TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Model Predictive Control

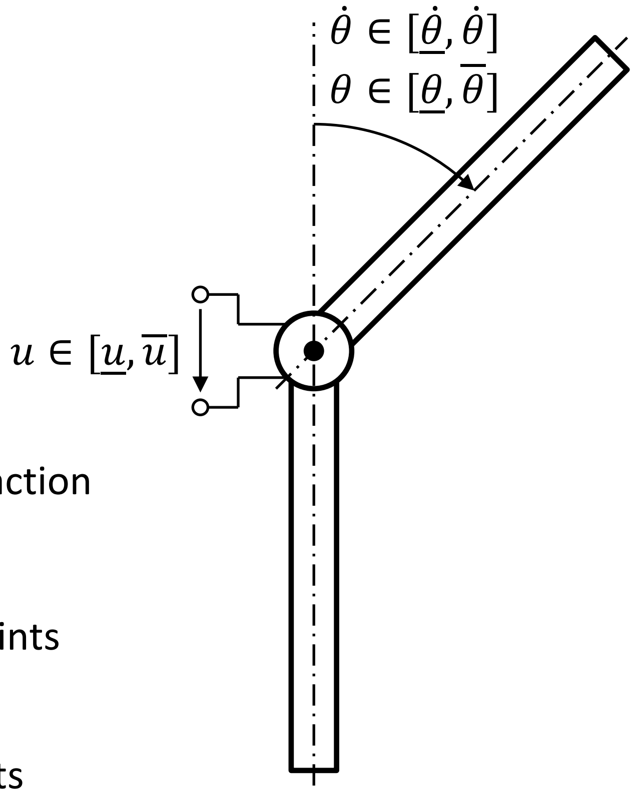
5. Model Predictive Control with Constraints

Jun.-Prof. Dr.-Ing. Daniel Görges
Juniorprofessur für Elektromobilität
Technische Universität Kaiserslautern

Constraints

Types of Constraints

- **All physical systems have constraints!**
- **Physical Constraints**
  - Input constraints, e.g. minimum and maximum voltage u
  - State constraints, e.g. minimum and maximum angle θ
- **Safety Constraints**
  - E.g. minimum and maximum angular velocity $\dot{\theta}$ for human interaction
- **Performance Constraints**
  - Many systems are controlled optimally by exploiting the constraints
  - E.g. minimum positioning time with maximum voltage
  - Performance specifications can partly be expressed as constraints
  - E.g. maximum overshoot



Example Robot Manipulator

Handling Input Constraints

Saturation

- **Basic Idea**
  - Design a control law ignoring the input constraints (e.g. an LQR)
  - Implement the control law using a saturation
- **Control Law**
  - Unconstrained control law $\boldsymbol{u}^{\text{free}}(k)$
  - Saturated control law

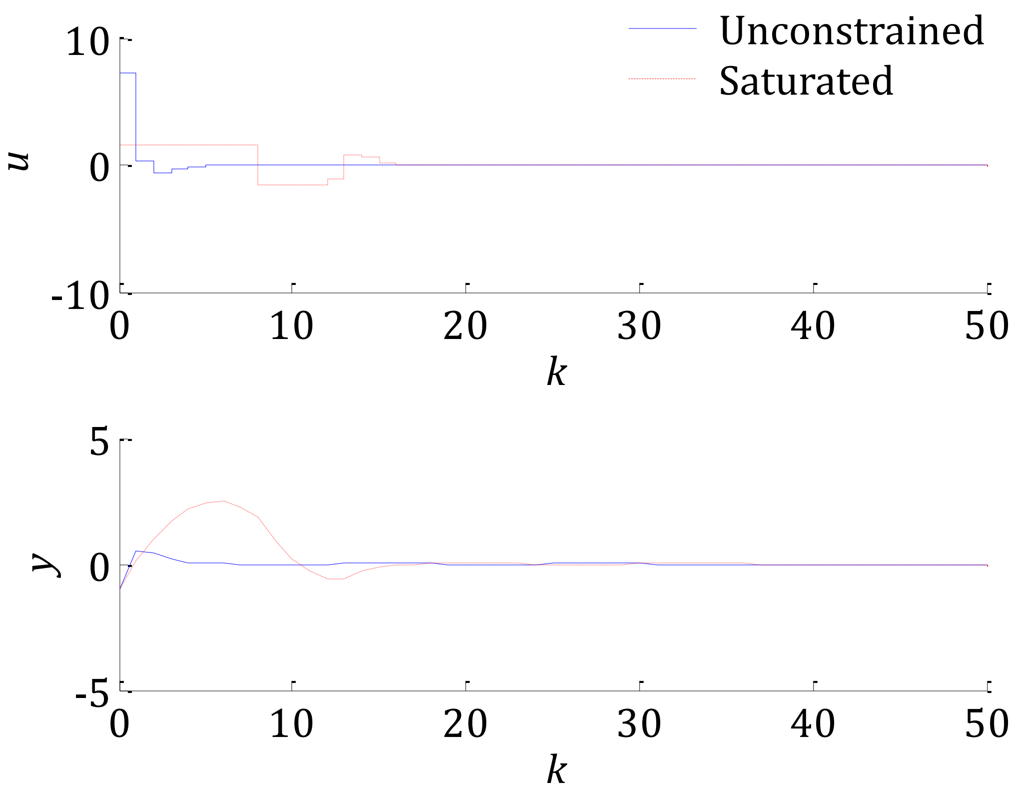
$$u_w(k) = \begin{cases} \underline{u}_w & \text{for } u_w^{\text{free}}(k) < \underline{u}_w \\ u_w^{\text{free}}(k) & \text{for } \underline{u}_w \leq u_w^{\text{free}}(k) \leq \overline{u}_w \\ \overline{u}_w & \text{for } \overline{u}_w < u_w^{\text{free}}(k) \end{cases}, \quad w \in \{1, \dots, m\}$$

- **Properties**
  - Response often poor and oscillatory
  - Closed-loop stability not guaranteed

Handling Input Constraints

Saturation

- **Illustrative Example**



Example from Chapter 4

$\boldsymbol{x}(0) = (0.5 \quad -0.5)^T$

$y(k) = (-1 \quad 1)\boldsymbol{x}(k)$

Constraint $-1.5 \leq u(k) \leq 1.5$

Input weight $R = 0.01$

LQR $u^{\text{free}}(k) = \boldsymbol{K}_{\text{LQR}}\boldsymbol{x}(k)$

Response poor and oscillatory

Unstable for $-0.5 \leq u(k) \leq 0.5$

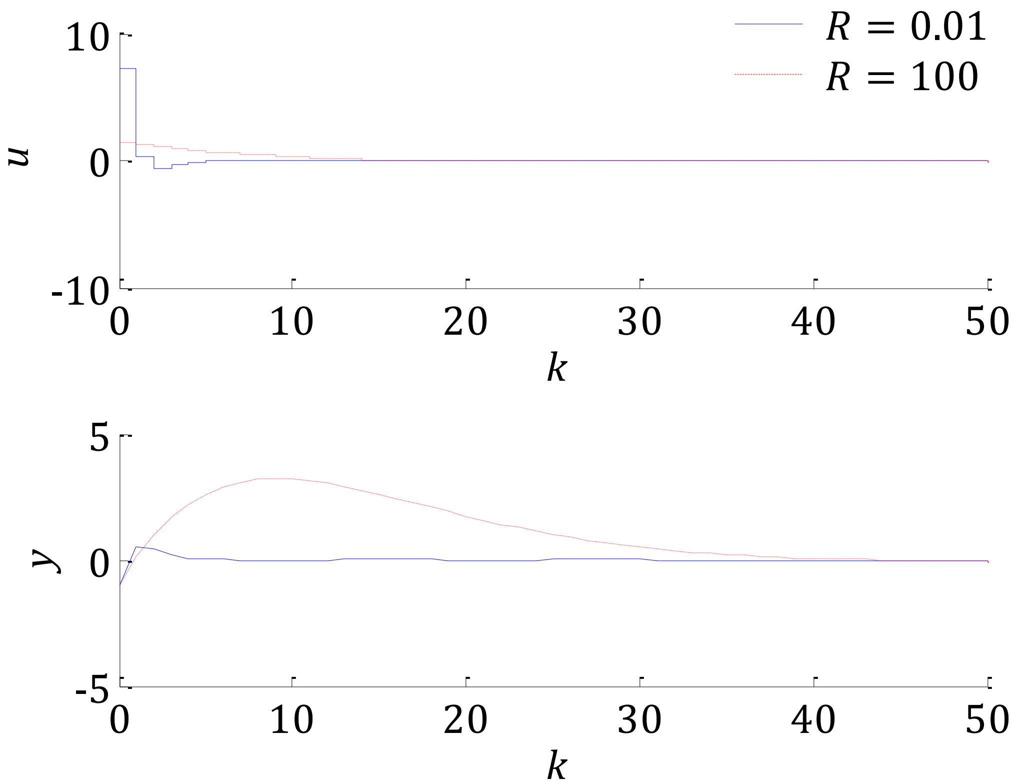
Handling Input Constraints

De-Tuned Optimal Control

- **Basic Idea**
  - Design an LQR
  - Increase the input weighting matrix $\boldsymbol{R}$ until the input constraints are satisfied
- **Control Law**
  - LQR $\boldsymbol{u}^*(k) = \boldsymbol{K}_{\text{LQR}}\boldsymbol{x}(k)$
- **Properties**
  - Response often very slow
  - Closed-loop stability guaranteed but often only of theoretical value

De-Tuned Optimal Control

- **Illustrative Example**



Example from Chapter 4

$\boldsymbol{x}(0) = (0.5 \quad -0.5)^T$

$y(k) = (-1 \quad 1)\boldsymbol{x}(k)$

Constraint $-1.5 \leq u(k) \leq 1.5$

LQR $u(k) = \boldsymbol{K}_{\mathrm{LQR}}\boldsymbol{x}(k)$ $(R = 100)$

Response very slow

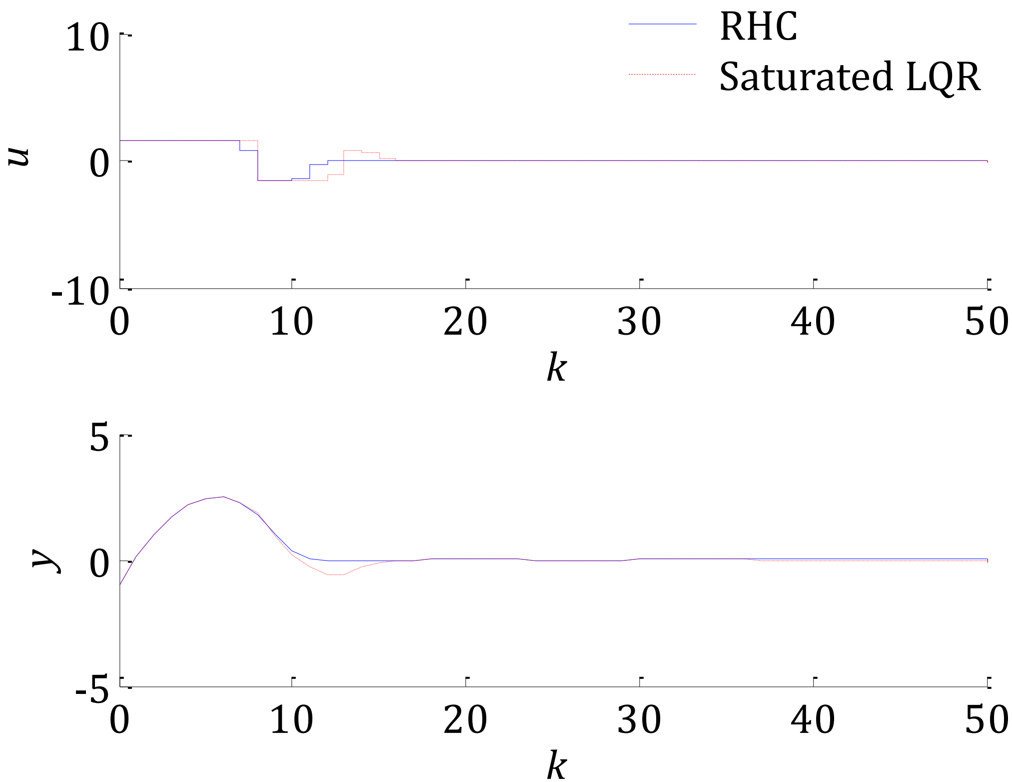
Handling Input Constraints

Anti-Windup Strategies

- **Motivation**
  - Controllers with integral action incur integrator windup when the input constraints are active
  - The integrator continues integrating despite the input constraints being active
  - The integral must therefore be reduced first when the control error changes sign
  - This can make the response very slow and even lead to instability
- **Basic Idea**
  - Stop integrating when the input constraints are active
- **Control Law**
  - Various anti-windup strategies available, cf. Lineare Regelungen and [ÅW90, Section 8.3]
- **Properties**
  - Response usually better and less oscillatory than for pure saturation
  - Closed-loop stability usually not guaranteed

Receding Horizon Control

- **Illustrative Example**



Example from Chapter 4

$\boldsymbol{x}(0) = (0.5 \quad -0.5)^T$

$y(k) = (-1 \quad 1)\boldsymbol{x}(k)$

Constraint $-1.5 \leq u(k) \leq 1.5$

Input weight $R = 0.01$

RHC (prediction horizon $N = 16$)

Response very good

Closed-loop stability guaranteed (using the methods in Chapter 6)

Optimization Problem

Problem 5.1 For the discrete-time linear time-invariant system (4.1) and the current state $\boldsymbol{x}(k)$ find an input sequence $\boldsymbol{U}^*(k)$ such that the discrete-time quadratic cost function (4.3) is minimized, i.e.

$$\min_{\boldsymbol{U}(k)} V_N(\boldsymbol{x}(k), \boldsymbol{U}(k))$$

$$\text{subject to} \begin{cases} \boldsymbol{x}(k+i+1) = \boldsymbol{A}\boldsymbol{x}(k+i) + \boldsymbol{B}\boldsymbol{u}(k+i), i = 0,1,\dots,N-1 \\ \boldsymbol{x}(k+i) \in \mathbb{X}(k+i) \subseteq \mathbb{R}^n,\ \ i = 1,2,\dots,N \\ \boldsymbol{u}(k+i) \in \mathbb{U}(k+i) \subseteq \mathbb{R}^m, i = 0,1,\dots,N-1 \end{cases}$$

- **Remarks**
  - Problem 5.1 corresponds to Problem 4.1 except the constraints
  - The prediction model (4.4) and the cost function in matrix form (4.5) can thus still be utilized
  - We only need to concentrate on the constraint model
  - Problem 5.1 can then be solved in a “batch” way using quadratic programming
  - Note that a numerical solution is required in the constrained case

Constraint Model

- **Standard Form**

$$\boldsymbol{M}(k+i)\boldsymbol{x}(k+i) + \boldsymbol{E}(k+i)\boldsymbol{u}(k+i) \leq \boldsymbol{b}(k+i), \quad i = 0,1,\dots,N-1$$
$$\boldsymbol{M}(k+N)\boldsymbol{x}(k+N) \leq \boldsymbol{b}(k+N)$$

- **Special Forms**

$\boldsymbol{M}(k+i) = \boldsymbol{0}\ \forall i \in \{0,1,\dots,N\}\ \forall k \in \mathbb{N}_0$ → input constraints only

$\boldsymbol{E}(k+i) = \boldsymbol{0}\ \forall i \in \{0,1,\dots,N-1\}\ \forall k \in \mathbb{N}_0$ → state constraints only

- **Remarks**
  - The constraints in standard and special form can depend on the absolute time k and relative time i
  - The constraints in standard form can describe a coupling between input and state constraints
  - Note that due to the coupling also the state constraints at time k must be considered
  - For simplicity a coupling between input and state constraints is not considered in Problem 5.1
  - Problem 5.1 can, however, be reformulated w.r.t. a coupling between input and state constraints

Constraint Model

- **Representation in Matrix Form**

$$\underbrace{\begin{pmatrix} \boldsymbol{M}(k) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}}_{\boldsymbol{\mathcal{D}}(k)} \boldsymbol{x}(k) + \underbrace{\begin{pmatrix} \mathbf{0} & \cdots & \mathbf{0} \\ \boldsymbol{M}(k+1) & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{M}(k+N) \end{pmatrix}}_{\boldsymbol{\mathcal{M}}(k)} \underbrace{\begin{pmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{pmatrix}}_{\boldsymbol{X}(k)} + \underbrace{\begin{pmatrix} \boldsymbol{E}(k) & \cdots & \mathbf{0} \\ \vdots & \cdots & \vdots \\ \mathbf{0} & \ddots & \boldsymbol{E}(k+N-1) \\ \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix}}_{\boldsymbol{\mathcal{E}}(k)} \underbrace{\begin{pmatrix} \boldsymbol{u}(k) \\ \boldsymbol{u}(k+1) \\ \vdots \\ \boldsymbol{u}(k+N-1) \end{pmatrix}}_{\boldsymbol{U}(k)} \leq \underbrace{\begin{pmatrix} \boldsymbol{b}(k) \\ \boldsymbol{b}(k+1) \\ \vdots \\ \boldsymbol{b}(k+N) \end{pmatrix}}_{\boldsymbol{\mathcal{b}}(k)} \quad (5.1)$$

- **Substitution of the Prediction Model** $\boldsymbol{X}(k) = \boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Gamma}\boldsymbol{U}(k)$ **(4.4)**

$$\begin{aligned}
&\boldsymbol{\mathcal{D}}(k)\boldsymbol{x}(k) + \boldsymbol{\mathcal{M}}(k)\big(\boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Gamma}\boldsymbol{U}(k)\big) + \boldsymbol{\mathcal{E}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) && \Leftrightarrow \\
&(\boldsymbol{\mathcal{D}}(k) + \boldsymbol{\mathcal{M}}(k)\boldsymbol{\Phi})\boldsymbol{x}(k) + \big(\boldsymbol{\mathcal{M}}(k)\boldsymbol{\Gamma} + \boldsymbol{\mathcal{E}}(k)\big)\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) && \Leftrightarrow \\
&\underbrace{\big(\boldsymbol{\mathcal{M}}(k)\boldsymbol{\Gamma} + \boldsymbol{\mathcal{E}}(k)\big)}_{\boldsymbol{\mathcal{A}}(k)}\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) + \underbrace{\big(-\boldsymbol{\mathcal{D}}(k) - \boldsymbol{\mathcal{M}}(k)\boldsymbol{\Phi}\big)}_{\boldsymbol{\mathcal{W}}(k)}\boldsymbol{x}(k) && \Leftrightarrow \\
&\boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k)
\end{aligned}$$

Box Constraints

- **Constraint Model**

$$\underline{\boldsymbol{u}}(k+i) \leq \boldsymbol{u}(k+i) \leq \overline{\boldsymbol{u}}(k+i),\ \ i = 0,1,\dots,N-1$$

$$\underline{\boldsymbol{y}}(k+i) \leq \boldsymbol{y}(k+i) \leq \overline{\boldsymbol{y}}(k+i),\ \ i = 0,1,\dots,N$$

$$\boldsymbol{y}(k+i) = \boldsymbol{C}\boldsymbol{x}(k+i)$$

$$\underline{\boldsymbol{u}}(k+i) \leq \boldsymbol{u}(k+i) \Leftrightarrow -\boldsymbol{u}(k+i) \leq -\underline{\boldsymbol{u}}(k+i)$$

$$\underline{\boldsymbol{y}}(k+i) \leq \boldsymbol{y}(k+i) \Leftrightarrow -\boldsymbol{y}(k+i) \leq -\underline{\boldsymbol{y}}(k+i)$$

- **Representation in Standard Form**

$$\underbrace{\begin{pmatrix} \boldsymbol{0}_{m\times n} \\ \boldsymbol{0}_{m\times n} \\ -\boldsymbol{C} \\ +\boldsymbol{C} \end{pmatrix}}_{\boldsymbol{M}(k+i)} \boldsymbol{x}(k+i) + \underbrace{\begin{pmatrix} -\boldsymbol{I}_{m\times m} \\ +\boldsymbol{I}_{m\times m} \\ \boldsymbol{0}_{p\times m} \\ \boldsymbol{0}_{p\times m} \end{pmatrix}}_{\boldsymbol{E}(k+i)} \boldsymbol{u}(k+i) \leq \underbrace{\begin{pmatrix} -\underline{\boldsymbol{u}}(k+i) \\ +\overline{\boldsymbol{u}}(k+i) \\ -\underline{\boldsymbol{y}}(k+i) \\ +\overline{\boldsymbol{y}}(k+i) \end{pmatrix}}_{\boldsymbol{b}(k+i)},\ \ i = 0,1,\dots,N-1$$

$$\underbrace{\begin{pmatrix} -\boldsymbol{C} \\ +\boldsymbol{C} \end{pmatrix}}_{\boldsymbol{M}(k+N)} \boldsymbol{x}(k+N) \leq \underbrace{\begin{pmatrix} -\underline{\boldsymbol{y}}(k+N) \\ +\overline{\boldsymbol{y}}(k+N) \end{pmatrix}}_{\boldsymbol{b}(N)}$$

Rate Constraints

- **Constraint Model**

$$\Delta\underline{\boldsymbol{u}}(k+i) \leq \boldsymbol{u}(k+i) - \boldsymbol{u}(k+i-1) \leq \Delta\overline{\boldsymbol{u}}(k+i),\ \ i = 1,2,\dots,N-1$$

- **Representation in Standard Form**

$$\underbrace{\begin{pmatrix} +\boldsymbol{I}_{m\times m} & -\boldsymbol{I}_{m\times m} & \boldsymbol{0}_{m\times m} & \boldsymbol{0}_{m\times m} & \cdots & \boldsymbol{0}_{m\times m} & \boldsymbol{0}_{m\times m} \\ -\boldsymbol{I}_{m\times m} & +\boldsymbol{I}_{m\times m} & \boldsymbol{0}_{m\times m} & \boldsymbol{0}_{m\times m} & \cdots & \boldsymbol{0}_{m\times m} & \boldsymbol{0}_{m\times m} \\ \boldsymbol{0}_{m\times m} & +\boldsymbol{I}_{m\times m} & -\boldsymbol{I}_{m\times m} & \boldsymbol{0}_{m\times m} & \cdots & \boldsymbol{0}_{m\times m} & \boldsymbol{0}_{m\times m} \\ \boldsymbol{0}_{m\times m} & -\boldsymbol{I}_{m\times m} & +\boldsymbol{I}_{m\times m} & \boldsymbol{0}_{m\times m} & \cdots & \boldsymbol{0}_{m\times m} & \boldsymbol{0}_{m\times m} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \end{pmatrix}}_{\boldsymbol{\mathcal{E}}(k)} \underbrace{\begin{pmatrix} \boldsymbol{u}(k) \\ \boldsymbol{u}(k+1) \\ \boldsymbol{u}(k+2) \\ \vdots \end{pmatrix}}_{\boldsymbol{U}(k)} \leq \underbrace{\begin{pmatrix} -\Delta\underline{\boldsymbol{u}}(k+1) \\ \Delta\overline{\boldsymbol{u}}(k+1) \\ -\Delta\underline{\boldsymbol{u}}(k+2) \\ \Delta\overline{\boldsymbol{u}}(k+2) \\ \vdots \end{pmatrix}}_{\boldsymbol{\mathscr{b}}(k)}$$

- **Remarks**
  - Rate constraints arise e.g. in power plants where the power change is usually limited
  - Rate constraints can be formulated analogously for states and outputs

Performance Constraints

- **Overshoot Constraints**

  $$\boldsymbol{y}(k+i) \leq \boldsymbol{r}(k_\mathrm{s}), i = k_\mathrm{s}, \dots, k_\mathrm{e}$$

  where $\boldsymbol{r}(k_\mathrm{s})$ is the reference input and $k_\mathrm{s} \geq 1$ and $k_\mathrm{e} \leq N$ are the start and end of the transient

  – Representation in standard form analogous to box constraints

- **Monotonic Behavior**

  $$\boldsymbol{y}(k+i) \leq \boldsymbol{y}(k+i+1) \text{ if } \boldsymbol{y}(k) < \boldsymbol{r}(k), i = 1, \dots, N-1$$
  $$\boldsymbol{y}(k+i) \geq \boldsymbol{y}(k+i+1) \text{ if } \boldsymbol{y}(k) > \boldsymbol{r}(k), i = 1, \dots, N-1$$

  where $\boldsymbol{r}(k)$ is the reference input

  – Constraints on monotonic behavior prevent oscillations

  – Representation in standard form analogous to rate constraints

Finite Horizon Control

Performance Constraints

- **Non-Minimum Phase Behavior**

$$\boldsymbol{y}(k+i) \geq \boldsymbol{y}(k) \text{ if } \boldsymbol{y}(k) < \boldsymbol{r}(k), i = 1, \dots, N$$

$$\boldsymbol{y}(k+i) \leq \boldsymbol{y}(k) \text{ if } \boldsymbol{y}(k) > \boldsymbol{r}(k), i = 1, \dots, N$$

where $\boldsymbol{r}(k)$ is the reference input

  – Constraints on non-minimum phase behavior prevent movement in the opposite direction

  – Representation in standard form analogous to rate constraints

- **Remark**

  – Note that also nonlinear effects like dead zones can be handled by constraints

  – More details and further references are given in [CB04, Section 7.1]

Technische Universität Kaiserslautern

Finite Horizon Control

Optimization Problem (Cont'd)

- **Representation in Matrix Form using (4.5)**

Term is independent of $\boldsymbol{U}(k)$

Term is therefore not relevant!

$$\min_{\boldsymbol{U}(k)} \frac{1}{2}\boldsymbol{U}^T(k)\boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{U}^T(k)\boldsymbol{F}\boldsymbol{x}(k) + \cancel{\boldsymbol{x}^T(k)(\boldsymbol{Q}+\boldsymbol{\Phi}^T\boldsymbol{\Omega}\boldsymbol{\Phi})\boldsymbol{x}(k)}$$

$$\text{subject to } \boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k)$$

The current state $\boldsymbol{x}(k)$ occurs here!

- **Solution based on Quadratic Programming**
  - The representation in matrix form can be easily written as a quadratic program (cf. Slide 3-25)

$$\min_{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{f}^T\boldsymbol{\theta}$$

$$\text{subject to } \boldsymbol{A}_{\text{ieq}}\boldsymbol{\theta} \leq \boldsymbol{b}_{\text{ieq}}$$

by setting $\quad \boldsymbol{\theta} \coloneqq \boldsymbol{U}(k), \quad \boldsymbol{H} \coloneqq \boldsymbol{H}, \quad \boldsymbol{f} \coloneqq \boldsymbol{F}\boldsymbol{x}(k), \quad \boldsymbol{A}_{\text{ieq}} \coloneqq \boldsymbol{\mathcal{A}}(k), \quad \boldsymbol{b}_{\text{ieq}} \coloneqq \boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k)$

  - The quadratic program is convex iff $\boldsymbol{H} \succcurlyeq \boldsymbol{0}$. The solution $\boldsymbol{U}^*(k)$ is then a global minimizer
  - The quadratic program is strictly convex iff $\boldsymbol{H} \succ \boldsymbol{0}$. The solution $\boldsymbol{U}^*(k)$ is then a unique global minim.

Jun.-Prof. Dr.-Ing. Daniel Görges
Juniorprofessur für Elektromobilität
Technische Universität Kaiserslautern

Optimization Problem (Cont'd)

- **Solution based on Quadratic Programming**
  - The solution $\boldsymbol{U}^*(k)$ can be determined under MATLAB using

    $\boldsymbol{U}^*(k)$ = `quadprog(`$\boldsymbol{H}$`,`$\boldsymbol{F} * \boldsymbol{x}(k)$`,`$\boldsymbol{\mathcal{A}}(k)$`,`$\boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k) * \boldsymbol{x}(k)$`)`

- **Remark**
  - From a computation perspective substituting the prediction model (4.4) into the cost function (4.5) and the constraint model (5.1) may not be beneficial
  - The quadratic programming problem can alternatively be formulated with the cost function (4.5), the prediction model (4.4) as equality constraint, and the constraint model (5.1) as inequality constraint
  - This will on the one hand increase the number of decision variables and constraints (bad)
  - This will on the other hand render the matrices $\boldsymbol{H}$ and $\boldsymbol{A}_{\mathrm{ieq}}$ banded which considerably speeds up the decomposition used in the active set method (cf. Slide 3-29) and in the interior point method (good)
  - A detailed discussion can be found in [Mac02, Section 3.3]

Receding Horizon Controller

- **Optimal Control Law**

$$\boldsymbol{u}^*(k) = (\boldsymbol{I}_{m\times m} \quad \boldsymbol{0}_{m\times m} \quad \cdots \quad \boldsymbol{0}_{m\times m})\boldsymbol{U}^*(k) \tag{5.2}$$

- **Remarks**
  - It can be shown that $\boldsymbol{U}^*(k)$ is a nonlinear function of $\boldsymbol{x}(k)$, cf. [BBM15, Sec. 12.3], [Mac02, Sec. 3.2.2]
  - A receding horizon controller is hence a nonlinear state feedback controller in the constrained case
  - The optimal input sequence must $\boldsymbol{U}^*(k)$ must be calculated online in the constrained case
    - The online optimization can be very time-consuming
    - The online optimization must, however, be finished within the sampling period h
    - Receding horizon control has therefore been limited to slow systems for many years
    - Receding horizon control has increasingly been applied to fast system in recent years, primarily due to advances in computer hardware and model predictive control algorithms
    - The online optimization can partly be moved to an offline optimization, leading to explicit model predictive control as detailed in [BBM15, Section 12.3]

Warm Starting

- **Motivation**
  - The active set method allows using an initial guess for reducing the computation time (cf. Slide 3-31)
- **Approach**
  - Consider that at time k the optimal input sequence $\boldsymbol{U}^*(k)$ has been computed
  - At time $k+1$ a good initial guess is then the "shifted" optimal input sequence $\widetilde{\boldsymbol{U}}(k+1)$, i.e.

$$\boldsymbol{U}^*(k) = \Big(\underbrace{\boldsymbol{u}^{*T}(k)}_{\text{implemented}} \quad \boldsymbol{u}^{*T}(k+1) \quad \boldsymbol{u}^{*T}(k+2) \quad \cdots \quad \boldsymbol{u}^{*T}(k+N-2) \quad \boldsymbol{u}^{*T}(k+N-1)\Big)^T$$

$$\widetilde{\boldsymbol{U}}(k+1) = \Big(\underbrace{\boldsymbol{u}^{*T}(k+1) \quad \boldsymbol{u}^{*T}(k+2) \quad \cdots \quad \boldsymbol{u}^{*T}(k+N-2) \quad \boldsymbol{u}^{*T}(k+N-1)}_{\text{feasible (by definition)}} \quad \underbrace{\boldsymbol{u}^{T}(k+N)}_{\text{possibly infeasible}}\Big)^T$$

  - Choose $\boldsymbol{u}(k+N)$ such that

$$\boldsymbol{u}(k+N) \in \mathbb{U}(k+N) \tag{5.3}$$

$$\boldsymbol{x}(k+N+1) = \boldsymbol{A}\boldsymbol{x}^*(k+N) + \boldsymbol{B}\boldsymbol{u}(k+N) \in \mathbb{X}(k+N+1) \tag{5.4}$$

Warm Starting

- **Approach**
  - Choosing $\boldsymbol{u}(k+N)$ such that (5.3), (5.4) are fulfilled requires the construction of an admissible set, see Definition 6.2 and Slide 6-14 for details and ideas
- **Remarks**
  - The computation time can usually not be reduced using an initial guess if there are large disturbances or large reference changes. Then the worst-case computation time must be considered.
  - The solution $\boldsymbol{U}^*(k)$ can be determined considering the initial guess $\widetilde{\boldsymbol{U}}(k)$ under MATLAB using

    $\boldsymbol{U}^*(k)$ = `quadprog(`$\boldsymbol{H}$`,`$\boldsymbol{F} * \boldsymbol{x}(k)$`,`$\boldsymbol{\mathcal{A}}(k)$`,`$\boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k) * \boldsymbol{x}(k)$`,[],[],[],[],`$\widetilde{\boldsymbol{U}}(k)$`)`

Multiple Horizons

- **Motivation**
  - The computation time depends on the number of decision variables and constraints
  - This observation led to the concept of multiple horizons
- **Approach**
  - Modify Problem 5.1 to

$$\min_{\boldsymbol{U}(k)} V_N(\boldsymbol{x}(k), \boldsymbol{U}(k))$$

$$\text{subject to} \begin{cases} \boldsymbol{x}(k+i+1) = \boldsymbol{A}\boldsymbol{x}(k+i) + \boldsymbol{B}\boldsymbol{u}(k+i), i = 0,1,\dots,N-1 \\ \boldsymbol{x}(k+i) \in \mathbb{X}(k+i) \subseteq \mathbb{R}^n,\ \ i = 1,2,\dots,N_{\boldsymbol{x}} \\ \boldsymbol{u}(k+i) \in \mathbb{U}(k+i) \subseteq \mathbb{R}^m, i = 0,1,\dots,N_{\boldsymbol{u}} \\ \boldsymbol{u}(k+i) = \boldsymbol{K}\boldsymbol{x}(k+i), \qquad i = N_{\mathrm{c}},\dots,N-1 \end{cases}$$

with the state constraint horizon $N_{\boldsymbol{x}} \leq N$, the input constraint horizon $N_{\boldsymbol{u}} \leq N-1$, the control horizon $N_{\mathrm{c}} \leq N-1$, and some feedback matrix $\boldsymbol{K}$ (e.g. $\boldsymbol{K}_{\mathrm{LQR}}$)

Extensions

Multiple Horizons

- **Approach**
  - By selecting $N_c < N - 1$ the number of decision variables can essentially be reduced
  - By selecting $N_x < N$ and $N_u < N - 1$ the number of constraints can be reduced
- **Remarks**
  - The performance is reduced for $N_c < N - 1$ since the degrees of freedom are reduced
  - The state and input constraints are not ensured for $N_x < N$ and $N_u < N - 1$
    The constraints are, however, often not violated at the end of the prediction horizon (cf. Slide 6-28)
  - The stability conditions in Chapter 6 are not applicable or must be modified for multiple horizons
  - Another approach for reducing the computation time consists in move blocking, i.e.

    $$\boldsymbol{u}(k+i) = \boldsymbol{u}(k+i-1), i = N_c, \dots, N-1$$

    whereby the number of decision variables can essentially be reduced
  - More details and further references are given in [BBM15, Section 13.5] and [Mac02, Section 2.2]

Scaling

- **Motivation**
  - The magnitudes of the states and inputs can differ significantly
  - This can render Problem 5.1 ill-conditioned
- **Approach**
  - Consider that the magnitudes of the states and inputs are characterized by

$$x_v(k+i) \in [\underline{x}_v, \overline{x}_v], v \in \{1, \dots, n\}, \qquad u_w(k+i) \in [\underline{u}_w, \overline{u}_w], w \in \{1, \dots, m\}$$

  - Introduce the state and input scaling matrix

$$\boldsymbol{S_x} = \operatorname{diag}\left(\frac{1}{\max(|\underline{x}_1|,|\overline{x}_1|)}, \dots, \frac{1}{\max(|\underline{x}_n|,|\overline{x}_n|)}\right), \quad \boldsymbol{S_u} = \operatorname{diag}\left(\frac{1}{\max(|\underline{u}_1|,|\overline{u}_1|)}, \dots, \frac{1}{\max(|\underline{u}_m|,|\overline{u}_m|)}\right)$$

where $\operatorname{diag}(\cdot)$ denotes a diagonal matrix

  - Introduce the scaled state and input vector

$$\tilde{\boldsymbol{x}}(k) = \boldsymbol{S_x}\boldsymbol{x}(k) \Leftrightarrow \boldsymbol{x}(k) = \boldsymbol{S_x}^{-1}\tilde{\boldsymbol{x}}(k), \qquad \tilde{\boldsymbol{u}}(k) = \boldsymbol{S_u}\boldsymbol{u}(k) \Leftrightarrow \boldsymbol{u}(k) = \boldsymbol{S_u}^{-1}\tilde{\boldsymbol{u}}(k)$$

Extensions

Scaling

- **Approach**
  - This leads to the scaled discrete-time linear time-invariant state equation

$$\boldsymbol{S}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}(k+i+1) = \boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}(k+i) + \boldsymbol{B}\boldsymbol{S}_{\boldsymbol{u}}^{-1}\tilde{\boldsymbol{u}}(k+i) \Leftrightarrow$$

$$\tilde{\boldsymbol{x}}(k+i+1) = \boldsymbol{S}_{\boldsymbol{x}}\boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}(k+i) + \boldsymbol{S}_{\boldsymbol{x}}\boldsymbol{B}\boldsymbol{S}_{\boldsymbol{u}}^{-1}\tilde{\boldsymbol{u}}(k+i) = \widetilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}(k+i) + \widetilde{\boldsymbol{B}}\tilde{\boldsymbol{u}}(k+i),$$

  the scaled discrete-time quadratic cost function

$$\begin{aligned}\tilde{V}_N\left(\tilde{\boldsymbol{x}}(k), \widetilde{\boldsymbol{U}}(k)\right) &= \tilde{\boldsymbol{x}}^T(k+N)\boldsymbol{S}_{\boldsymbol{x}}^{-T}\boldsymbol{P}\boldsymbol{S}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}(k+N) + \sum_{i=0}^{N-1}\tilde{\boldsymbol{x}}^T(k+i)\boldsymbol{S}_{\boldsymbol{x}}^{-T}\boldsymbol{Q}\boldsymbol{S}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}(k+i) + \tilde{\boldsymbol{u}}^T(k+i)\boldsymbol{S}_{\boldsymbol{u}}^{-T}\boldsymbol{R}\boldsymbol{S}_{\boldsymbol{u}}^{-1}\tilde{\boldsymbol{u}}(k+i) \\ &= \tilde{\boldsymbol{x}}^T(k+N)\widetilde{\boldsymbol{P}}\tilde{\boldsymbol{x}}(k+N) + \sum_{i=0}^{N-1}\tilde{\boldsymbol{x}}^T(k+i)\widetilde{\boldsymbol{Q}}\tilde{\boldsymbol{x}}(k+i) + \tilde{\boldsymbol{u}}^T(k+i)\widetilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}(k+i),\end{aligned}$$

  the scaled state and input constraints

$$\boldsymbol{S}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}(k+i) \in \mathbb{X}(k+i) \subseteq \mathbb{R}^n, i = 1,2,\dots,N, \quad \boldsymbol{S}_{\boldsymbol{u}}^{-1}\tilde{\boldsymbol{u}}(k+i) \in \mathbb{U}(k+i) \subseteq \mathbb{R}^m, i = 0,1,\dots,N-1$$

  - Problem 5.1 is then formulated w.r.t the scaled state equation, cost function, and constraints
  - Note that the constraints in standard form can be scaled analogously

Linear Cost Function

- **Discrete-Time Linear Cost Function**

$$V_N\big(\boldsymbol{x}(k), \boldsymbol{U}(k)\big) = \|\boldsymbol{P}\boldsymbol{x}(k+N)\|_p + \sum_{i=0}^{N-1} \|\boldsymbol{Q}\boldsymbol{x}(k+i)\|_p + \|\boldsymbol{R}\boldsymbol{u}(k+i)\|_p \quad \text{with} \quad p \in \{1, \infty\} \qquad (5.5)$$

- **Symbols**
  - $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ full rank — state weighting matrix
  - $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ full rank — input weighting matrix
  - $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ full rank — terminal weighting matrix
  - $\|\boldsymbol{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$ — 1-norm or sum norm
  - $\|\boldsymbol{x}\|_\infty = \max(|x_1|, |x_2|, \dots, |x_n|)$ — ∞-norm or maximum norm

- **Remarks**
  - Problem 5.1 with linear cost function (5.5) can be formulated as a linear programming problem
  - For this purpose the "trick" $\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x}\|_1 \Leftrightarrow \min_{\boldsymbol{x},\boldsymbol{\gamma}\in\mathbb{R}^n} (\boldsymbol{I} \quad \boldsymbol{0}) \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{x} \end{pmatrix}$ subject to $\boldsymbol{\gamma} \geq \boldsymbol{x}, \boldsymbol{\gamma} \geq -\boldsymbol{x}$ is used

Linear Cost Function

Linear Cost Function

- Linear program (computation time smaller)
- More constraints (computation time larger)
- Interpretation of the cost function less intuitive
- Optimal input sequence $\boldsymbol{U}^*(k)$ usually on the intersection of the constraints and possibly not unique (cf. Slide 3-24)
- Leads to non-smooth behavior
- Makes tuning difficult

Quadratic Cost Function

- Quadratic program (computation time larger)
- Less constraints (computation time smaller)
- Interpretation of the cost function more intuitive
- Optimal input sequence $\boldsymbol{U}^*(k)$ generally inside or on boundary of feasible set and unique for $\boldsymbol{H} \succ \boldsymbol{0}$ (cf. Slide 3-26)
- Leads to smooth behavior
- Makes tuning simple
- Connection to linear-quadratic control theory

More details and further references are given in [Mac02, Section 5.4] and [BBM15, Section 13.5]

Extensions

Soft Constraints

- **Motivation**
  - Problem 5.1 can become infeasible
  - Reasons for infeasibility are large disturbances, large uncertainties (mismatch between prediction model and physical system), wrong RHC formulations (e.g. prediction horizon too small), etc.
  - Input constraints are usually "hard" (e.g. maximum voltages in robot control)
  - State constraints are sometimes "soft" (e.g. temperatures in building climate control)
  - This observation led to the concept of soft constraints to handle infeasibility
- **Quadratic Penalty**

$$\min_{\boldsymbol{U}(k),\boldsymbol{\varepsilon}(k)} \frac{1}{2}\boldsymbol{U}^T(k)\boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{U}^T(k)\boldsymbol{F}\boldsymbol{x}(k) + \rho\|\boldsymbol{\varepsilon}(k)\|_2^2$$

$$\text{subject to } \boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k) + \boldsymbol{\varepsilon}(k),\ \boldsymbol{\varepsilon}(k) \geq \boldsymbol{0}$$

where $\boldsymbol{\varepsilon}(k)$ is a non-negative vector with $\dim \boldsymbol{\varepsilon}(k) = \dim \boldsymbol{\mathcal{b}}(k)$ and ρ is a non-negative scalar

Extensions

Soft Constraints

- **Quadratic Penalty**
  - The optimization problem remains a quadratic program with additional variables and constraints
  - $\rho = 0$ leads to the unconstrained problem, $\rho \to \infty$ to the hard-constrained problem
  - ρ finite allows a constraint violation (unfortunately also if a feasible solution exists)
- **Linear Penalty**

$$\min_{\boldsymbol{U}(k),\boldsymbol{\varepsilon}(k)} \frac{1}{2}\boldsymbol{U}^T(k)\boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{U}^T(k)\boldsymbol{F}\boldsymbol{x}(k) + \rho\|\boldsymbol{\varepsilon}(k)\|_p$$

$$\text{subject to } \boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \le \boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k) + \boldsymbol{\varepsilon}(k),\ \boldsymbol{\varepsilon}(k) \ge \boldsymbol{0}$$

where $\boldsymbol{\varepsilon}(k)$ is a non-neg. vector with $\dim \boldsymbol{\varepsilon}(k) = \dim \boldsymbol{\mathcal{b}}(k)$, ρ is a non-neg. scalar, and $p \in \{1, \infty\}$

  - The optimization problem remains a quadratic program with additional variables and constraints
  - $\rho = 0$, $\rho \to \infty$, and ρ finite have the same effect as for a quadratic penalty
  - ρ finite but large enough does, however, not lead to a constraint violation if a feasible solution exists

Extensions

Soft Constraints

- **Remarks**
  - To reduce the number of variables and therefore the computation time a non-negative scalar ε and a non-negative weighting vector $\boldsymbol{\mathscr{b}}^{\mathrm{p}}(k)$ quantifying the importance of the constraints can be used

$$\min_{\boldsymbol{U}(k),\varepsilon} \frac{1}{2}\boldsymbol{U}^T(k)\boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{U}^T(k)\boldsymbol{F}\boldsymbol{x}(k) + \rho\varepsilon^2$$
$$\text{subject to } \boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathscr{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k) + \boldsymbol{\mathscr{b}}^{\mathrm{p}}(k)\varepsilon,\ \varepsilon \geq 0$$

$$\min_{\boldsymbol{U}(k),\varepsilon} \frac{1}{2}\boldsymbol{U}^T(k)\boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{U}^T(k)\boldsymbol{F}\boldsymbol{x}(k) + \rho\varepsilon$$
$$\text{subject to } \boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathscr{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k) + \boldsymbol{\mathscr{b}}^{\mathrm{p}}(k)\varepsilon,\ \varepsilon \geq 0$$

  - Hard constraints can be enforced by setting the related elements in $\boldsymbol{\varepsilon}(k)$ or $\boldsymbol{\mathscr{b}}^{\mathrm{p}}(k)$ to zero
  - More details and further references are given in [BBM15, Section 13.5] and [Mac02, Section 3.4]

Chance Constraints and Constraint Management

- **Chance Constraints**

$$\mathrm{P}\big(\boldsymbol{\mathcal{A}}(k)\boldsymbol{U}(k) \leq \boldsymbol{\mathcal{b}}(k) + \boldsymbol{\mathcal{W}}(k)\boldsymbol{x}(k)\big) \geq 1 - \varepsilon(k) \tag{5.6}$$

where $\mathrm{P}(\cdot)$ denotes the probability and $\varepsilon(k) \in (0,1)$

  – Note that (5.6) can also be formulated for each constraint individually (i.e. row-wise)

- **Constraint Management**

  – Constraint management consists in removing the least critical constraints until the problem is feasible

  – Constraint management is still subject to research

  – More details and further references are given in [Mac02, Section 10.2] and [CB04, Section 7.7]