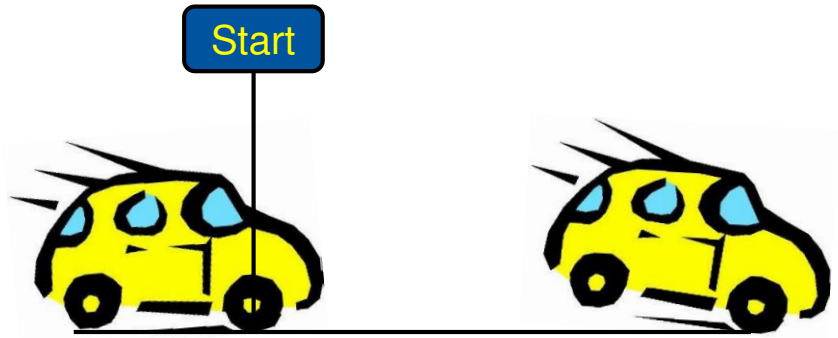
Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: examples

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Example: Optimal Control



Aim:

1. Cover maximal distance in t ∈ $[0, t_f]$
2. Start and end with zero velocity.
3. Bounded velocity and acceleration $\forall t \in [0, t_f]$.

Variables:

- $s(t)$: Distance
- $v(t)$: Velocity

} states

- $u(t)$: Acceleration

} control

Optimization problem:

$$\max_{s(\cdot),v(\cdot),u(\cdot)} \quad \left(s(t_f)\right)$$ objective function

s.t. $\dot{s}(t) = v(t)$, $\dot{v}(t) = u(t)$ model

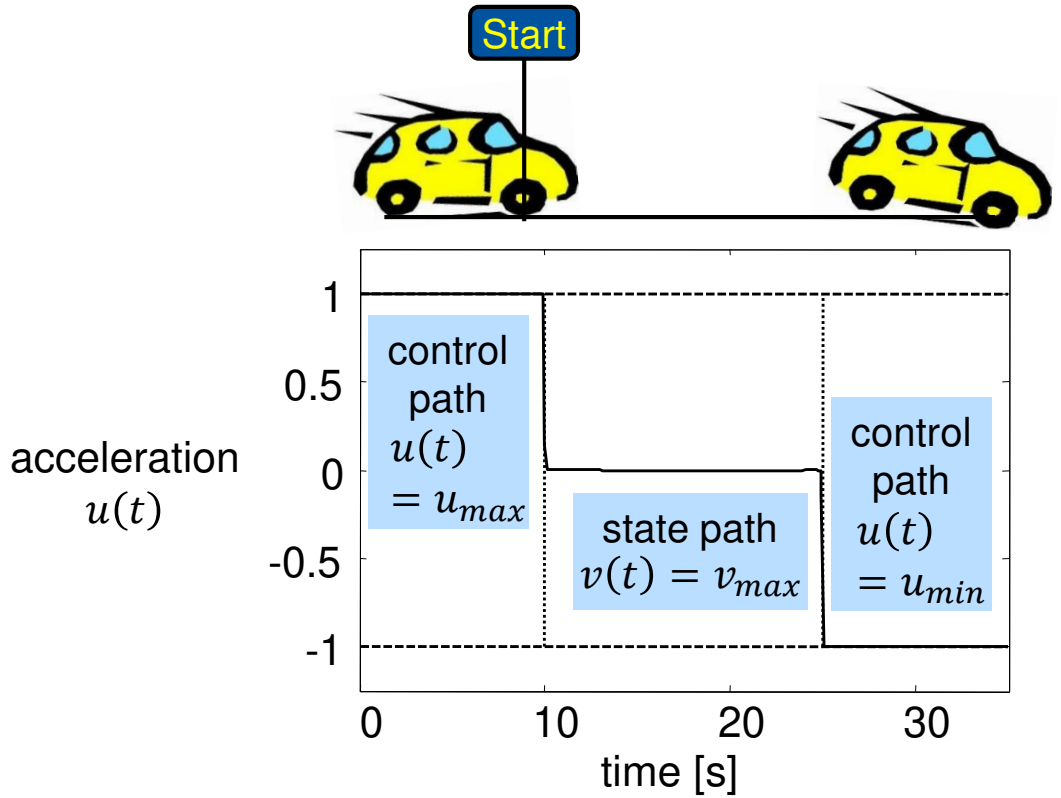
$s(t = 0) = 0$, $v(t = 0) = 0$ initial conditions

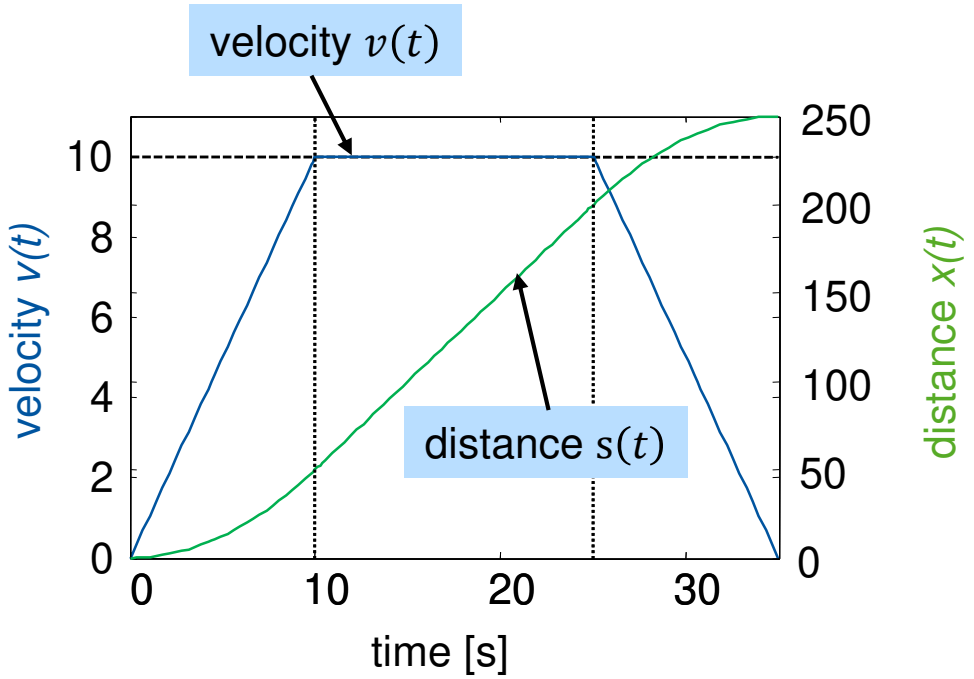
$v(t) \leq v_{max}$ state path constraint

$u_{min} \leq u(t) \leq u_{max}$ control path constraint

$v(t_f) = 0$ terminal constraint

Example: Optimal Control – Solution





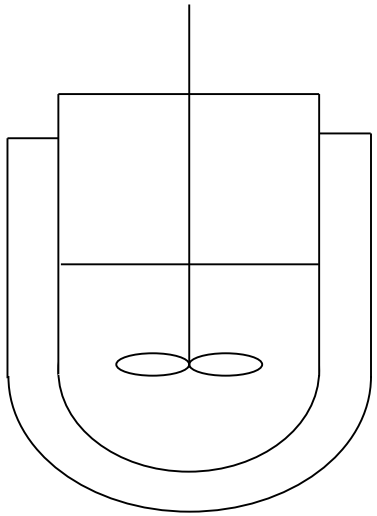
Optimal control $u(t)$ can be split into parts:

(i) maximal acceleration until the velocity limit is reached,

(ii) driving at maximal velocity

(iii) maximal slowdown until terminal constraint is satisfied.

Parameter Estimation in Batch Reactor: Problem



$$A + B \overset{k_1}{\rightarrow} C$$

$$B + C \overset{k_2}{\rightarrow} D$$

$$\left.\frac{dc_A}{dt}\right|_t = -k_1 \cdot c_A(t) \cdot c_B(t)$$

$$\left.\frac{dc_B}{dt}\right|_t = -k_1 \cdot c_A(t) \cdot c_B(t) - k_2 \cdot c_B(t) \cdot c_C(t)$$

$$\left.\frac{dc_C}{dt}\right|_t = +k_1 \cdot c_A(t) \cdot c_B(t) - k_2 \cdot c_B(t) \cdot c_C(t)$$

$$\left.\frac{dc_D}{dt}\right|_t = +k_2 \cdot c_B(t) \cdot c_C(t)$$

- Known initial concentrations: $c_A(t=0) = c_{A,0}$, $c_B(t=0) = c_{B,0}$, $c_C(t=0) = c_{C,0}$, $c_D(t=0) = c_{D,0}$
- Concentrations of A, D measured at time instances t_j: $c_{A,j}^{meas}, c_{D,j}^{meas}$.
- Unknown/uncertain parameters k_1 and k_2
- How to formulate the parameter estimation problem?

Parameter Estimation in Batch Reactor: Optimization Problem

Objective function: $$\min_{k_1,k_2,c(\cdot)} \sum_{j=1}^{m} \left[w_1 \left[c_{A,j}^{meas} - c_A(t_j) \right]^2 + w_2 \left[c_{D,j}^{meas} - c_D(t_j) \right]^2 \right]$$

Model:

s.t.
$$\left.\frac{dc_A}{dt}\right|_t = -k_1 \cdot c_A(t) \cdot c_B(t)$$
$$\left.\frac{dc_B}{dt}\right|_t = -k_1 \cdot c_A(t) \cdot c_B(t) - k_2 \cdot c_B(t) \cdot c_C(t)$$
$$\left.\frac{dc_C}{dt}\right|_t = +k_1 \cdot c_A(t) \cdot c_B(t) - k_2 \cdot c_B(t) \cdot c_C(t)$$
$$\left.\frac{dc_D}{dt}\right|_t = +k_2 \cdot c_B(t) \cdot c_C(t)$$

Initial conditions:
$$c_A(t=0) = c_{A,0}$$
$$c_B(t=0) = c_{B,0}$$
$$c_C(t=0) = c_{C,0}$$
$$c_D(t=0) = c_{D,0}$$

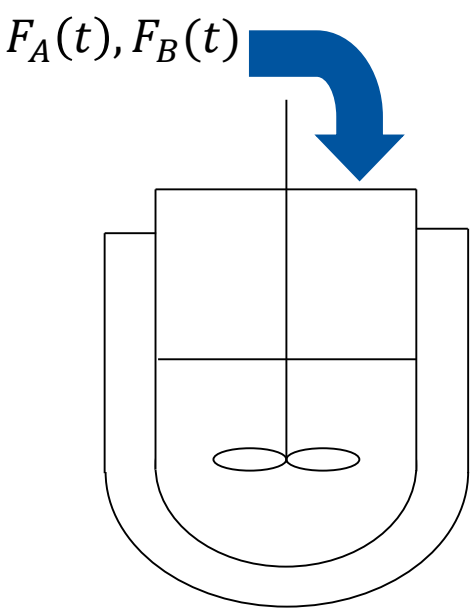
General form:
$$\min_{\boldsymbol{p},\boldsymbol{x}(\cdot)} \; \Phi(\boldsymbol{x}(t_1), \dots, \boldsymbol{x}(t_m), \boldsymbol{p})$$

s.t.
$$\left.\frac{d\boldsymbol{x}}{dt}\right|_t \equiv \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{p})$$
$$\boldsymbol{x}(t=0) = \boldsymbol{x}_0$$

$\boldsymbol{x}$: states

$\boldsymbol{p}$: parameters

Example: Optimization of Semi-batch Reactor Operation

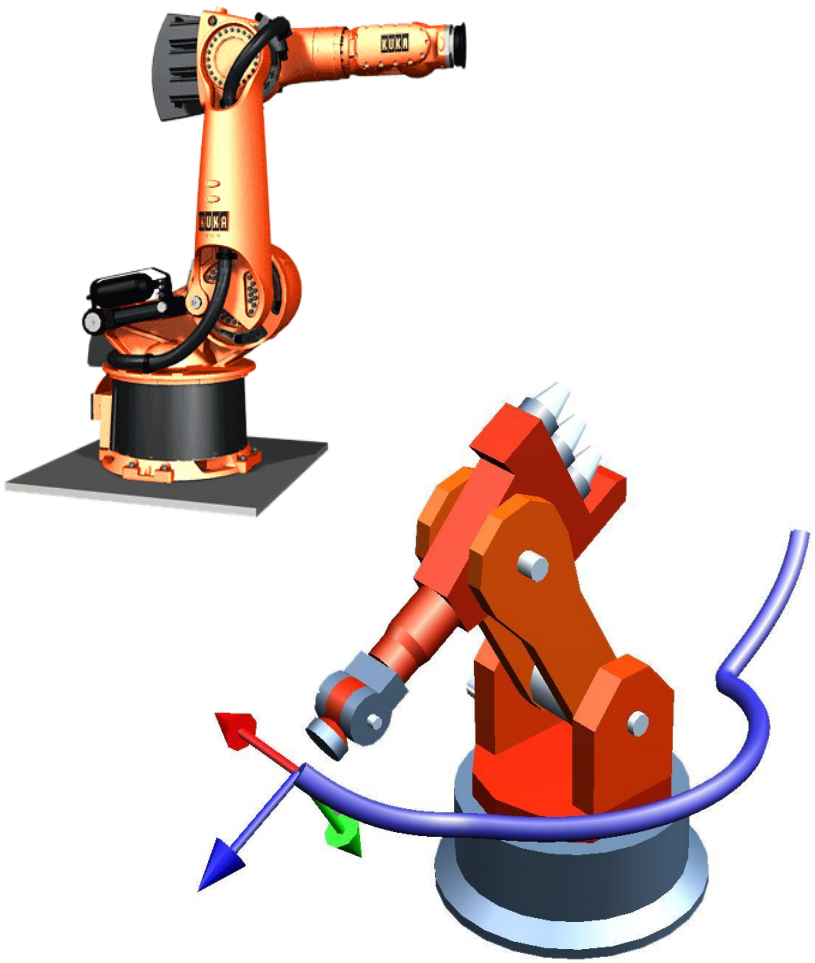


$$A + B \xrightarrow{k_1} C$$

$$B + C \xrightarrow{k_2} D$$

- Model is similar to parameter estimation problem
  - Use fitted kinetics
  - Allow for semi-batch reactor
- Fixed time interval $t \in [0, t_f]$
- Degrees of freedom: dosage of reactants: $F_A(t), F_B(t)$
  - **Functions of time.**
- Objective: maximize the production rate: $\max C_C(t_f)$
  - Terminal objective
- Guarantee a minimal selectivity: $\frac{C_C(t_f)}{C_D(t_f)} \geq S_{min}$
  - Terminal constraint
- Ensure safe operation, e.g., $C_A(t) \leq C_{max}, \forall t \in [0, t_f]$
  - Path constraint

Example: Optimal Motion Planning of Robots



- Model from mechanics
  - Structure of robot
  - Newton laws
  - Definition of acceleration
- Degrees of freedom: torque of motors as **function of time**
- Objective: minimal time: $\min t_f$
- Fixed initial positions of part
  - Initial condition
- Given final positions of part
  - Terminal constraint
- No collisions with obstacles
  - Path constraint

Figures: FG Simulation und Systemoptimierung, TU Darmstadt; Kuka Roboter GmbH

Check Yourself

- What comprises a dynamic optimization problem? Where do such problems occur in applications?
- What is the main difficulty that characterizes these problems?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: formulation and solution strategies

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Motivation

- Until now we had finite # variables and constraints
  - continuous variables: $\boldsymbol{x} \in R^{n_x}$
  - discrete variables: $\boldsymbol{y} \in \{0,1\}^{n_y}$
  - equality constraints: $c_i(\boldsymbol{x}, \boldsymbol{y}) = 0, \forall\, i \in E$
  - inequality constraints: $c_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \forall\, i \in I$

- In dynamic optimization, time-dependence results in infinite size
  - time-dependent (decision) variables
  - differential equations.

- To discuss
  - Mathematical formulation
  - Optimality conditions
  - Solution strategies

General Optimal Control Problem Formulation

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi\left(\boldsymbol{x}(t_f)\right)$$ objective

s.t. $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right), t \in [t_0, t_f]$ state equations (model)

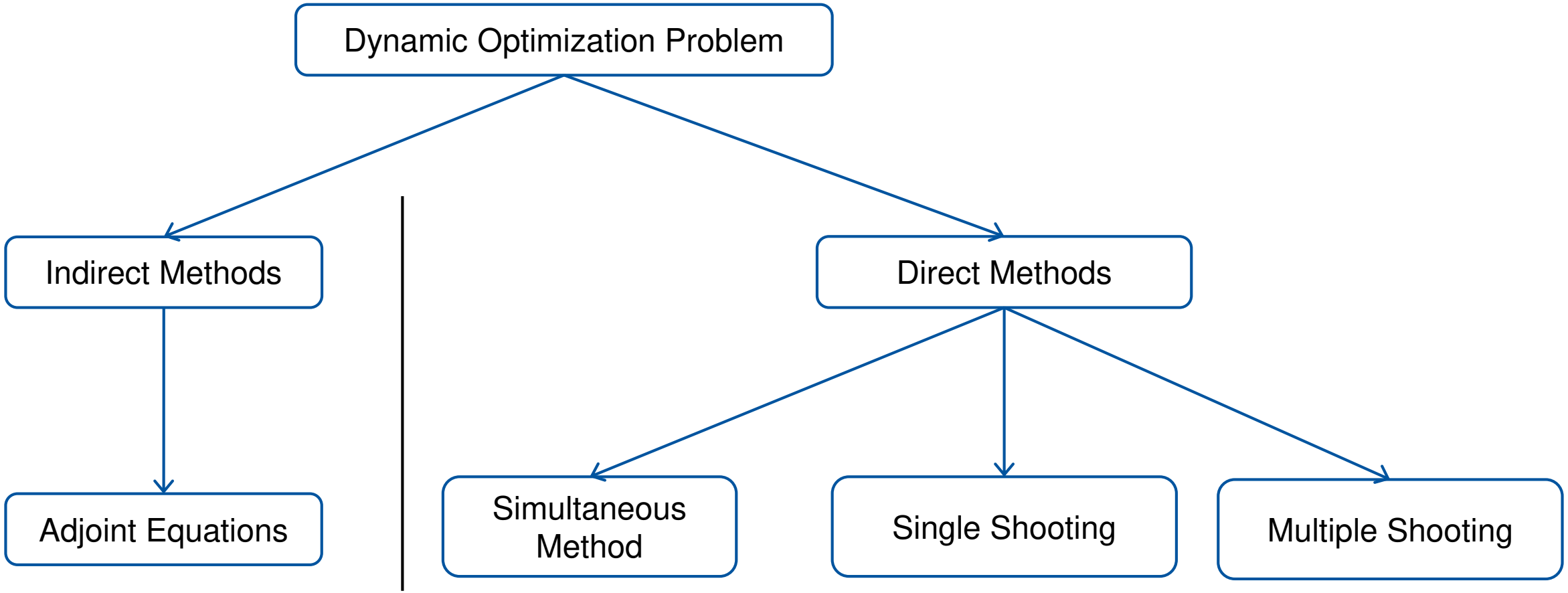
$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ initial conditions

$\boldsymbol{g}_{\mathrm{P}}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \leq \boldsymbol{0}\ \forall t \in [t_0, t_f]$ path constraints, incl. bounds on controls

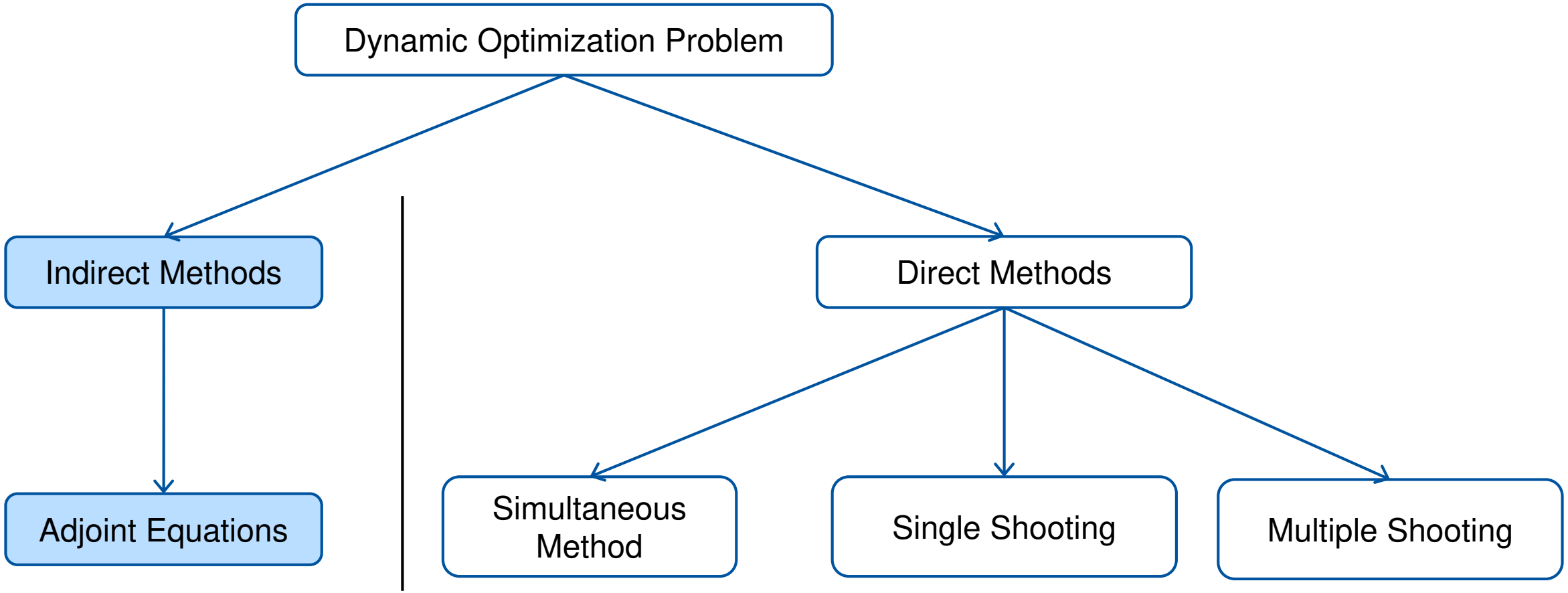
$\boldsymbol{g}_{\mathrm{T}}\left(\boldsymbol{x}(t_f)\right) \leq \boldsymbol{0}$ terminal conditions

- $\boldsymbol{u}(t) \in R^m$ controls (degrees of freedom)
- $\boldsymbol{x}(t) \in R^n$ states (determined by model)
- The initial states $\boldsymbol{x}_0$ can be additional optimization variables
- Final time t_f assumed fixed
  - variable t_f can be reformulated by scaling
- Φ is the objective functional of Mayer type
  - integral objective can be reformulated using additional state
- How to solve for infinite # variables and constraints?

Methods for Dynamic Optimization Problems



Methods for Dynamic Optimization Problems



Indirect Methods: Optimize – then – discretize

- Construction of Hamilton-form by adjoining the constraints (model equations and constraints)
  - Introduction of adjoint variables, similar to Lagrangian variables in NLP
- Derivation of necessary optimality conditions
- Numerical solution of the two-point boundary value problem
  - initial conditions of the states
  - terminal conditions for the adjoint variables
  - additional difficulty for path constraints as switching times (constraint active or not) are unknown and complementary constraints are added
- “Optimize-then-discretize”: set up optimality conditions and solve these by a discretization method.
  - Difficult or impossible for problems with path constraints

Check Yourself

- What comprises a dynamic optimization problem?
- How are dynamic problems stated?
- What is the main difficulty that characterizes these problems?
- How can these problems be solved?

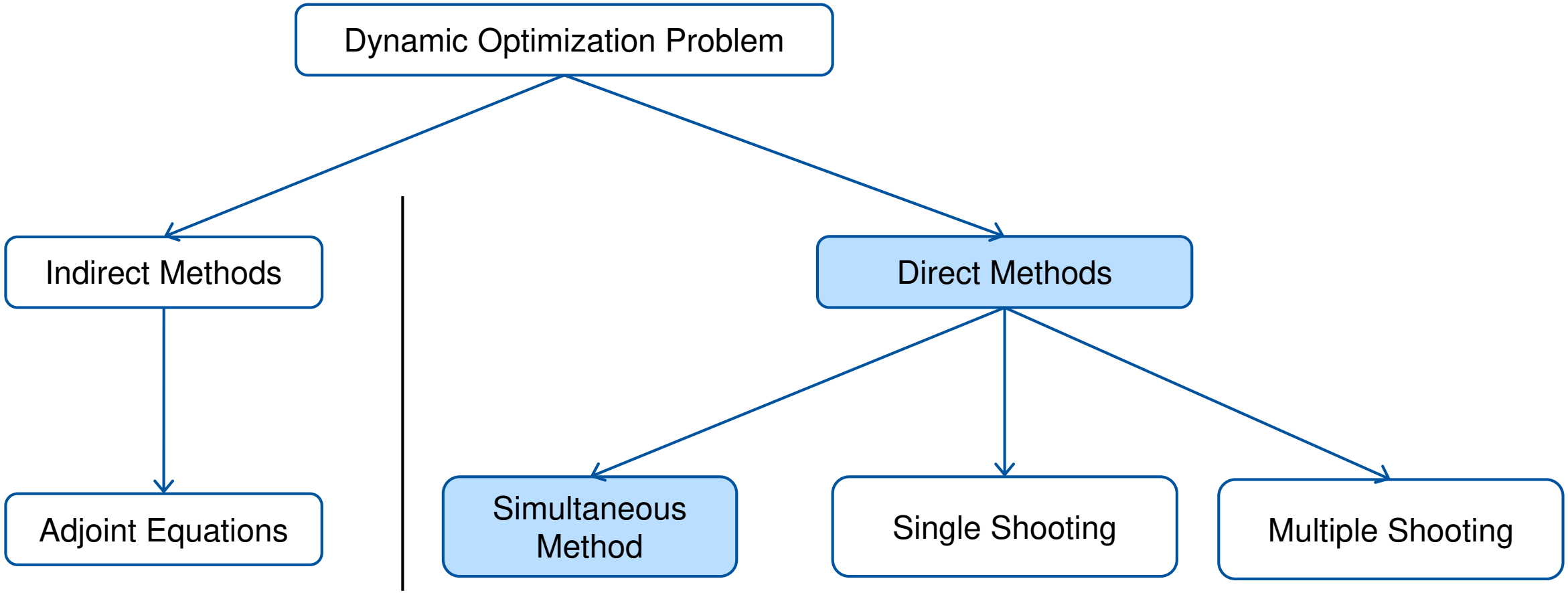
Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: simultaneous method

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Methods for Dynamic Optimization Problems



Simultaneous Method

Continuous formulation:
(w/o inequality constraints)

Objective	$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi[\boldsymbol{x}(t_f)]$
Model	s.t. $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$
I.C.	$\boldsymbol{x}(t=0) = \boldsymbol{x}_0$

Discretize $\boldsymbol{x}(t)$ & $\boldsymbol{u}(t)$ → NLP approximation:

Time discretization (e.g., equidistant):

$$0 = t_0 < t_1 < \cdots < t_{M-1} < t_M = t_f \qquad \tau = t_{k+1} - t_k$$

NLP variables:

$$\boldsymbol{x}(t_k) = \boldsymbol{x}_k,\ \boldsymbol{u}(t_k) = \boldsymbol{u}_k$$

$$\boldsymbol{y}^T = (\boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{M-1}, \boldsymbol{x}_M)$$

Objective function:

$$\hat{F}(\boldsymbol{y}) = \Phi(\boldsymbol{x}_M)$$

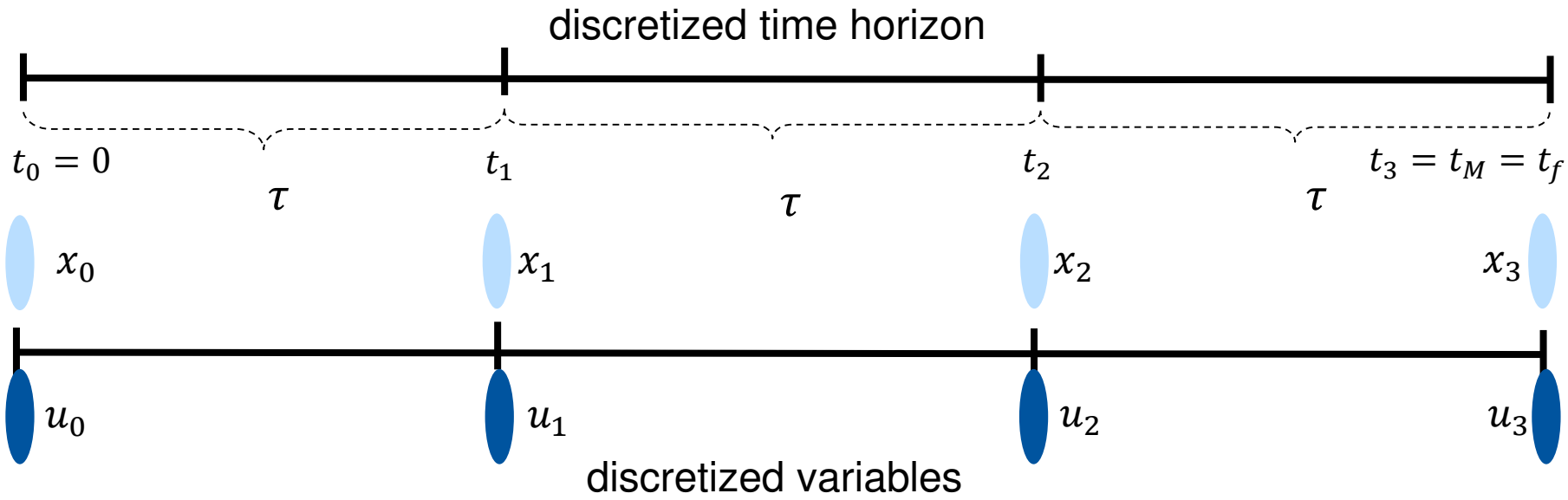
Approximate derivatives (e.g. explicit Euler):

$$\dot{\boldsymbol{x}}(t_k) \approx \frac{1}{\tau}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$$

Model equations:

$$\boldsymbol{c}_k(\boldsymbol{y}) = \boldsymbol{x}_k + \tau \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) - \boldsymbol{x}_{k+1} = \boldsymbol{0}, \forall\, k = 1, \ldots, M-1$$

Simultaneous Method – Visualization



Discretize the differential equation (assume x and u are scalar)

- Explicit Euler $c_k(y) = x_{k+1} - x_k - \tau f(x_k, u_k) = 0$
- Implicit Euler $c_k(y) = x_{k+1} - x_k - \tau f(x_{k+1}, u_{k+1}) = 0$
- Trapezoidal rule, orthogonal collocation, Runge-Kutta, …
- NLP is solved iteratively, so that implicit methods pose no additional complication

Optimality Conditions of NLP Converge to Optimality Conditions of Continuous Problem

Lagrangian function:

$$L(\boldsymbol{y}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots \boldsymbol{\lambda}_k) = \hat{F}(\boldsymbol{y}) + \sum_{k=0}^{M-1} \boldsymbol{\lambda}_k{}^T \boldsymbol{c}_k(\boldsymbol{y})$$

$$= \Phi(\boldsymbol{x}_M) + \sum_{k=0}^{M-1} \boldsymbol{\lambda}_k{}^T [\boldsymbol{x}_k + \tau \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) - \boldsymbol{x}_{k+1}]$$

expl. Euler

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \ \Phi[\boldsymbol{x}(t_f)]$$
$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$
$$\boldsymbol{x}(t=0) = \boldsymbol{x}_0$$

$$\min_{\boldsymbol{y}} \ \hat{F}(\boldsymbol{y})$$
$$\text{s.t.} \quad \boldsymbol{c}_k(\boldsymbol{y}) = \boldsymbol{0} \ \forall k$$

KKT conditions:

$$\frac{\partial L}{\partial \boldsymbol{x}_k} = (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}) + \tau \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_k}\Big|^T_{\boldsymbol{x}_k, \boldsymbol{u}_k} \boldsymbol{\lambda}_k = \boldsymbol{0}, \ k = 1, \dots, M-1$$

$$\frac{\partial L}{\partial \boldsymbol{x}_M} = -\boldsymbol{\lambda}_{M-1} + \frac{\partial \Phi}{\partial \boldsymbol{x}_M}\Big|_{\boldsymbol{x}_M} = \boldsymbol{0}$$

$$\frac{\partial L}{\partial \boldsymbol{u}_k} = \tau \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_k}\Big|^T_{\boldsymbol{x}_k, \boldsymbol{u}_k} \boldsymbol{\lambda}_k = \boldsymbol{0}, \ k = 0, \dots, M-1$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_k} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \tau \boldsymbol{f}(x_k, u_k) = \boldsymbol{0}, \ k = 0, \dots, M-1$$

$$\nabla L(\boldsymbol{y}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots \boldsymbol{\lambda}_k) = \begin{bmatrix} \nabla_{\boldsymbol{y}} \hat{F}(\boldsymbol{y}) + \sum_{k=0}^{M-1} (\nabla_{\boldsymbol{y}} \boldsymbol{c}_k(\boldsymbol{y}))^T \boldsymbol{\lambda}_k \\ \boldsymbol{c}_1(\boldsymbol{y}) \\ \boldsymbol{c}_2(\boldsymbol{y}) \\ \dots \\ \boldsymbol{c}_k(\boldsymbol{y}) \end{bmatrix} = \boldsymbol{0}$$

NLP Structure: Sparse Linear Systems

Size of NLP:

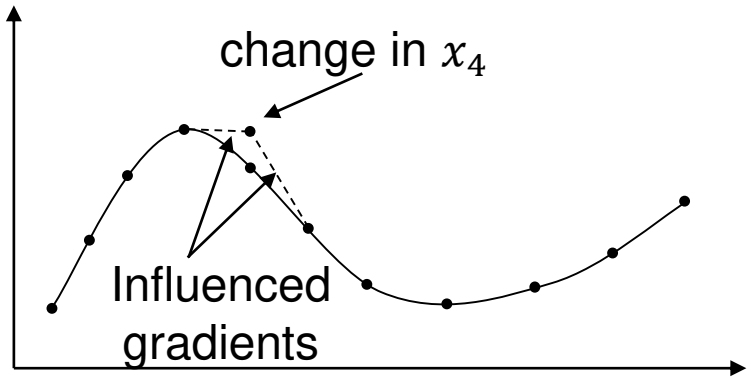
N_x: States
N_u: Controls
$M + 1$: Discretization points

$(N_x + N_u) \cdot M$ discretized NLP-variables

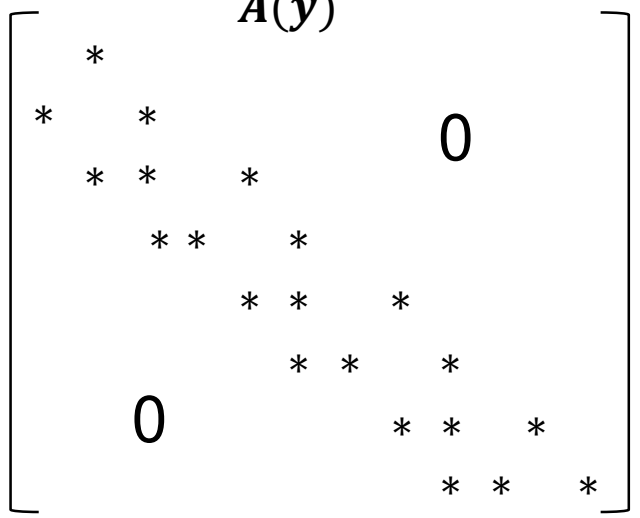
$(N_x + N_u) \cdot N_x \cdot M^2$ entries of $\boldsymbol{A} = \boldsymbol{\nabla_y c(y)}$

- M large for accurate integration
- # entries in $\boldsymbol{A}$ quadratic in M,
  # nonzero entries much smaller
- Large sparse NLP: need specialized solvers.

Discretization results in sparsity



$\boldsymbol{A(y)}$



Handling of Inequality Constraints (ICs)

The car example: Discretization of inequality constraints

$\max_{s(\cdot),v(\cdot),u(\cdot)} \quad (s(t_f))$	objective function
s.t. $\dot{s}(t) = v(t),\ \dot{v}(t) = u(t)$	model
$s(t=0) = 0,\ v(t=0) = 0$	initial conditions
$v(t) \le v_{max}$	state path constraint
$u_{min} \le u(t) \le u_{max}$	control path constraint
$v(t_f) = 0$	terminal constraint

$$v_k \le v_{max}$$
$$u_k \le u_{max}$$
$$-u_k \le u_{min}$$
$$k = 1, \dots, M$$

- Discretization of path constraints leads to many inequalities!
- Active-set methods have combinatorial complexity.
- Interior-point methods are more suitable.

Check Yourself

- What is the main idea of full discretization? Is this method always efficient? What is the main drawback?
- Is the explicit Euler method a good choice and why?
- Can the NLPs resulting from full discretization be handled efficiently? How?

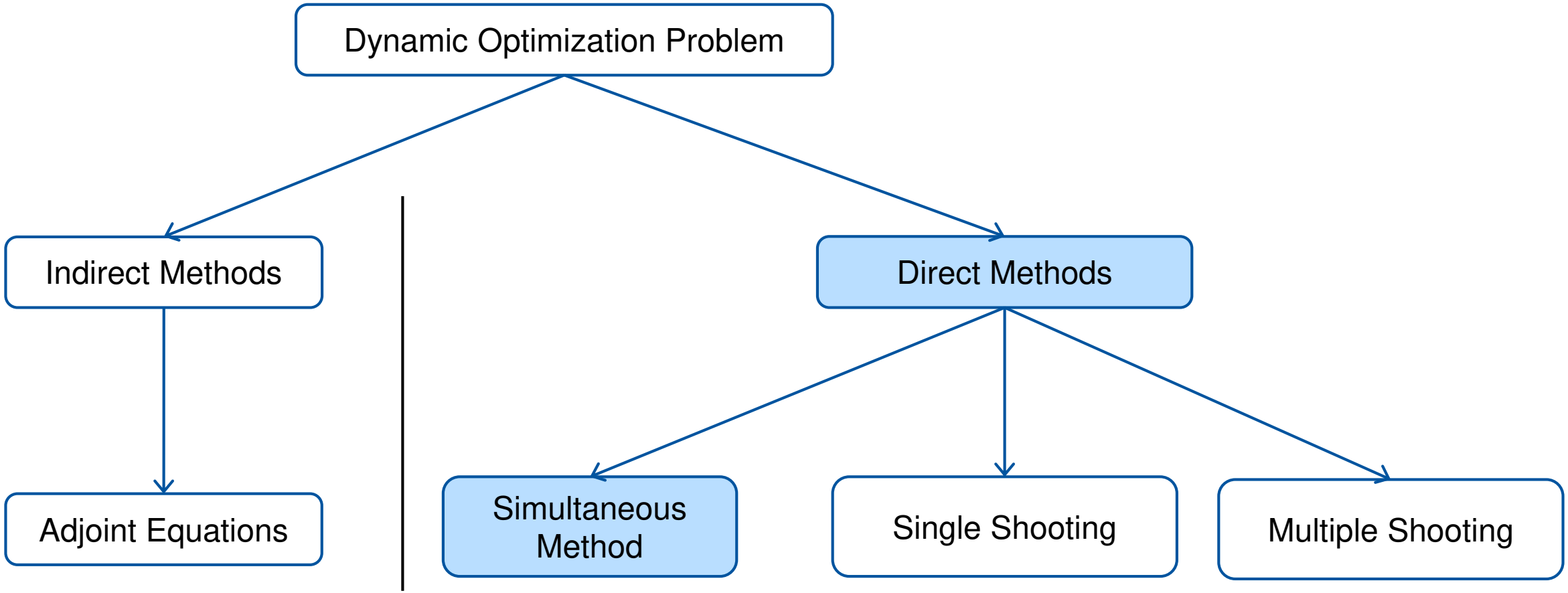
Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: collocation

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Methods for Dynamic Optimization Problems



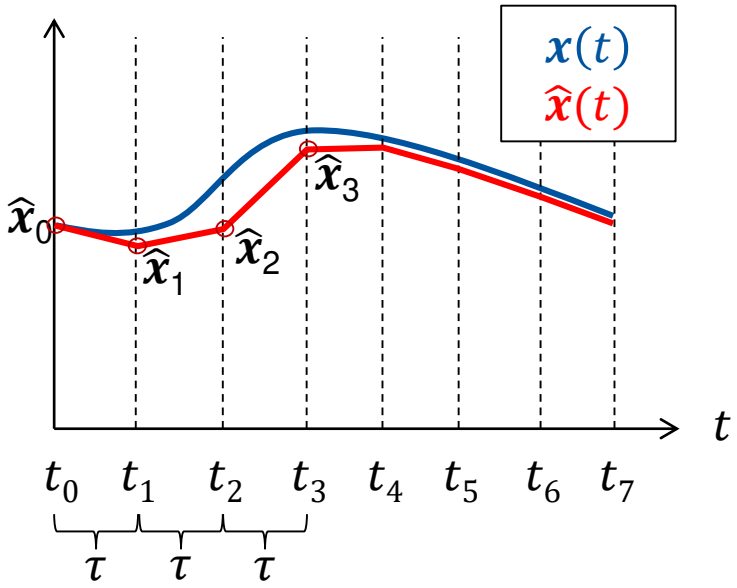
Discussion on Integration Schemes for Full Discretization

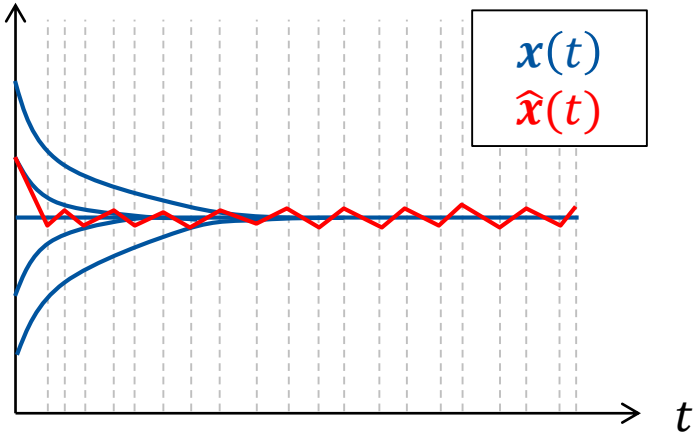
Nomenclature:

- $\boldsymbol{x}(t)$ is the trajectory of $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0$
- $\hat{\boldsymbol{x}}(t)$ is the trajectory resulting from discretization

Explicit Euler discretization:

- Small consistency order
- Only for $t = t_0$, $\boldsymbol{x}(t) = \hat{\boldsymbol{x}}(t)$
- Unstable for large τ





Desired properties:

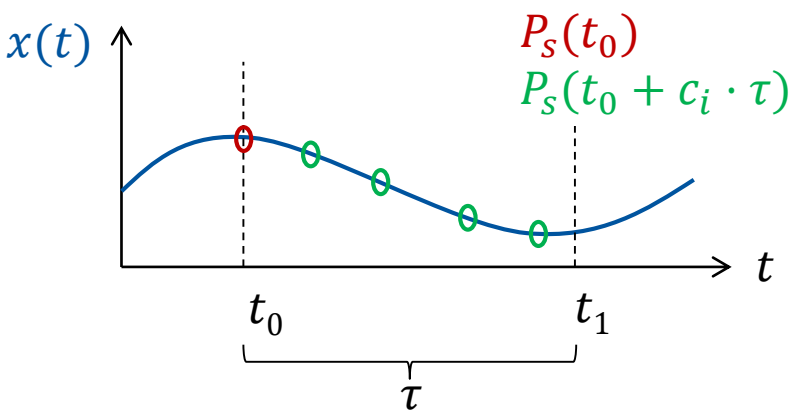
- High consistency order
- Stability → implicit methods

Collocation Method

ODE

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t)\big)$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \in \mathbb{R}^n$$



- Assume an s^{th} degree polynomial $\boldsymbol{P}_s(t)$

  $$\boldsymbol{P}_s(t) = (t - t_0)^s \cdot \boldsymbol{\alpha}_s + (t - t_0)^{s-1} \cdot \boldsymbol{\alpha}_{s-1} + \cdots + (t - t_0) \cdot \boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_0$$

  $\Rightarrow (s + 1)$ x n coefficients: $\boldsymbol{\alpha}_j \in R^n,\ j = 0,1, \ldots, s$

- $\boldsymbol{P}_s(t)$ should exactly fulfill the initial condition and the ODE at s time-points in $[t_0, t_0 + \tau]$

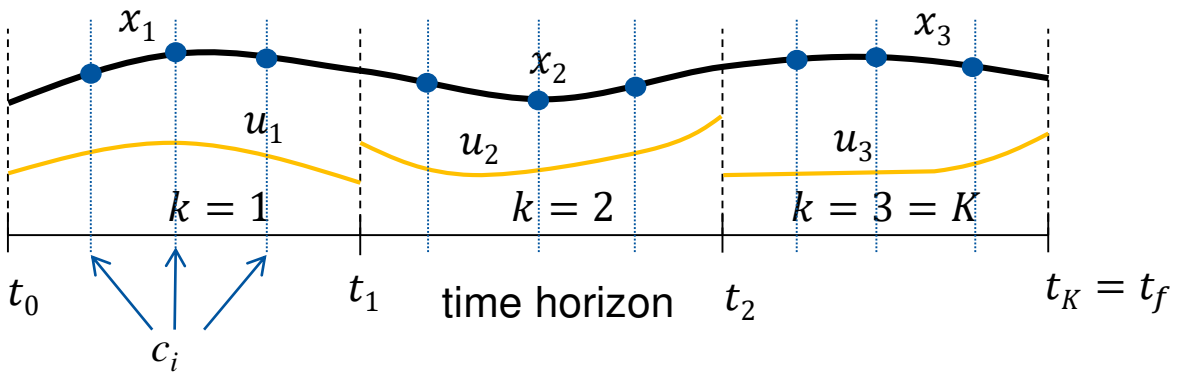
  $$\boldsymbol{P}_s(t_0) = \boldsymbol{x}_0$$
  $$\dot{\boldsymbol{P}}_s(t_0 + c_i\tau) = \boldsymbol{f}(t_0 + c_i\tau, \underbrace{\boldsymbol{P}_s(t_0 + c_i\tau)}_{\boldsymbol{x}(t_0 + c_i\tau)})$$

  } $(s + 1)$ x n nonlinear equations

- Normally $0 \leq c_1 < c_2 < \cdots < c_s \leq 1$ mutually distinct and ascending

Collocation Method: Implementation

- Divide the time horizon into K parts (index k), so-called finite elements:



- The control and differential variables ($\boldsymbol{u}_k(t)$ and $\boldsymbol{x}_k(t)$) are represented by Lagrange polynomials:

$$\boldsymbol{u}_k(\sigma) = \boldsymbol{P}_u(t_{k-1} + \sigma\tau) = \sum_{j=1}^{s} L_j(\sigma) u_{k,j} \;,\; \boldsymbol{x}_k(\sigma) = \boldsymbol{P}_x(t_{k-1} + \sigma\tau) = \sum_{j=0}^{s} L_j(\sigma) \boldsymbol{x}_{k,j} \rightarrow s \text{ collocation points}$$

- The ODE on the finite element k is approximated by solving the nonlinear equation system at each collocation point $c_i, i \in \{1, \dots, s\}$

$$\sum_{j=0}^{s} \dot{L}_j(c_i) \cdot \underbrace{\boldsymbol{x}_{k,j}}_{\boldsymbol{P}_x(t_{k-1}+c_j\tau)} = \dot{\boldsymbol{P}}_x(t_{k-1} + c_i\tau) = \boldsymbol{f}\Big(\underbrace{\boldsymbol{x}_{k,i}}_{\boldsymbol{P}_x(t_{k-1}+c_i\tau)}, \underbrace{\boldsymbol{u}_{k,i}}_{\boldsymbol{P}_u(t_{k-1}+c_i\tau)} \Big)$$

- This nonlinear equation system uniquely determines the values of $\boldsymbol{x}_{k,j}$ for fixed $u_{k,i}$ (and fixed c_i)

Direct Collocation NLP

The following NLP results using the simultaneous approach with collocation method

- Objective $\min_{\boldsymbol{x},\boldsymbol{u}} \Phi(\boldsymbol{x}_{K+1,0})$
- ODE $\sum_{j=0}^{s} \dot{L}_j(c_i) \cdot \boldsymbol{x}_{k,j} = \boldsymbol{f}(\boldsymbol{x}_{k,i}, \boldsymbol{u}_{k,i}), i = 1, \dots s$
- Connect elements $\boldsymbol{x}_{k+1,0} = \sum_{j=0}^{s} L_j(1) \cdot \boldsymbol{x}_{k,j}$
- Initial point $\boldsymbol{x}_{1,0} = \boldsymbol{x}_0$
- Endpoint $\boldsymbol{x}(t_f)$ $\boldsymbol{x}_{K+1,0} = \sum_{j=0}^{s} L_j(1) \cdot \boldsymbol{x}_{K,j}$
- Bounds $\underline{\boldsymbol{u}} \leq \boldsymbol{u}_{k,j} \leq \overline{\boldsymbol{u}}$

$$\begin{aligned} \min_{\boldsymbol{u}(\cdot),\boldsymbol{x}(\cdot)} \quad & \Phi(\boldsymbol{x}(t_f)) \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \in R^n \\ & \underline{\boldsymbol{u}} \leq \boldsymbol{u}(t) \leq \overline{\boldsymbol{u}} \in R^m \end{aligned}$$

- Path- and terminal constraints can be added accordingly.
- Optimization variables: control and differential variables.

Check Yourself

- Is the explicit Euler method a good choice and why?
- What is the key idea of collocation compared to Euler?

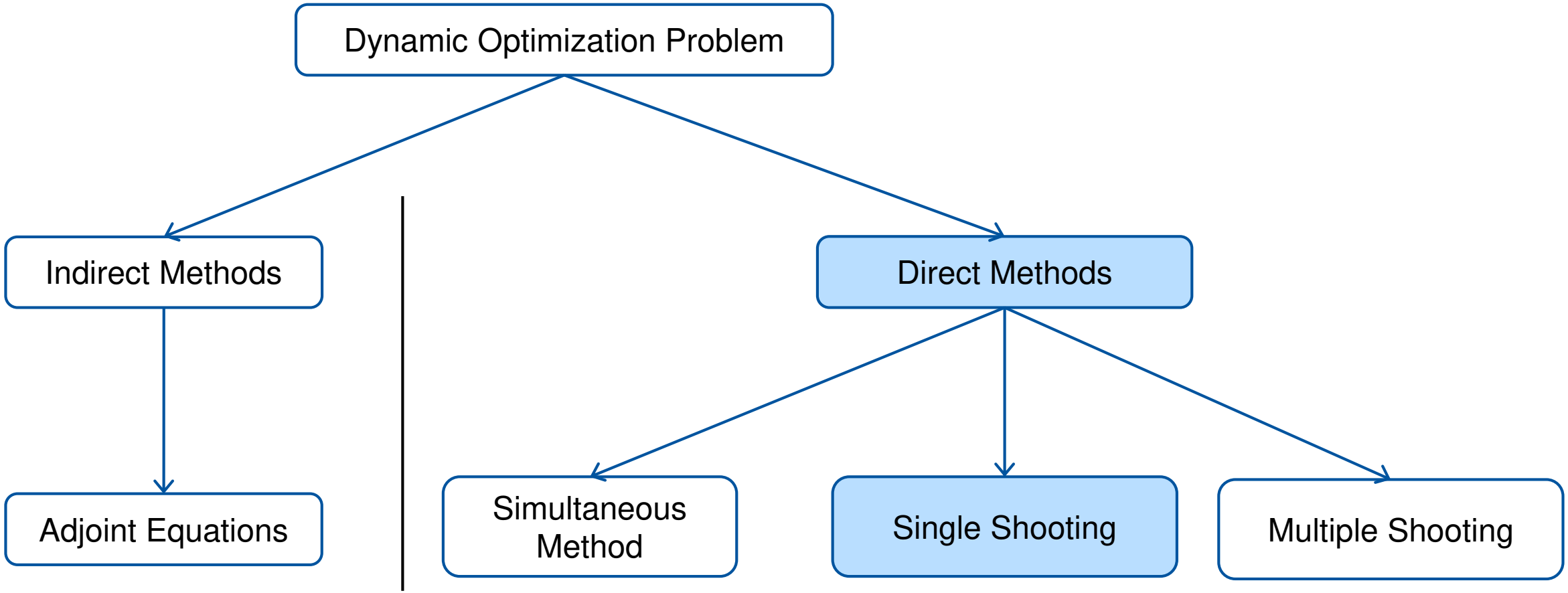
Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: sequential method

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Methods for Dynamic Optimization Problems



Discretization of Control Variables

The control variables $\boldsymbol{u}(t) \in R^{n_u}$ are approximated by a series of (typically orthonormal) basis functions $\sigma_{i,k}(t)$ and corresponding coefficients $p_{i,k}$:

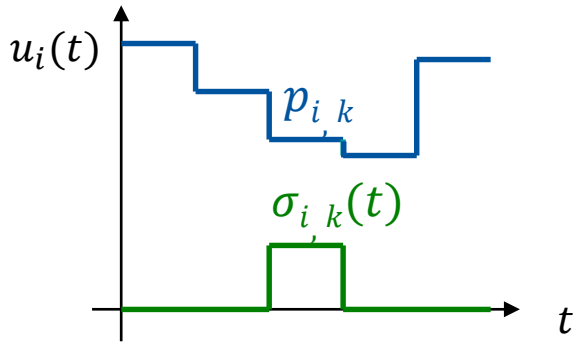
$$u_i(t) \approx \sum_{k=1}^{s} p_{i,k} \cdot \sigma_{i,k}(t)\,,\ i = 1, \dots, n_u$$

$p_{i,k}$ parameters

$\sigma_{i,k}(t)$ basis functions

Example:

$$\sigma_{i,k}(t) = \begin{cases} 1 & \forall t \in [t_k, t_{k+1}] \\ 0 & \forall t \notin [t_k, t_{k+1}] \end{cases}$$



- We have restricted the degrees of freedom from $\boldsymbol{u}(t)$ to $p_{i,k}$
  - Constraints on $\boldsymbol{u}$ imply constraints on $\boldsymbol{p}$
  - The more parameters, the less restricted
- For fixed $\boldsymbol{p}$ we determine $\boldsymbol{x}(t)$ by the differential equations!
  - Initial conditions parametrized $\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}_0}(\boldsymbol{p})$

Single Shooting Approach: Simple Constraints

- Original problem

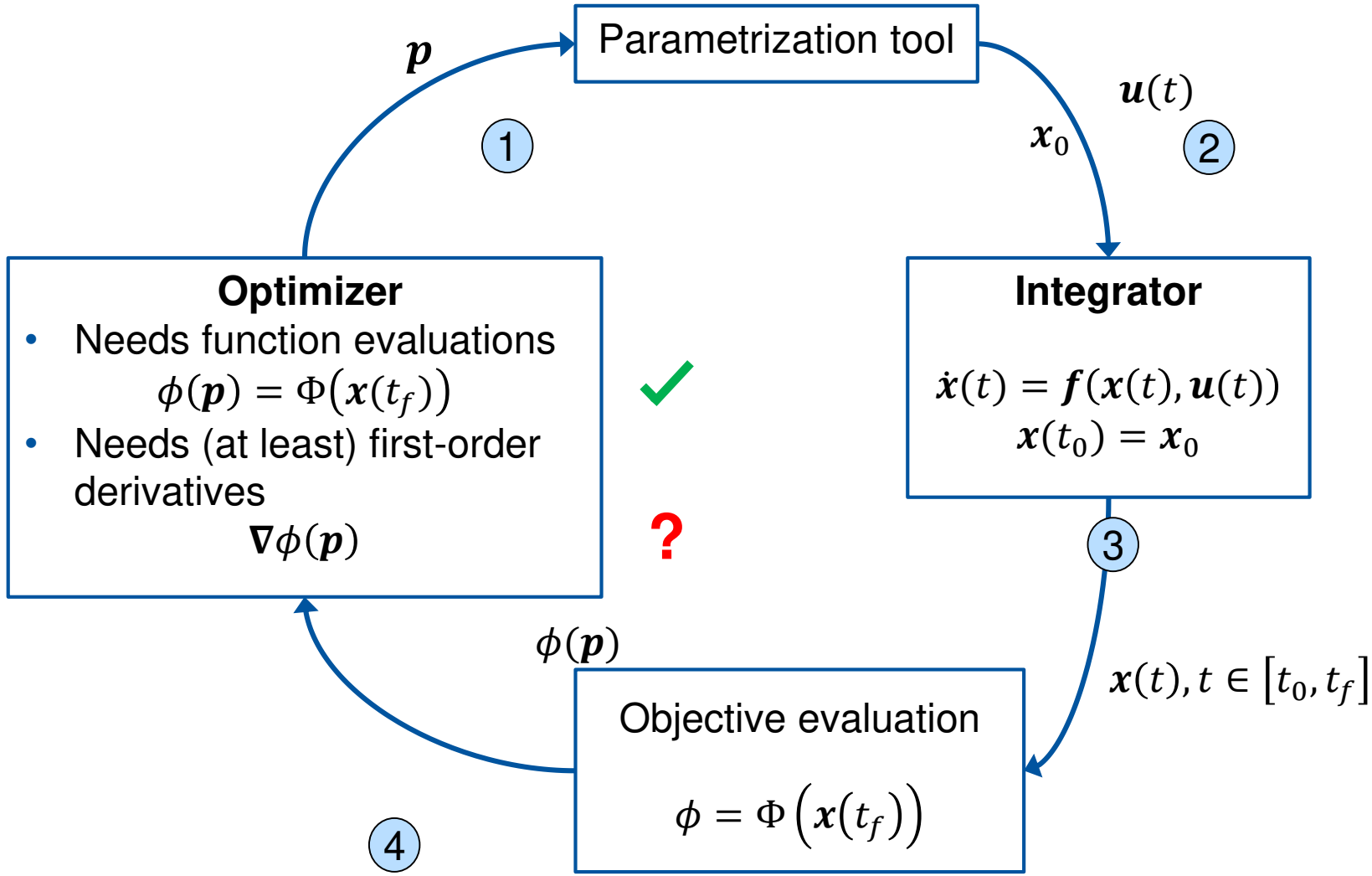
$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi\left(\boldsymbol{x}(t_f)\right)$$
$$\text{s.t. } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$$
$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$$
$$\boldsymbol{u}^L \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^U$$

- Restricted to

$$\min_{\boldsymbol{p}} \phi(\boldsymbol{p})$$
$$\text{s.t. } \boldsymbol{c}(\boldsymbol{p}) \leq \boldsymbol{0}$$

where
  - $\phi(\boldsymbol{p}) = \Phi\left(\boldsymbol{x}(t_f)\right)$
  - $\boldsymbol{u}(t)$ parametrized by $\boldsymbol{p}$
  - $\boldsymbol{c}(\boldsymbol{p}) \leq \boldsymbol{0}$ represents $\boldsymbol{u}^L \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^U$
  - $\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}_0}(\boldsymbol{p})$
  - $\boldsymbol{x}(t)$ solves

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$
$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$$



Check Yourself

- What is the main difference of single shooting in comparison to full discretization?
- What comprises single shooting method?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: calculation of derivatives

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Single Shooting Approach: Simple Constraints

- Original problem

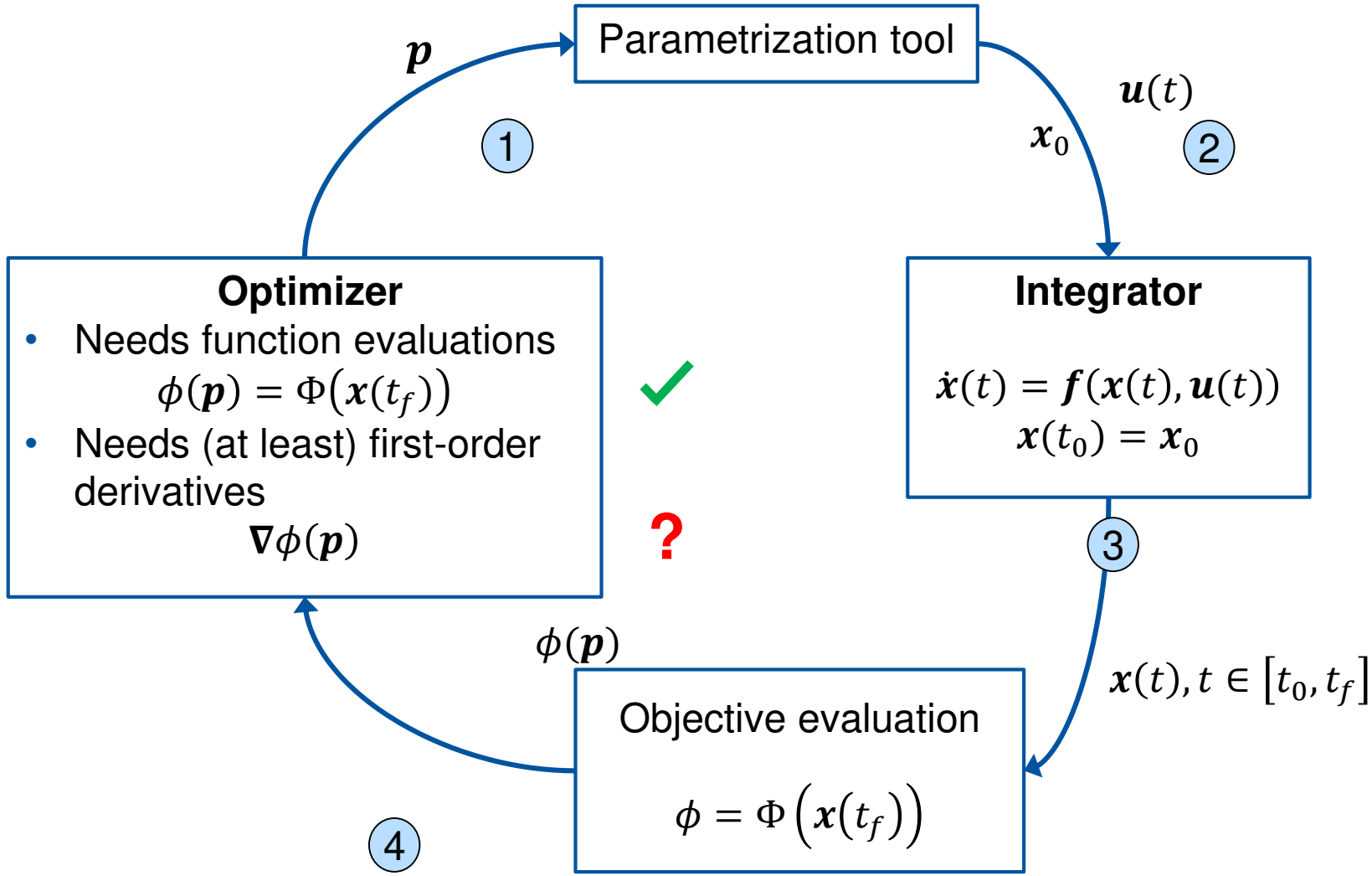
$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \Phi\left(\boldsymbol{x}(t_f)\right)$$
$$\text{s.t. } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$$
$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$$
$$\boldsymbol{u}^L \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^U$$

- Restricted to

$$\min_{\boldsymbol{p}} \phi(\boldsymbol{p})$$
$$\text{s.t. } \boldsymbol{c}(\boldsymbol{p}) \leq \boldsymbol{0}$$

where
  - $\phi(\boldsymbol{p}) = \Phi\left(\boldsymbol{x}(t_f)\right)$
  - $\boldsymbol{u}(t)$ parametrized by $\boldsymbol{p}$
  - $\boldsymbol{c}(\boldsymbol{p}) \leq \boldsymbol{0}$ represents $\boldsymbol{u}^L \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^U$
  - $\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}_0}(\boldsymbol{p})$
  - $\boldsymbol{x}(t)$ solves

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$
$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$$



Gradients by Perturbation (Forward Finite Differences)

- We need to evaluate gradient

$$\nabla\phi(\boldsymbol{p}) = \left(\left.\frac{\partial\phi}{\partial p_1}\right|_{\boldsymbol{p}}, \dots, \left.\frac{\partial\phi}{\partial p_j}\right|_{\boldsymbol{p}}, \dots \right)^T$$

- Approximate derivatives by finite differences, one parameter at a time

$$\left.\frac{\partial\phi}{\partial p_j}\right|_{\boldsymbol{p}} = \frac{\phi(\boldsymbol{p}+\boldsymbol{e}_j\Delta)-\phi(\boldsymbol{p})}{\Delta}, \qquad \boldsymbol{e}_j=(0,\dots,0,1,0,\dots)^T$$

- Each function evaluation requires integration of the model equations for the given parameter value

- Advantages and disadvantages:
  - ☺ Easy to implement (for-loop around the simulator)
  - ☹ Limited approximation quality (how to select Δ)
  - ☹ High computational effort: as many extra integrations as parameters

Gradients by Forward Sensitivities

- We need to evaluate gradient $\nabla\phi(\boldsymbol{p}) = \left(\left.\frac{\partial\phi}{\partial p_1}\right|_{\boldsymbol{p}}, \dots, \left.\frac{\partial\phi}{\partial p_j}\right|_{\boldsymbol{p}}, \dots \right)^T$
  - $\phi(\boldsymbol{p}) = \Phi\big(\boldsymbol{x}(t_f)\big)$
  - $\boldsymbol{u}(t)$ parametrized by $\boldsymbol{p}$
  - $\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}_0}(\boldsymbol{p})$
  - $\boldsymbol{x}(t)$ solves $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \boldsymbol{x}(t_0) = \boldsymbol{x}_0$
- Chain rule $\left.\frac{\partial\phi}{\partial p_j}\right|_{\boldsymbol{p}} = \sum_i \left(\left.\frac{\partial\Phi}{\partial x_i}\right|_{\boldsymbol{p}} \cdot \left.\frac{\partial x_i}{\partial p_j}\right|_{\boldsymbol{p}, t=t_f} \right)$
- We get $\left.\frac{\partial\Phi}{\partial x_i}\right|_{\boldsymbol{p}}$ from knowledge of Φ and solution of $\boldsymbol{x}(t)$ for given $\boldsymbol{p}$
- With sensitivities $S_{i,j}(t) = \left.\frac{\partial x_i}{\partial p_j}\right|_t$ we can evaluate $\nabla\phi(\boldsymbol{p}) = \left(\sum_i \left(\left.\frac{\partial\Phi}{\partial x_i}\right|_{\boldsymbol{p}} \cdot S_{i,1}(t_f) \right), \dots, \sum_i \left(\left.\frac{\partial\Phi}{\partial x_i}\right|_{\boldsymbol{p}} \cdot S_{i,j}(t_f) \right), \dots \right)^T$

Calculation of Forward Sensitivities $S_{i,j}(t) = \left.\frac{\partial x_i}{\partial p_j}\right|_t$

- Differentiate model equations wrt $\boldsymbol{p}$, recalling (hidden) $\boldsymbol{p}$-dependence
  - $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \boldsymbol{x}(t_0) = \boldsymbol{x}_0$
  - $\left.\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{p}}\right|_t = \left.\frac{d\boldsymbol{f}}{d\boldsymbol{p}}\right|_{\boldsymbol{x}(t),\boldsymbol{p},\boldsymbol{u}(t)} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(t),\boldsymbol{p},\boldsymbol{u}(t)} \left.\frac{d\boldsymbol{x}}{d\boldsymbol{p}}\right|_t + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{p},\boldsymbol{u}(t)} \left.\frac{d\boldsymbol{u}}{d\boldsymbol{p}}\right|_{\boldsymbol{p},t}$
- Known $\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(t),\boldsymbol{p},\boldsymbol{u}(t)}$ and $\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{p},\boldsymbol{u}(t)}$ from given $\boldsymbol{f}$
  - manual, symbolic or algorithmic differentiation
- We have differential equation for $\boldsymbol{S}(t)$:
  - $\dot{\boldsymbol{S}}(t) = \left.\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{p}}\right|_t = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(t),\boldsymbol{p},\boldsymbol{u}(t)} \boldsymbol{S}(t) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{p},\boldsymbol{u}(t)} \left.\frac{d\boldsymbol{u}}{d\boldsymbol{p}}\right|_{\boldsymbol{p},t}$
  - $\left.\frac{d\boldsymbol{u}}{d\boldsymbol{p}}\right|_{\boldsymbol{p},t}$ from parametrization
  - I.C.: $\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}_0}(\boldsymbol{p}) \rightarrow \boldsymbol{S}(t_0) = \left.\frac{d\widetilde{\boldsymbol{x}_0}}{d\boldsymbol{p}}\right|_{\boldsymbol{p},t}$

Remarks on Forward Sensitivities

- Gradients of terminal constraints $\boldsymbol{g}_{\mathrm{T}}\left(\boldsymbol{x}(t_f)\right) \leq \mathbf{0}$ can be calculated without substantial extra cost
  - $\left.\frac{\partial g_{\mathrm{T,k}}}{\partial p_j}\right|_{\boldsymbol{p}} = \sum_i \left(\left.\frac{\partial g_{\mathrm{T,k}}}{\partial x_i}\right|_{\boldsymbol{p}} \cdot S_{i,j}(t) \right)$
  - Path-constraints more complicated

- Advantages and disadvantages:
  - ☺ Exact in theory, accurate in practice
  - ☺ If implemented efficiently, faster than finite difference
  - ☹ Hard to implement efficiently
  - ☹ Computational effort scales with # parameters

- More efficient alternative: adjoint (backward mode)
  - ☺ Computational effort does not scales with # inputs (= parameters)
  - ☹ Computational effort scales with # outputs (objective function, constraints, …)
  - ☹ Very hard to understand and implement efficiently

Check Yourself

- How are gradients computed?
- What comprises the sensitivity equation system?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: multiple shooting, solvers, and global optimization

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Sequential vs Simultaneous Method

- Advantages of sequential method
  - enables error-controlled sophisticated integration (DASPK, IDAS, LIMEX, NIXE,..)
  - applicable to very large and stiff models
  - small NLP → standard NLP solvers can be employed
  - suitable for black-box algorithms, e.g., stochastic global

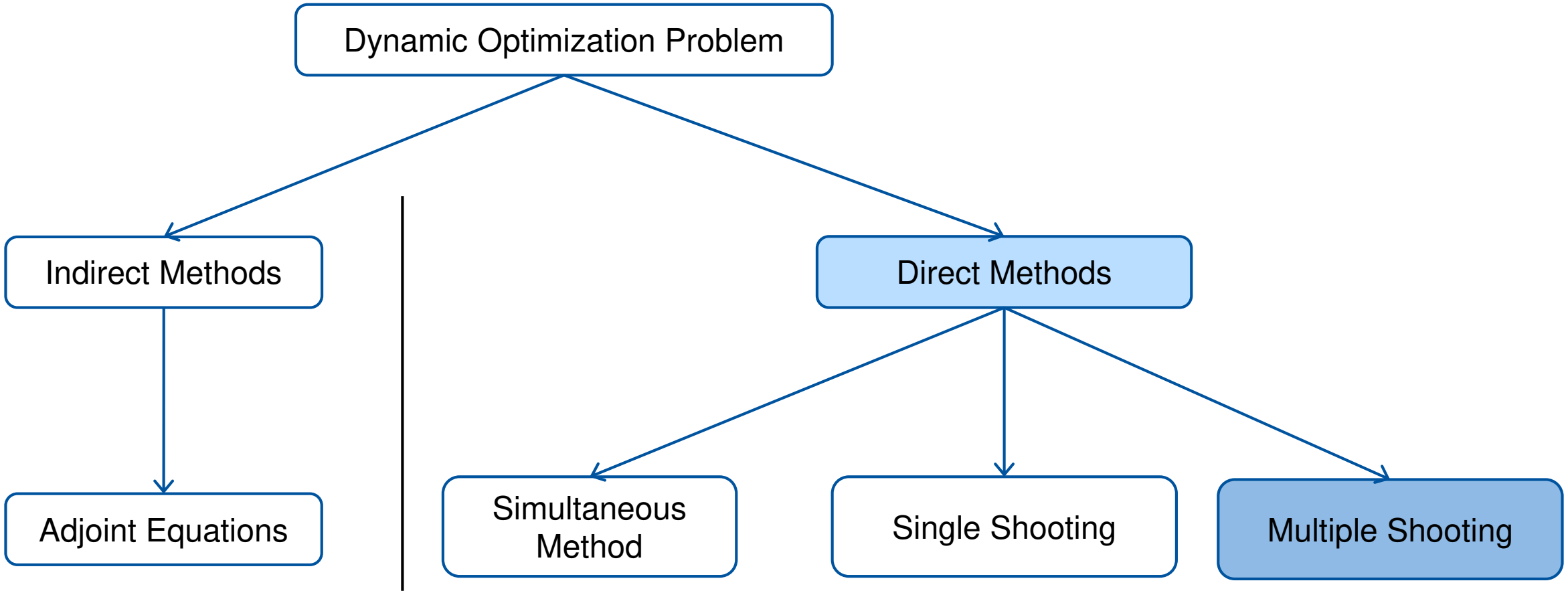
- Disadvantages of sequential method
  - requires the solution of the simulation problem (and gradients) in every (major) iteration
  - problematic for unstable systems
  - 2nd derivatives are expensive

- For sequential method choose NLP solver that requires few (major) iterations

- For simultaneous method choose NLP solver that scales well with many variables and constraints

- Suitable parameterization of the control variables is difficult

Methods for Dynamic Optimization Problems



Sensitivities in Single Shooting Methods

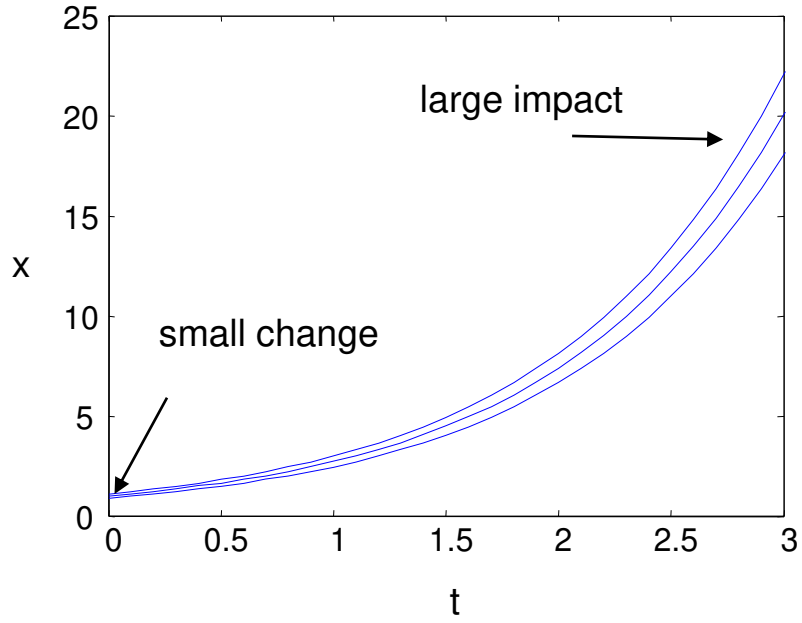
Single shooting methods face fundamental limitations for unstable systems.

Example $\dot{x}(t) = x(t), \quad x(t_0) = x_0$

The analytical solution is:

$$x(t) = x_0 \cdot e^{(t-t_0)}$$

Take $x_0 = p$ to meet $x(t_f) = x_f$



Analytical solution

$$c(p) = p \cdot e^{(t_f-t_0)} - x_f = 0 \qquad \left.\frac{\partial c}{\partial p}\right|_p = e^{(t_f-t_0)}$$

A small change in p causes a large change in the value of the constraint.

The problem is very sensitive with respect to the decision variable.

Multiple Shooting Method – Idea

To avoid large sensitivities, the integration domain is split in, e.g., two intervals: $t_I \in [t_0, t_1]$

$$t_{II} \in [t_1, t_f]$$

The initial value of the 2nd integration problem is treated as an additional degree of freedom.

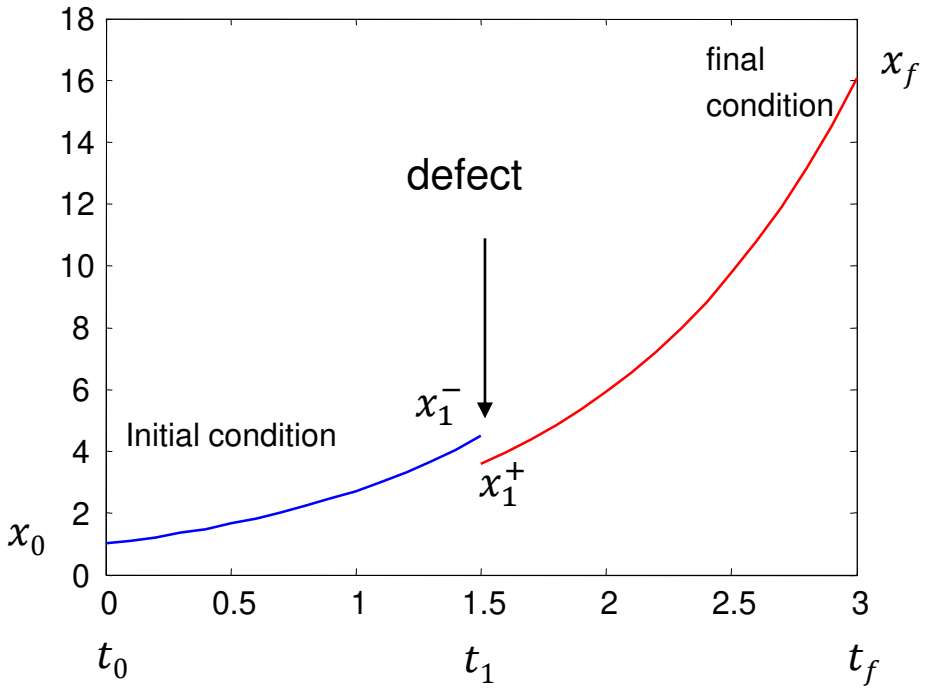
$$x_1^+ = p_2$$

$$x_0 = p_1$$

An additional constraint guarantees continuity:

$$c_1(x) = x_1^+ - x_1^- = p_2 - p_1 \cdot e^{(t_1 - t_0)} = 0$$

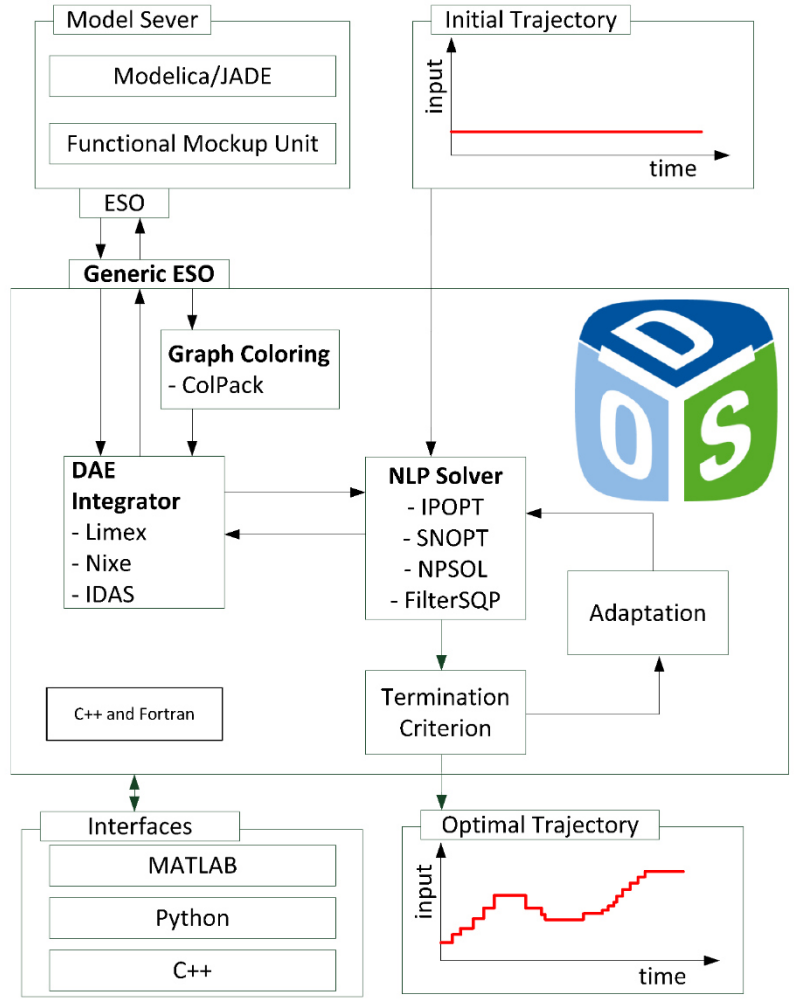
$$c_2(x) = x(t_f) - x_f = p_2 \cdot e^{(t_f - t_1)} - x_f = 0$$



- The sensitivity with respect to the decision variables decreases, but the problem size increases!
  The KKT-Matrix is sparse and partially decoupled.
- Structure exploitation and parallel computing architectures are applicable.

Software Tools For Dynamic Optimization

- Single and multiple shooting implemented in our open source tool DyOS (Dynamic Optimization Software)
  - Features for adaptation of control grid
  - Control structure detection (covered later)
  - Available on https://git.rwth-aachen.de/avt.svt/public/dyos
- Commercial/open source codes for single/multiple shooting available:
  - gPROMS
  - Dymola Optimization library
  - MUSCOD-II
  - DOTcvpSB
  - CasADI
  - ...
- Codes for full discretization
  - APMonitor
  - CasADI
  - pyomo.dae
  - ...



Challenges in Global Dynamic Optimization

- Some approaches for indirect methods

- Simultaneous method in principle possible but very large NLP
  - Reduced space formulations possible: Mitsos, Chachuat & Barton SIOPT 2009

- For sequential method relaxation of functions is nontrivial
  - Typically via differential inequalities of differential inclusions $\boldsymbol{f}^L\left(\boldsymbol{x}(t)\right) \leq \dot{\boldsymbol{x}}(t) \leq \boldsymbol{f}^U\left(\boldsymbol{x}(t)\right)$
  - Relaxations weak, especially for increasing time
  - So far only small-dimensional problems solved

- Simpler model structures are more tractable, e.g.,
  - Linear dynamics, Singer & Barton JOTA 2004
  - Hammerstein-Wiener models, Kappatou et int. Mitsos arxiv 2020

Check Yourself

- What are advantages and disadvantages of single shooting methods compared to simultaneous method?
- How does the multiple shooting method remedy the problem?
- What are the disadvantages of multiple shooting compared to single shooting?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

Dynamic optimization: path constraints and switching structure

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Dealing with (Path) Constraints in Shooting Methods

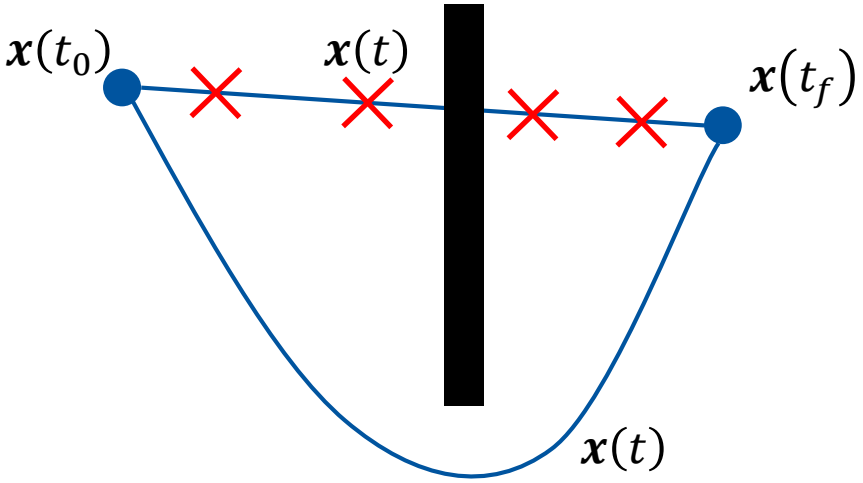
- Path constraints $\boldsymbol{g}_{\mathrm{P}}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0}$ are typically enforced on finite # times
  $$\boldsymbol{g}_{\mathrm{p}}\big(\boldsymbol{x}(t_k, \boldsymbol{p}), \boldsymbol{u}(t_k, \boldsymbol{p})\big) \leq 0, k = 0, \ldots, N$$
- Violations between points possible
- Example: Trajectory planning with obstacle

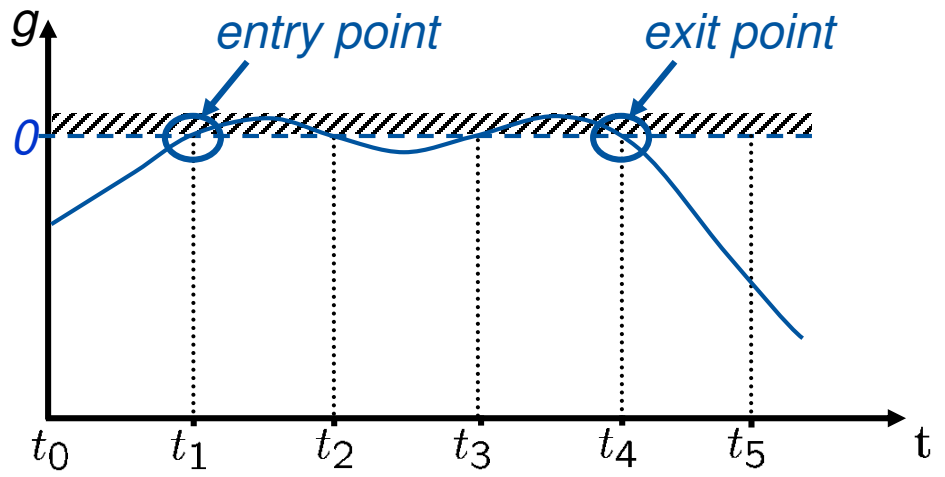
$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \quad \Phi\big(\boldsymbol{x}(t_f)\big) \qquad \text{Objective}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \qquad \text{Model}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \qquad \text{Initial conditions}$$

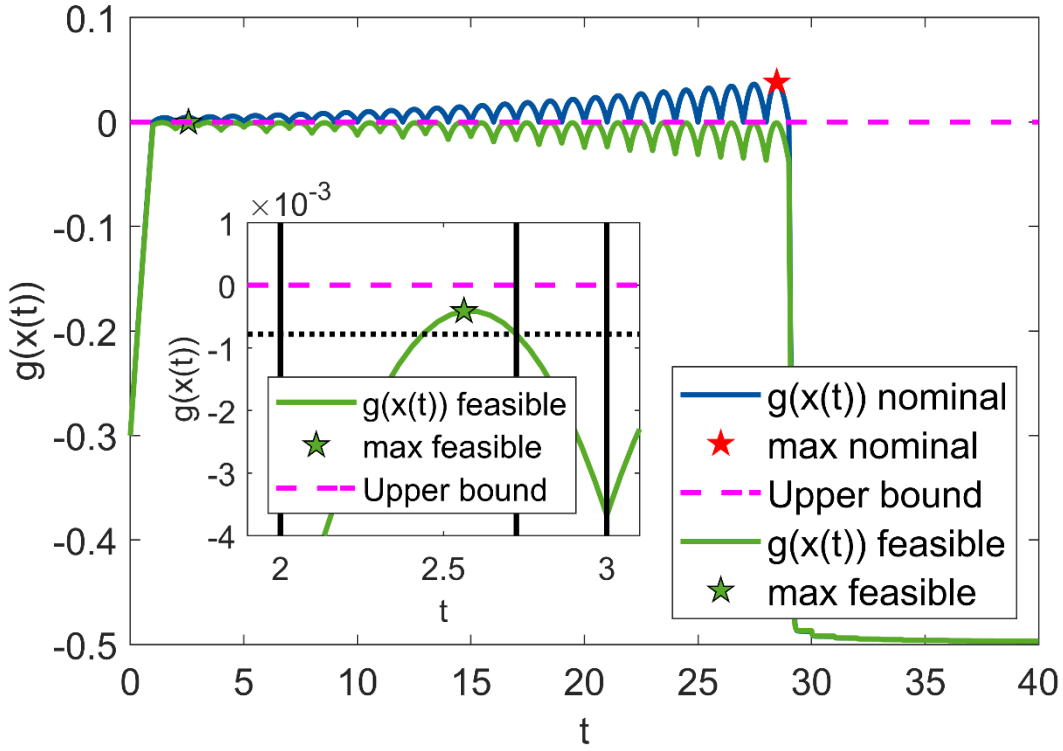
$$\boldsymbol{g}_{\mathrm{P}}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0},\ t \in [t_0, t_f] \qquad \text{Path constraints}$$

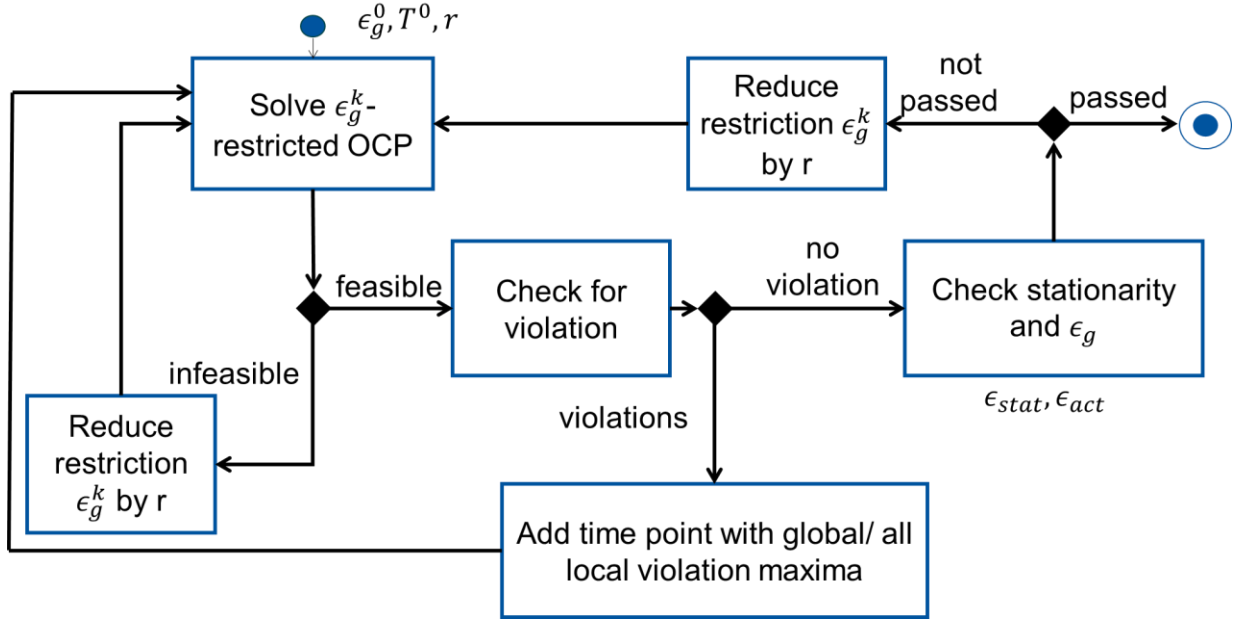




Guaranteed Feasibility of Path Constraints by Semi-Infinite Programming Methods

- Path constraint $g(x(t)) \leq 0$ for a given example



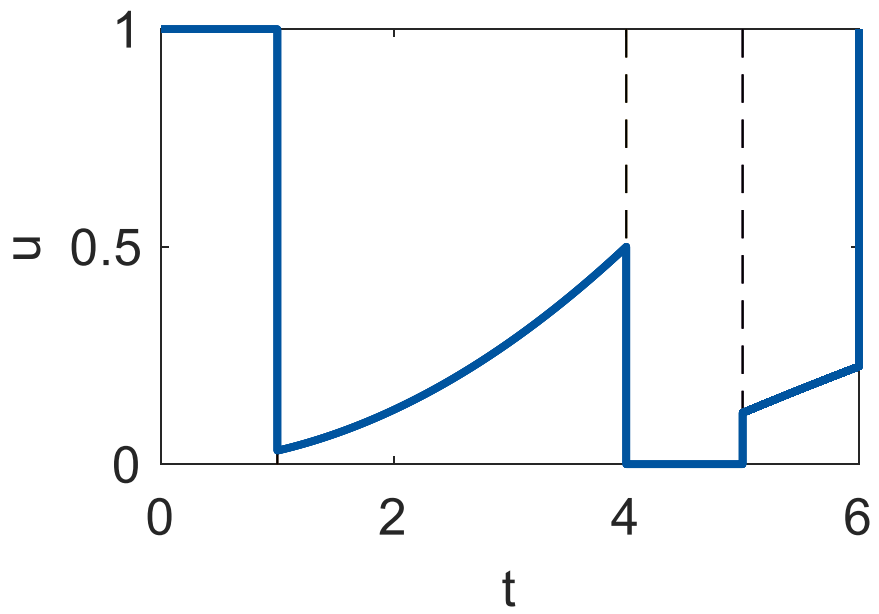


- Use techniques of semi-infinite programming (SIP) to deal with path constraints
- Guaranteed feasibility of path constraint obtained in finitely many iterations [1]

[1] Jun Fu, Johannes M.M. Faust, Benoît Chachuat, Alexander Mitsos. *Automatica*, Volume 62, 2015

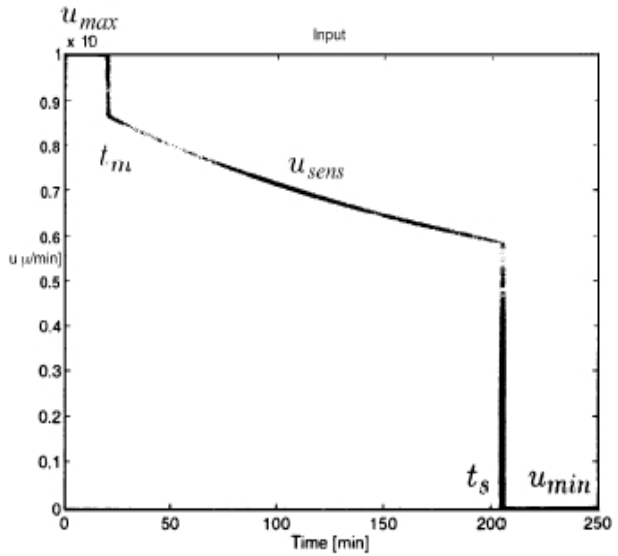
Switching Structure of the Control Variables

- 5 possible types of arcs
  - $u(t)$ at upper bound — u_{max}
  - $u(t)$ follows path constraint — u_{path}
  - $u(t)$ at lower bound — u_{min}
  - $u(t)$ not determined by constraint — u_{sens}
  - $u(t)$ active at isolated points („touch-and-go points“) — u_{touch}

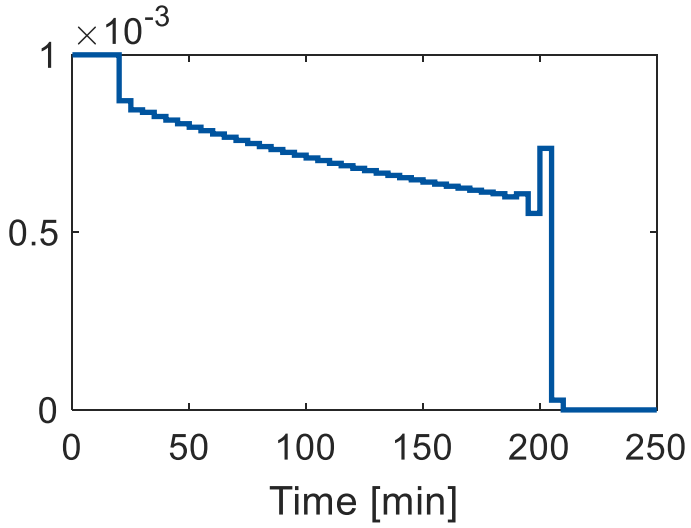


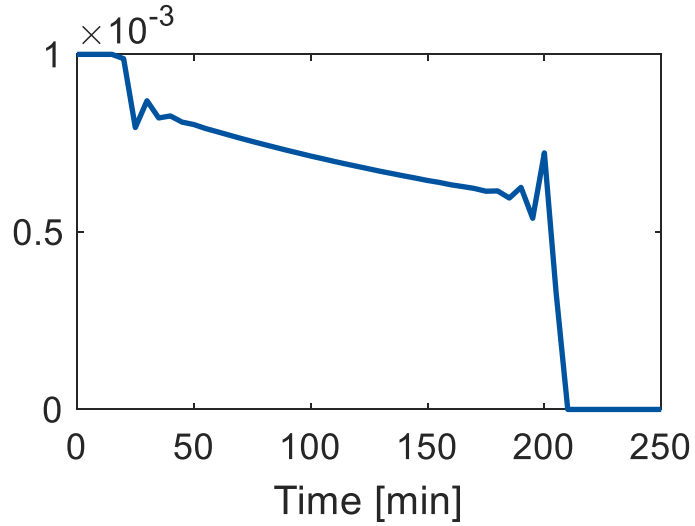
- At switching times the optimal controls can be discontinuous.
- The switching structure and times are usually not known.

Example: Semi-Batch-Reactor – Analytical and Numerical Solution



Analytical solution



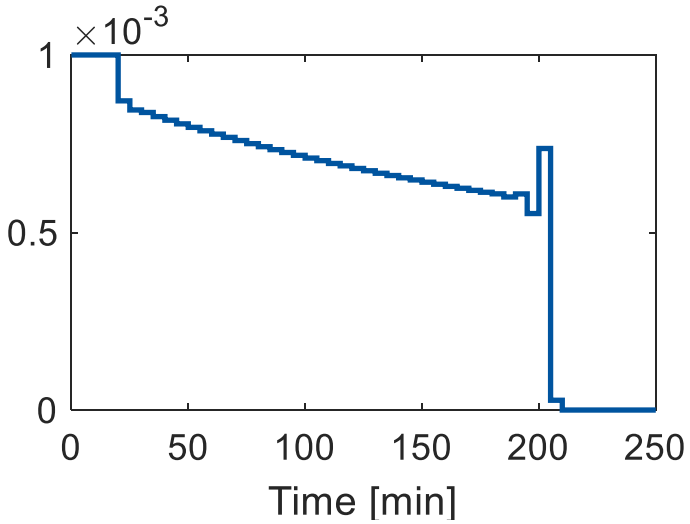


Typical numerical solutions

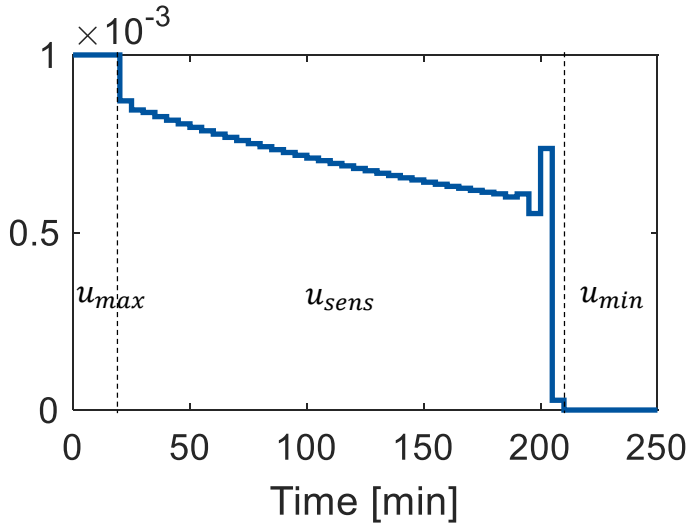
- Optimal discretized solution over-parameterized near the switching time-points
- Switching time-points not exactly met.
- Switching structure provides important insight into the process

Srinivasan et al., 2003

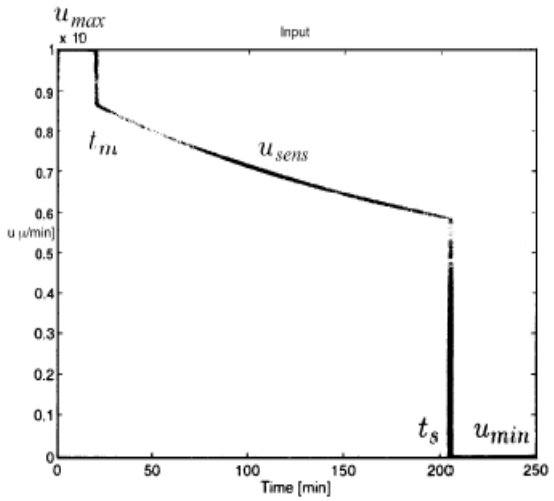
Structure Detection and Reparametrization

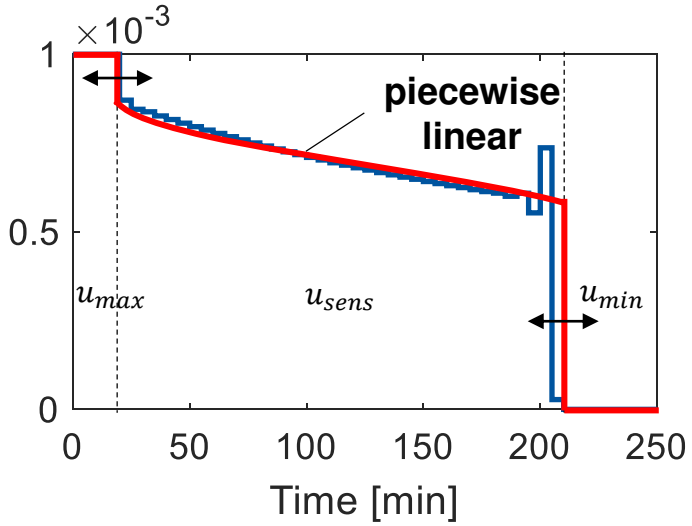


1. Calculate an approximate solution



2. Analyze the NLP results to determine switching structure





3. Fix the switching structure. Recalculate with the interval lengths as degrees of freedom

Check Yourself

- What are complications of path constraints?
- How does the switching structure influence the optimization results?

Applied Numerical Optimization

Prof. Alexander Mitsos, Ph.D.

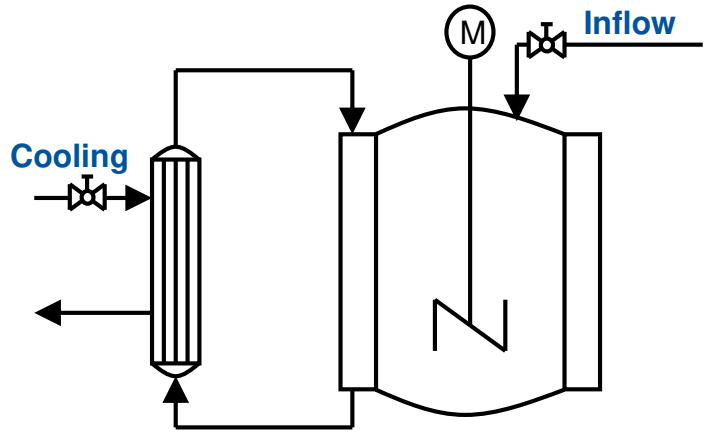
Dynamic optimization: Nonlinear Model Predictive Control (NMPC)

Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Applications in Process Systems Engineering

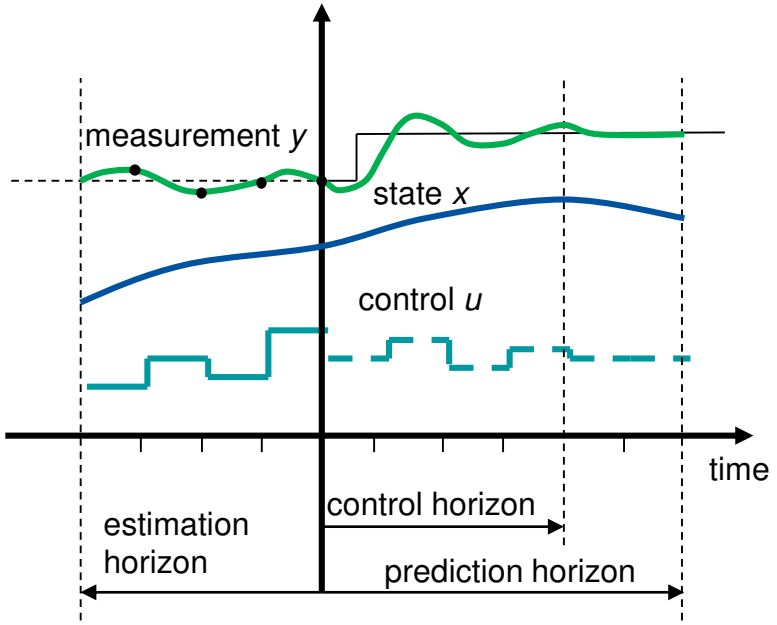
Offline Applications

- Determination of operational strategy during process design
- Simultaneous optimization of design and operation
- Improvement of operation for existing plants



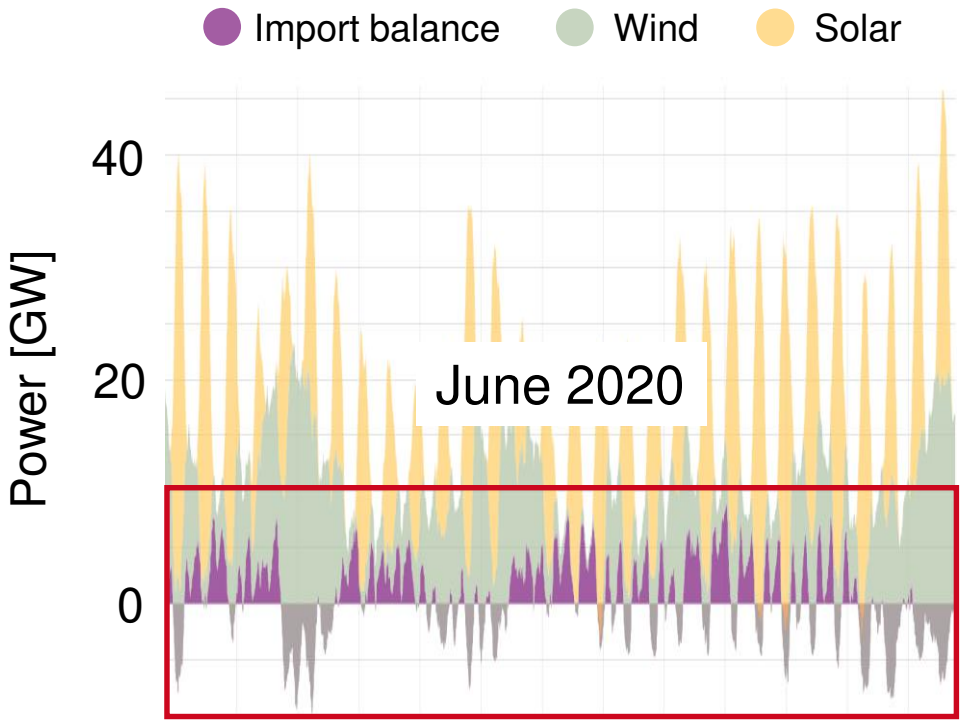
Online Applications

- Model-predictive control (MPC)
- Trajectory optimization (D-RTO)
- Application over moving horizons



Divergence of Production and Utilization of Power in Germany

Power production and export in Germany

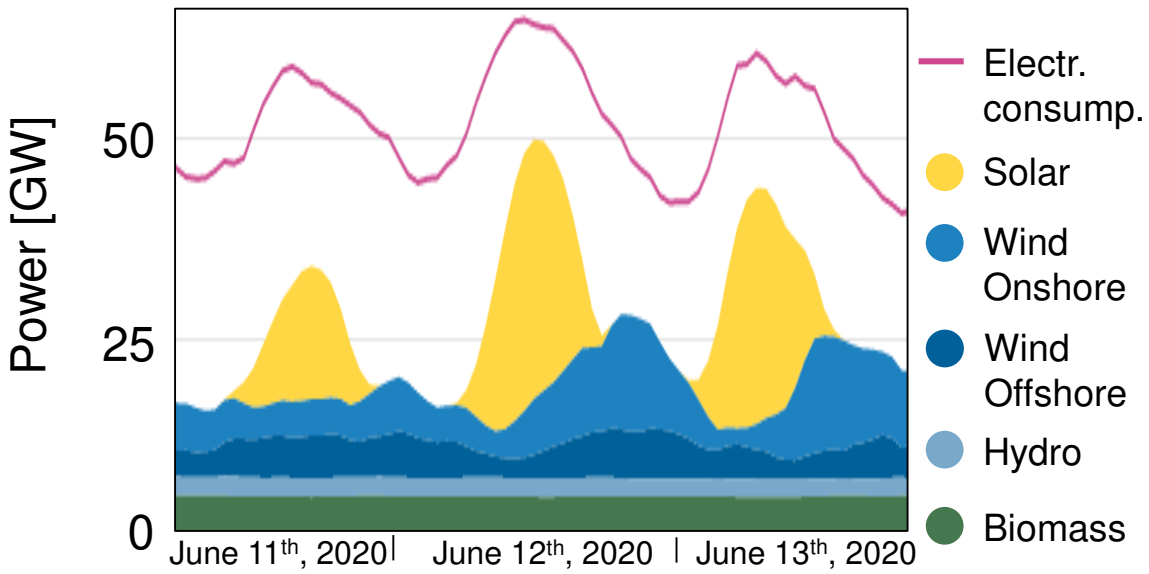


Fraunhofer https://www.energy-charts.de/power_de.htm

- Power-surplus usually exported (often at low prices)

Production of power from renewables

(three exemplary days)



https://www.agora-energiewende.de/de/themen/-agothem-/Produkt/produkt/76/Agorameter/

- Day-to-day variation in power production from renewable sources

Storage and reasonable utilization of the available energy

Applied Numerical Optimization
Prof. Alexander Mitsos, Ph.D.

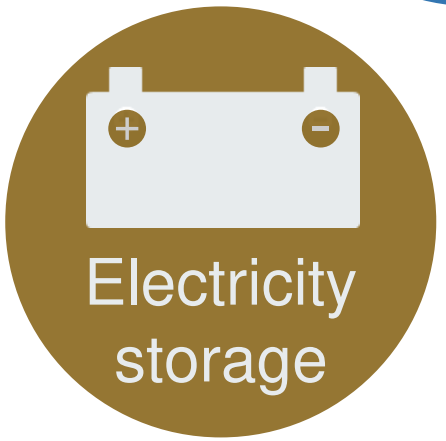
Aachener Verfahrenstechnik | RWTH AACHEN UNIVERSITY

Power-to-X

Demand side management (DSM)
("Power-to-Dynamic Processes")

Addressing system dynamics by ...

- Oversizing techniques
- Optimal control



Power-to-**Fuels**

Power-to-**Battery Storage**

Power-to-**NH_3-to-Power**

Power-to-**CAES-to-Power**

Power-to-**Cl_2 electrolysis**

Power-to-**Flex ASU**

Electricity storage
("Power-to-Y-to-Power")

- Chemical
- Mechanical
- Thermal

e-Production
("Power-to-Green Products")

- New processes and process routes
- Optimization-based process design

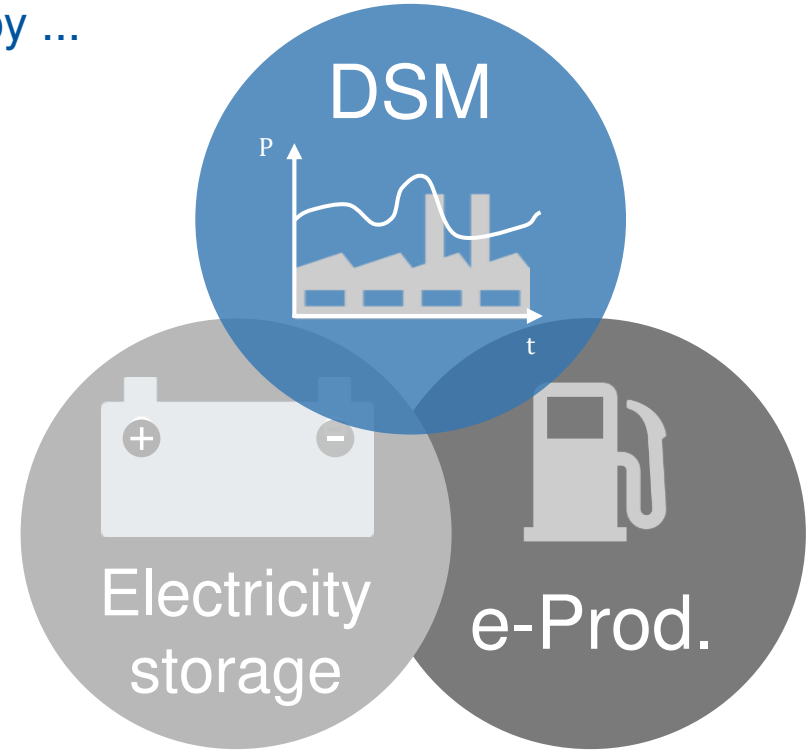
Burre, J., et int., Mitsos, A. (**2020**). *Chemie Ingenieur Technik*, 92(1-2), 74-84.

Power-to-X

Demand side management (DSM)
("Power-to-Dynamic Processes")

Addressing system dynamics by ...

- Oversizing techniques
- Optimal control



Power-to-Fuels | Power-to-Battery Storage

Power-to-NH_3-to-Power | Power-to-CAES-to-Power

Power-to-Cl_2 electrolysis | **Power-to-Flex ASU**

Electricity storage
("Power-to-Y-to-Power")

- Chemical
- Mechanical
- Thermal

e-Production
("Power-to-Green Products")

- New processes and process routes
- Optimization-based process design

Integrated Scheduling and Control for Air Separation: Setup

Air Separation

- High energy demand: 2-5% of industrial electricity demand
- Highly integrated process

Implementation

DyOS

python

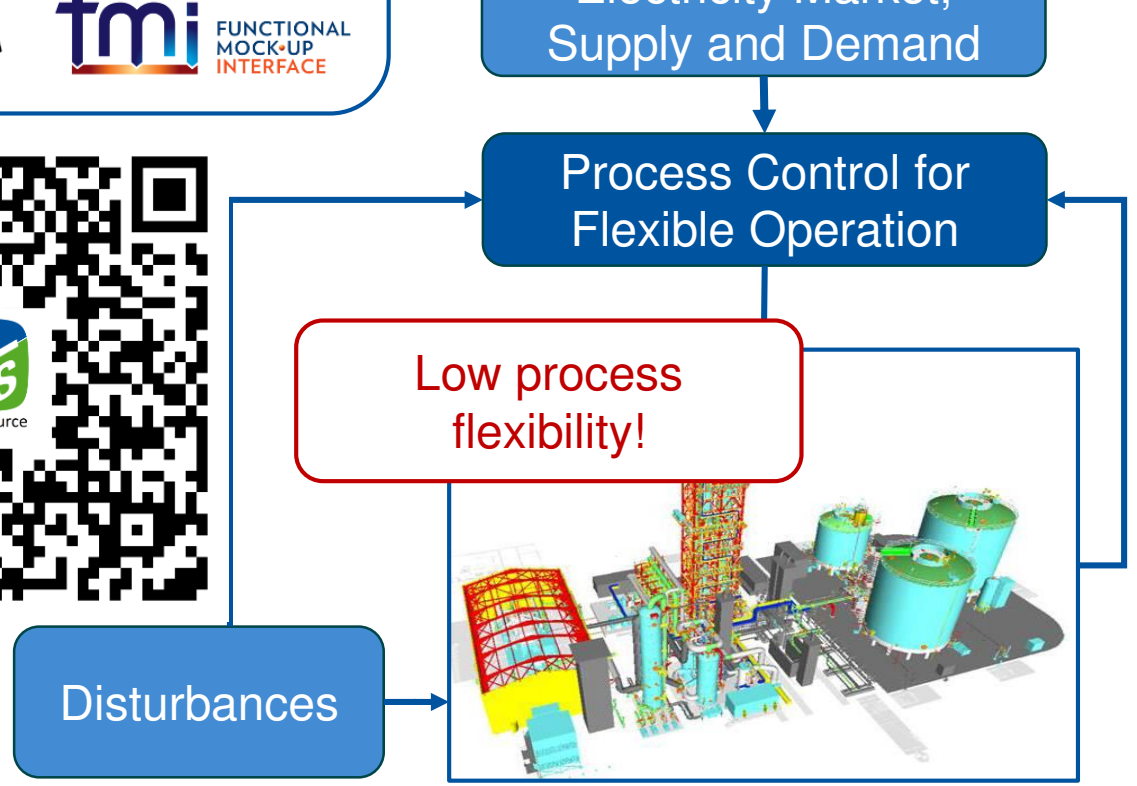
MODELICA

fmi FUNCTIONAL MOCK-UP INTERFACE

Operation Conditions

- Guarantee purity
- Guarantee production rate
- Use mass holdup for flexibility in operation
- Reject disturbances

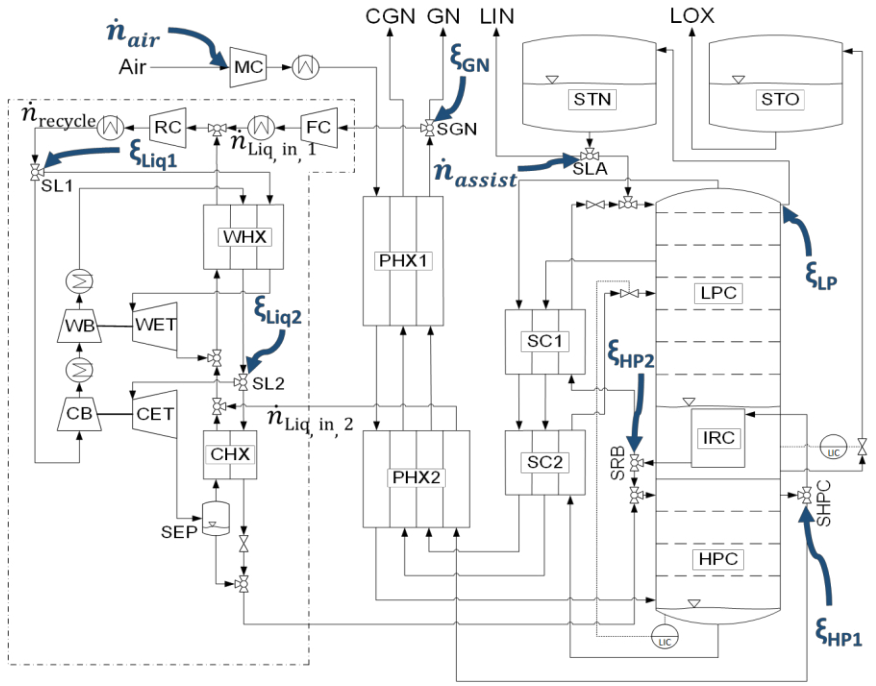
Open Source



Caspari, et int., Mitsos (**2018**). *IFAC-PapersOnLine*, *51*(20), 295-300.
Schäfer, et int., Mitsos (**2018**). *CACE* (Vol. 44, pp. 547-552). Elsevier.

A Flexible Air Separation Process Controlled by eNMPC

ASU with integrated liquefaction cycle

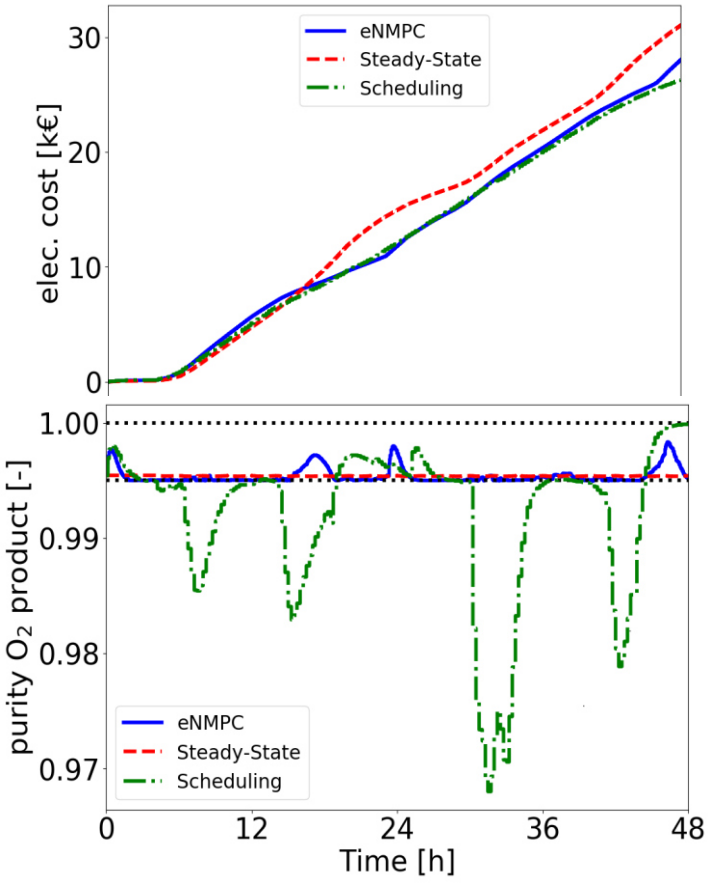


- **high flexibility** due to high liquid production
- large-scale process model

Economic Model Predictive Control (eNMPC)

- improvement: 14%, similar to unrealistic quasi-stationary schedule
- No constraint violation

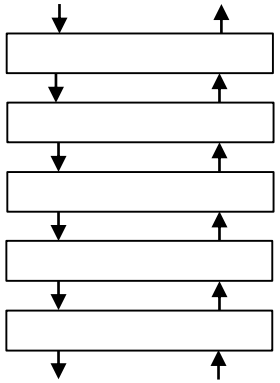
High computational effort for optimal control



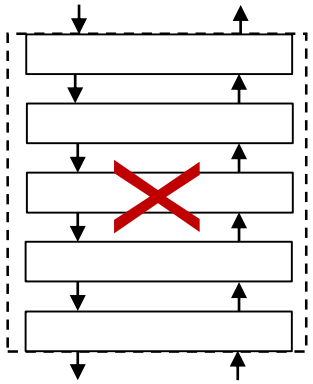
Caspari, et int., Mitsos (**2019**). *AIChE Journal*, *65*(11).

From the Classic Compartment to the Hybrid Compartment

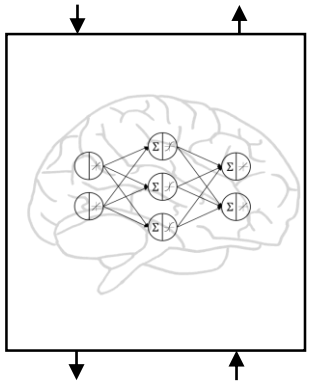
Stage-to-stage calculation



Classic compartment



Hybrid compartment



I. Aggregation of dynamics

- Compartment balance + stationary stages
- ➢ Exact steady-state
- ➢ Significant reduction of differential equations with minor influence on the dynamics

II. Substitution of inner MESH-equations

- Flash equations still included => **bottleneck!**
- ➢ Substitute stationary stage-to-stage calculations with explicit surrogate model
- ➢ Artificial neural networks as continuous surrogate model

➔ > 95% CPU time savings while staying highly accurate!

Schäfer, et int., Mitsos (**2019**). *AIChE J.* ,65.

Check Yourself

- What is the relation of NMPC to dynamic optimization?
- What are opportunities and challenges of NMPC?