &\text{for all } i \in \mathscr{E} \cup \mathscr{I}
\end{aligned}$$

- The components of λ^* are called Lagrange multipliers
- Notice that a multiplier is zero when the corresponding constraint is inactive

Linear programs (LP) problems

LP in **standard form**

$$\min_x c^T x \qquad \text{subject to } Ax = b,\ x \geq 0$$

where: $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $\text{rank}(A) = m$

Optimality conditions

- Lagrangian function: $\mathcal{L}(x, \pi, s) = c^T x - \pi^T (Ax - b) - s^T x$
- If x^* is solution of the linear program, then:

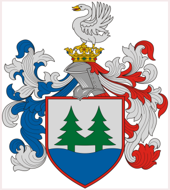
$$\begin{aligned} A^T \pi^* + s^* &= c \\ Ax^* &= b \\ x^* &\geq 0 \\ s^* &\geq 0 \\ x_i^* s_i^* &= 0, \qquad i = 1, 2, \ldots n \end{aligned}$$

LP: the simplex method

The “base points”

A point $x \in \mathbb{R}^n$ is a base point if

- $Ax = b$ and $x \geq 0$
- At most m components of x are nonzero
- The columns of A corresponding to the nonzero elements are linearly independent



Fundamental aspects of the simplex method

- Base points are vertices of the feasibility region
- The solution is a base point
- The simplex method iterates from a base point x_k to another one x_{k+1}, and stops when all components of s_k are nonnegative
- When a component of s_k is negative, a new base point x_{k+1} in which the corresponding element of x_k is nonzero is selected

Quadratic Programming (QP)

Standard form

$$\min_x \frac{1}{2} x^T G x + x^T d$$

subject to:

$$\begin{aligned} a_i^T x &= b_i, \qquad i \in \mathcal{E} \\ a_i^T x &\geq b_i, \qquad i \in \mathcal{I} \end{aligned}$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric (positive definite), $d \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, and $a_i \in \mathbb{R}^n$, for all $i \in \mathcal{E} \cup \mathcal{I}$

The **active set**

$$\mathcal{A}(x^*) = \left\{ i \in \mathcal{E} \cup \mathcal{I} : a_i^T x^* = b_i \right\}$$

Quadratic Programming (QP) problems (2/2)

Lagrangian function

$$\mathscr{L}(x,\lambda) = \frac{1}{2} x^T G x + x^T - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i (a_i^T x - b_i)$$

Optimality conditions (KKT)

$$Gx^* + d - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0$$

$$\begin{aligned} a_i^T x^* &= b_i, && \text{for all } i \in \mathcal{A}(x^*) \\ a_i^T x^* &> b_i, && \text{for all } i \in \mathcal{I} \setminus \mathcal{A}(x^*) \\ \lambda_i^* &\geq 0, && \text{for all } i \in \mathcal{I} \cap \mathcal{A}(x^*) \end{aligned}$$

Active set methods for convex QP problems

Fundamental steps

1. Given a feasible x_k, we evaluate its active set and build the matrix A whose row are $\{a_i^T\}$, $i \in \mathscr{A}(x_k)$
2. Solve the KKT linear system

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p_k \\ \lambda_k \end{bmatrix} = \begin{bmatrix} d + Gx_k \\ Ax_k - b \end{bmatrix}$$

3. If $\|p_k\| \le \rho$, check if $\lambda_i^* \ge 0$ for all $i \in \mathscr{A}(x_k)$. If so, stop.
4. If a multiplier $\lambda_i^* < 0$ for some $i \in \mathscr{A}(x_k)$, remove the i−th constraint from the active set.
5. If $\|p_k\| > \rho$, define $x_{k+1} = x_k + \alpha_k p_k$, where α_k is the largest scalar in $(0, 1]$ such that no inequality constraint is violated. When a blocking constraint is found, it is included in the new active set $\mathscr{A}(x_{k+1})$

Nonlinear programming (NLP) problems via SQP algorithms

Nonlinear programming (NLP) problems

$$
\begin{aligned}
&\min_x f(x) && \text{s.t.} \\
&c_i(x) = 0 && i \in \mathscr{E} \\
&c_i(x) \geq 0 && i \in \mathscr{I}
\end{aligned}
$$

“**Sequential Quadratic Programming**” (SQP) approach

$$
\begin{aligned}
&\min_{p_k} \frac{1}{2} p_k^T W_k p_k + \nabla f(x_k)^T p_k && \text{s.t.} \\
&\nabla c_i(x_k)^T p_k + c_i(x_k) = 0 && i \in \mathscr{E} \\
&\nabla c_i(x_k)^T p_k + c_i(x_k) \geq 0 && i \in \mathscr{I}
\end{aligned}
$$

References I

E. J. Davison and H. W. Smith. Pole assignment in linear time-invariant multivariable systems with constant disturbances. *Automatica*, 7:489–498, 1971.

B. A. Francis and W. M. Wonham. The internal model principle of control theory. *Automatica*, 12: 457–465, 1976.

H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. John Wiley & Sons, 1972.

E. B. Lee and L. Markus. *Foundations of Optimal Control Theory*. John Wiley & Sons, New York, 1967.

P. Lundström, J. H. Lee, M. Morari, and S. Skogestad. Limitations of dynamic matrix control. *Comp. Chem. Eng.*, 19:409–421, 1995.

U. Maeder, F. Borrelli, and M. Morari. Linear offset-free model predictive control. *Automatica*, 45(10): 2214–2222, 2009.

K. R. Muske and T. A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *J. Proc. Contr.*, 12:617–632, 2002.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

G. Pannocchia. Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes. *J. Proc. Cont.*, 13:693–701, 2003.

G. Pannocchia and A. Bemporad. Combined design of disturbance model and observer for offset-free model predictive control. *IEEE Trans. Auto. Contr.*, 52(6):1048–1053, 2007.

G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model predictive control. *AIChE J.*, 49:426–437, 2003.

References II

M. R. Rajamani, J. B. Rawlings, and S. J. Qin. Achieving state estimation equivalence for misassigned disturbances in offset-free model predictive control. *AIChE J.*, 55(2):396–407, 2009.

J. B. Rawlings, E. S. Meadows, and K. R. Muske. Nonlinear model predictive control: a tutorial and survey. In *ADCHEM Conference*, pages 203–214, Kyoto, Japan, 1994.

H. W. Smith and E. J. Davison. Design of industrial regulators. Integral feedback and feedforward control. *Proc. IEE*, 119(8):1210–1216, 1972.