Highlights:

- Nonsmooth DAE without approximation of complementarity constraints by reformulation.
- Embedded NLP is replaced by its KKT conditions yielding a nonsmooth DAE system.
- DAE solver provides adjoint sensitivities for optimization of upper-level problem.
- Illustrative examples stemming from phase equilibrium assumption and dynamic flux balance analysis approach.
- The accompanying software to the article is made available open-source.

Direct single shooting for dynamic optimization of differential-algebraic equation systems with optimization criteria embedded

Tobias Ploch[a], Jens Deussen[b], Uwe Naumann[b], Alexander Mitsos[c,a,d], Ralf Hannemann-Tamás[a,*]

[a]*Process Systems Engineering (AVT.SVT), RWTH Aachen University, 52074 Aachen, Germany*
[b]*Software and Tools for Computational Engineering, RWTH Aachen University, 52062 Aachen, Germany*
[c]*JARA-CSD, 52056 Aachen, Germany*
[d]*Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany*

Abstract

Differential-algebraic equation systems with embedded optimization criteria (DAEOs) arise in a number of applications, most prominently dynamic models of separation processes based on phase equilibrium and microbial transformation processes modeled via the dynamic flux balance analysis approach.

We consider the optimization of DAEOs. To this end, we first reformulate the DAEO using first-order optimality conditions yielding a nonsmooth DAE system. This solution approach tracks a stationary point of the embedded optimization problem that may not be globally or locally optimal on the total time horizon for non-convex problems.

We use adjoint sensitivity analysis for the resulting nonsmooth DAEs for

*Corresponding author: R. Hannemann-Tamás, E-mail address: ralf.hannemann@rwth-aachen.de

gradient-based optimization in a direct single-shooting approach. We apply the solution method to the optimization of a single flash unit, calculation of an optimal start-up scenario of a rectification column, and the optimization of biomass growth in a microbial transformation process.
Keywords: dynamic optimization, bilevel optimization, nonsmooth dynamic systems, Karush-Kuhn-Tucker conditions, dynamic flux balance analysis, dynamic phase equilibrium

---

1. Introduction

A differential-algebraic equation system with optimization criteria embedded (DAEO) is a dynamic system where the solution of an embedded optimization problem determines all or a fraction of the algebraic variables. The motivations to use this modeling approach can be diverse. Among the most important examples in process systems engineering is the dynamic modeling of separation processes assuming thermodynamic equilibrium between coexisting phases (e.g., vapor-liquid equilibrium). Phase equilibrium is found at the (global) minimum of the Gibbs free energy [1, 21]. In this case, minimization of the Gibbs free energy leads to a differential-algebraic equation (DAE) system with an embedded nonlinear program (NLP) [10, 3, 32], i.e., a DAEO. Another example is the dynamic flux balance analysis (DFBA) approach [20, 24, 35]. Here, the formulation of the stoichiometry of metabolic reaction networks leads to a linear equation system that is often underdetermined. Additional inequality constraints are imposed to narrow the possible flux distributions inside the cell. The lack of knowledge with respect to the flux distribution is overcome by assuming that the cells behave optimally

with respect to some cell-specific optimization criterion such as maximum cell growth. In this way, one particular flux distribution is obtained as the solution of an embedded optimization problem with a linear or nonlinear objective function subject to linear equality and inequality constraints [20, 24]. Finally, the coupling of the intracellular fluxes to the dynamics of the reaction medium yields a DAEO system. Hence, all DFBA models are DAEOs. The DFBA modeling approach is particularly interesting as it allows simulation of microbial transformation processes under varying environmental conditions [15, 11, 16].

In this work, we present the dynamic optimization of DAEOs. Mathematically, this leads to a bilevel dynamic optimization problem. The DAEO formulation comprises the lower level problem that is solved to determine the algebraic variables. The upper level problem minimizes the cost function, e.g., an economic objective for optimal control or a least-square formulation in the context of parameter estimation.

Many popular approaches for optimization of DAEOs [30, 29, 15, 3, 6, 7, 5] are based on a common first step, that is, the substitution of the embedded optimization problem with its first-order optimality conditions, also known as Karush-Kuhn-Tucker (KKT) conditions. The local nature of the KKT conditions does not allow a conclusion about global optimality. Consequently, the exact solution of the original DAEO is only guaranteed for a particular class of embedded optimization problems (i.e., convex NLPs). Otherwise, a stationary point is tracked that may be globally optimal, locally optimal or even not optimal at all. In addition, it may change from a global optimum to a local optimum during dynamic simulation. The KKT embedding

reformulation introduces one complementarity condition for each inequality constraint of the lower-level problem (i.e., $g_i \cdot \lambda_i = 0; g_i \geqslant 0, \lambda_i \geqslant 0$, where g_i is the ith lower-level inequality[1] and λ_i the respective Lagrange multiplier) yielding a nonsmooth DAE system.

Based on this reformulation, the solution methods for solving the nonsmooth dynamic optimization problems may be divided into simultaneous and sequential approaches [3]. The simultaneous optimization strategy is based on full discretization of variables and equations leading to a single large-scale NLP that can be solved by appropriate solvers such as IPOPT [34]. In the context of DAEOs, the full discretization yields a mathematical program with complementarity constraints (MPCC) [3]. In their natural form, these MPCCs are hard to solve by classical NLP solvers because the complementarity constraints do not satisfy a constraint qualification such as the linear independence constraint qualification (LICQ) or Mangasarian-Fromovitz constraint qualification (MFCG) [3]. The MPCC can be approximated by reformulations to overcome this issue. A common approach is the relaxation of the complementarity constraint (i.e., $g_i \cdot \lambda_i = \epsilon, \epsilon > 0$) that requires solving a sequence of NLPs with ϵ approaching zero [3]. Another reformulation moves the complementarity conditions from the constraints to the objective function [3]. This way, a violation of the complementarity conditions is penalized. With an appropriate choice of the penalty function, only a single NLP needs to be solved [3]. Using the simultaneous approach,

[1]Note that the inequality constraints g_i depend on the variables and parameters of the system, actually we have $g_i = g_i(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p})$, where $\boldsymbol{y}_d(t)$, $\boldsymbol{y}_a(t)$ are the time-dependent differential and algebraic variables, respectively, t is the time and $\boldsymbol{p}$ are time-invariant parameters.

Raghunathan et al. [30] consider the optimization of distillation columns with binary and ternary mixtures in phase equilibrium using full discretization based on Radau collocation, penalty-based reformulation of the MPCC and an interior point method to solve the resulting NLP. Other contributions apply the optimization method to the parameter estimation of fermentation kinetics of *Saccharomyces cerevisiae* [29], to the optimization of fed-batch fermentation strategy for *S. cerevisiae* [15], to the nonlinear model predictive control of fed-batch process with *S. cerevisiae* [6], and to the optimization of biopharmaceutical production in *Pichia pastoris* [7]. In these cases, the DFBA approach is used to model the reaction kinetics leading to a DAEO. The simultaneous solution method for optimization of DAEOs requires a variable step-size discretization for an accurate detection of switching points of the complementarity conditions [2]. However, this also increases the problem complexity. In addition, the size of the discretized NLP grows with the number of complementarity conditions. The reformulated and relaxed Gibbs free energy minimization can be formulated with a single complementarity condition (per equilibrium tray). In contrast, the number of those conditions grows with the size of the metabolic network considered in the DFBA approach making the simultaneous approach restrictive to small to medium-scale metabolic networks [18]. To summarize, the limitations when optimizing DAEOs using the simultaneous solution approach are (i) costly detection of switching points of the complementarity conditions, and (ii) size restriction with respect to the number of complementarity conditions.

To overcome these issues, we are interested in solving DAEO optimization problems using the sequential approach that is based on the interaction of a

numerical integration algorithm with an NLP solver. To this end, only the control variables are discretized and the dynamic system is integrated using a solver for nonsmooth DAEs. The values of the objective and constraints and the respective gradients are passed to an NLP solver that updates the control variable values. This iterative procedure is repeated until convergence within a predefined optimality tolerance. The sequential approach has been used in literature to optimize DAEOs as detailed in the following. Ritschel et al. [31] use the sequential approach to solve two optimal control problems based on a UV-flash formulation. To this end, they model the vapor-liquid equilibrium (VLE) as a DAEO comprising the maximization of entropy and use a reformulation based on the KKT conditions. Their study is limited to the case where both phases are always present thereby avoiding the nonsmoothness caused by the complementarity conditions. Caspari et al. [5] solve optimization problems based on the thermodynamic equilibrium assumption with vanishing and appearing phases by smoothing the complementarity constraint. In an approach to make parameter estimation available for genome-scale DFBA models, Leppävuori et al. [18] use a sequential optimization strategy based on direct solution of the embedded optimization problem with an LP solver and calculation of sensitivities via differentiation of the discretized equation system. While this tailored approach seems to work for parameter estimation of genome-scale DFBA models, it suffers from the known limitations of the direct solution approach. In particular, active set changes cannot be tracked correctly [18] and attempted integration steps may lead to infeasibility of the embedded LP [13]. Scott et al. [33] also use the sequential approach for optimization of DAEOs to calculate optimal fed-

batch scenarios for microbial transformation processes based on the DFBA modeling approach. To circumvent the need for a nonsmooth integration algorithm, they use a penalty formulation similar to the ones used in simultaneous optimization of MPCCs to eliminate the inequality constraints of the lower-level problem [33]. They solve simulation and optimization examples comprising DFBA models in gPROMS.

In this work, we build on the sequential approach to optimize DAEO systems. We formulate first-order optimality conditions of the embedded optimization problem to obtain a nonsmooth DAE system [26]. As previously discussed, we cannot guarantee optimality of the lower level problem. Then, we use our numerical framework [12] to solve the nonsmooth dynamic systems and calculate the required gradients based on adjoint sensitivity analysis. Finally, we use an NLP solver for optimization of the upper level problem. We consider the exact nonsmooth system without any approximation by reformulation. Thereby, no smoothing parameter has to be chosen and only a single dynamic optimization problem needs to be solved.

The structure of this work is as follows. In the next section, we formally introduce the DAEO formulation and briefly summarize its reformulation yielding a nonsmooth DAE system. Furthermore, we describe how the nonsmooth DAE is solved and the choice of an NLP solver for optimization. The subsequent section presents three relevant case studies: dynamic optimization of a single flash unit, optimal start-up scenario of a distillation column and optimal batch conditions for biomass growth in an aerobic microbial transformation process based on the DFBA modeling approach. Finally, we end with concluding remarks. The software for dynamic op-

timization of DAEOs will be made available on the public git repository `http://permalink.avt.rwth-aachen.de/?id=646019`.

2. Methods

2.1. Theoretical background and model formulation

We consider DAEOs of the following form

$$\dot{\boldsymbol{y}}_d(t) = \boldsymbol{f}(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}), \quad \text{with} \quad \boldsymbol{y}_d(t_0) = \boldsymbol{y}_{d,0}(\boldsymbol{p}) \tag{1a}$$

$$\boldsymbol{y}_a(t) \in \arg\min_{\hat{\boldsymbol{y}}_a \in \mathbb{R}^{n_a}} h(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) \tag{1b}$$

$$\text{s.t. } 0 = g_k(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}), \quad k = 1, \ldots, n_e \tag{1c}$$

$$0 \leqslant g_k(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}), \quad k = n_e{+}1, \ldots, n_g \tag{1d}$$

where $\boldsymbol{y}_d(t) \in \mathbb{R}^{n_d}$ and $\boldsymbol{y}_a(t) \in \mathbb{R}^{n_a}$ are the differential and algebraic variables, respectively; the differential equations are given by a function $\boldsymbol{f} : \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \times [t_0, t_f] \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_d}$; the vector $\boldsymbol{p} \in \mathbb{R}^{n_p}$ comprises all model parameters; t_f is the final time and $t \in [t_0, t_f]$ is the independent time variable. Note that additional algebraic equations and algebraic variables outside the optimization problem are possible, if the additional algebraic variables can formally be eliminated by means of the additional algebraic equations. Then, they do not influence the solution method described below. For brevity, they are not included in formulation (1). The algebraic variables are given as a solution point of an embedded NLP comprising Eqs. (1b)–(1d), i.e., a vector $\boldsymbol{y}_a$ that satisfies the constraints and further minimizes an objective function $h : \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \times [t_0, t_f] \times \mathbb{R}^{n_p} \to \mathbb{R}$. The functions $\boldsymbol{f}, h, g_k, k = 1, \ldots n_g$ are assumed to be sufficiently smooth. What sufficiently smooth means depends

on the maximal order κ of the employed numerical integrator and will be further discussed in the next section.

We use first-order optimality conditions with time-dependent Lagrange multipliers $(\lambda_1, \ldots, \lambda_{n_g})^T = \boldsymbol{\lambda}(t) \in \mathbb{R}^{n_g}$ to reformulate the DAEO (1) into a nonsmooth DAE system [26]. With this approach some limitations arise. First, the local optimality conditions do not allow a conclusion about global optimality of the embedded optimization problem. Consequently, embedded NLPs with multiple local optima should be treated with caution. In addition, the solution of the embedded NLP may be non-unique, i.e., there may be multiple solution points satisfying the KKT conditions. The non-uniqueness arises particularly with embedded LPs. Possible remedies include lexicographic optimization, i.e., sequentially solving multiple LPs [16, 9], or reformulation into a strictly convex QP [26]. In addition, the algebraic equation system describing the time-dependent KKT conditions may become unsolvable. An example of this case is the dynamic modeling of system in liquid-liquid equilibrium [25]. However, for many problems of interest, it is feasible for DAEOs to use KKT embedding as reformulation.

For the reformulation, we divide the total time horizon $t \in [t_0, t_f]$ into open time intervals (t^{k-1}, t^k), $k = 1, 2, \ldots K$, where $t^K = t_f$, also called *stages*. On each stage, the DAEO behaves smoothly, i.e., nonsmooth events only occur at $t^1, \ldots, t^{K-1}$. Note that we use superscripts to indicate the association with the respective *stage*, i.e., $\boldsymbol{y}_d^k(t)$ refers to the differential variables at time $t \in (t_{k-1}, t_k)$ in the kth stage. For each of these stages, we have to select which of the inequality constraints (1d) will be added as algebraic equations to the reformulated DAE.

Readers familiar with the theory of constraint optimization, recall the definition of the active set (see [23, Definition 12.1])

$$\mathcal{A}(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) = \{j \mid g_j(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) = 0, \quad 1 \leqslant j \leqslant n_g\}\,. \tag{2}$$

The active set could in a straightforward KKT embedding approach determine which of the inequality constraints (1d) will be added as algebraic equations to the respective stage of the transformed differential-algebraic equation system. However, if the LICQ is violated, then this choice would lead to an over-determined DAE system. This situation can typically arise in linearly-constrained mathematical programs like LPs or QPs. Hence, for LPs or QPs we propose an alternative set from which we choose the algebraic equations to be added. We call this *the set of computationally active constraints* $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$. For $j \in I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$ we will add $g_j(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}) = 0$ to the DAE. Note that all indices of the equality constraints (1c) are contained in $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$ and that $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$ is a subset of the active set, i.e., we have the inclusions

$$\{1, \ldots, n_e\} \subset I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) \subset \mathcal{A}(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) \subset \{1, \ldots, n_g\}.$$

In general, the constraints that are in the active set but not in the set of of computationally active constraints are only weakly active, i.e., the corresponding Lagrange multipliers are zero. If an active set-solver is used to solve the linear program or quadratic program, the set $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$ can be chosen as the complement of the basic feasible set at the solution point. The latter approach is implemented for embedded LPs or QPs in the DAEO tool-

box [27], which we will partly use in our numerical experiments. However, for general nonlinear programs with nonlinear inequality constraints, one will tpyically choose $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) = \mathcal{A}(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$.

The set of computationally active constraints is constant with respect to t on the open intervals (t^{k-1}, t^k) for $k = 1, \ldots, K$, i.e., $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) \equiv I_a^k$ for some time-invariant index set I_a^k. For stage k, i.e., $t \in (t^{k-1}, t^k)$, we get

$$\dot{\boldsymbol{y}}_d^{\,k}(t) = \boldsymbol{f}(\boldsymbol{y}_d^k(t), \boldsymbol{y}_a^k(t), t, \boldsymbol{p}), \tag{3a}$$

$$\boldsymbol{y}_d^k(t^{k-1}) = \boldsymbol{y}_d^{k-1}(t^{k-1}), \tag{3b}$$

$$0 = g_j(\boldsymbol{y}_d^k(t), \boldsymbol{y}_a^k(t), t, \boldsymbol{p}), \qquad j \in I_a^k, \tag{3c}$$

$$0 = \lambda_j^k(t), \qquad j \notin I_a^k, \tag{3d}$$

$$0 = \nabla_{\boldsymbol{y}_a} L(\boldsymbol{y}_d^k(t), \boldsymbol{y}_a^k(t), \boldsymbol{\lambda}^k(t), t, \boldsymbol{p}), \tag{3e}$$

$$\sigma_j^k(t) = \begin{cases} g_j(\boldsymbol{y}_d^k(t), \boldsymbol{y}_a^k(t), t, \boldsymbol{p}), & j \notin I_a^k, \\ \lambda_j^k(t), & j \in I_a^k, \end{cases} \tag{3f}$$

where Equation (3a) comprises the differential equations with parametric initial conditions since $\boldsymbol{y}_d^{k-1}(t^{k-1})$ implicitly depends on $\boldsymbol{p}$. In DAEOs, the differential variables are continuous at the transition between two stages as stated by Equation (3b). Equations (3c) - (3e) comprise the time-dependent first-order optimality conditions of the embedded NLP where (3e) denotes the derivative of the Lagrangian function

$$L(\boldsymbol{y}_d, \boldsymbol{y}_a, \boldsymbol{\lambda}, t, \boldsymbol{p}) = h(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) - \sum_{k=1}^{n_g} \lambda_k \cdot g_k(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}),$$

with respect to the algebraic variables $\boldsymbol{y}_a$. Zero crossings of one component

$\sigma^k_{j(k)}, j(k) \in \{1, \ldots, n_g\}$ of the switching function $(\sigma^k_1, \ldots, \sigma^k_{n_g})^T = \boldsymbol{\sigma}^k$ indicate changes of the active set and introduce a new stage $k+1$.

Using the KKT embedding reformulation, we obtain a nonsmooth DAE system that enables us to consider the optimization of DAEOs. We are interested in solving the following dynamic optimization problem

$$\min_{\boldsymbol{p} \in \mathbb{P}} \Phi(\boldsymbol{y}_d(t_f)) \tag{4a}$$

$$\text{s.t. Nonsmooth DAE (3)} \tag{4b}$$

$$\boldsymbol{c}(\boldsymbol{y}_d, \boldsymbol{y}_a, \boldsymbol{p}) \leqslant \boldsymbol{0}, \tag{4c}$$

where the set $\mathbb{P}$ is the domain of the parameters $\boldsymbol{p}$, Φ is the objective function, and $\boldsymbol{c}$ denotes the constraint operator of the upper-level optimization problem. Note that, in principle, more general objective function formulations are possible but for simplicity of presentation we stick to the Mayer-type objective function (4a). In the next section, we describe how our numerical framework [12] provides sensitivity information to solve the upper-level optimization task and state the required assumptions.

2.2. Numerical optimization of DAEOs

We use a sequential method for the solution of (4). Starting with an initial guess for the upper-level optimization parameters, we use an integration algorithm to evaluate the values of the objective function and the constraints and calculate their sensitivities with respect to the parameter vector $\boldsymbol{p}$, i.e., $\Phi_p := \frac{\partial \Phi}{\partial \boldsymbol{p}}$ and $\boldsymbol{c}_p := \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{p}}$. Due to the nonsmoothness of the underlying model (3), we need to apply a specialized numerical solution method for integration and sensitivity analysis. Therefore, we extend our numeri-

cal framework [12] that provides adjoint sensitivity analysis for generalized Mayer-type objective functionals[2] subject to nonsmooth DAE systems such as the reformulated DAEO model (3). Adjoint sensitivity methods are beneficial for problems with a small number of model outputs (i.e., objective functional and constraints) but many optimization parameters [4]. In order to use adjoint sensitivity we rely on the following assumptions that take into account the assumptions in [12]. Assumption 1 in [12] deals with the well-posedness of the nDAE model. The verification of the well-posedness of nonsmooth DAEs is nontrivial even for piecewise affine systems and, at the time of writing, practically impossible for general nonlinear problems (see [19]). We assume well-posedness in the following sense.

Assumption 1. *For every $\boldsymbol{p} \in \mathbb{P}$ the transformed DAE system* (3) *has a differential index 1 and has a unique solution on $[t_0, t_f]$ with a finite number of switching points where the active set changes.*

Assumption 1 implies that K, the number of total stages, is finite. Hence, this DAE is solvable by integration algorithms with appropriate event detection. In general, violations of Assumption 1 will be detected by the integration algorithm.

The regularity of the partial Jacobian of the algebraic equations (3c)–(3e) with respect to the algebraic variables has been investigated by Kojima [17, Theorem 3.5].

[2]The objective function type allows formulation of the most relevant optimization problems in chemical engineering, for example, parameter estimation and optimal control problems.

Theorem 1. *In system* (3)*, the partial Jacobian of the algebraic equations with respect to the algebraic variables is regular if and only if*

1. *the LICQ holds with respect to the set of computationally active constraints* I_a^k*; and*
2. *the projected Hessian of the Lagrangian is regular, where the projection is performed with respect to the set computationally active constraints* I_a^k[3]*.*

Consequently, if these conditions are violated in the interior of one stage (t^{k-1}, t^k), Assumption 1 is not valid anymore and the DAE solver will fail. On the other hand, if the DAE solver successfully terminates, the conditions of Theorem 1 hold in the interior of each stage (t^{k-1}, t^k). For embedded NLPs, the set of computationally active constraints is identical with the active set. According to Theorem 1, the DAE solver will fail on violations of the LICQ. Violations of the LICQ with respect to the active set occur frequently in LPs and QPs. In these cases, the DAEO toolbox that is available open-source as part of our previous publication [26] can be used to choose the set of computationally active constraints to the complement of the basic feasible set at the solution point. I.e., even if the LICQ with respect to the active set is violated, it may not be violated with respect to the set of computationally active constraints. The DAEO toolbox has three core features: (i) it allows

[3]The projected Hessian of the Lagrangian is defined as follows (cf. Nocedal and Wright [23, page 558]): Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ be a basis of the subspace $\{\boldsymbol{v} \in \mathbb{R}^{n_a} : (\nabla_{\boldsymbol{y}_a} g_j(\boldsymbol{y}_d^k(t), \boldsymbol{y}_a^k(t), t, \boldsymbol{p}))^T \boldsymbol{v} = \boldsymbol{0},\ j \in I_a^k\}$. Let $V \in \mathbb{R}^{n_a \times k}$ be the matrix the columns of which are formed by $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$. Then, the projected Hessian is defined by $V^T \nabla^2_{\boldsymbol{y}_a \boldsymbol{y}_a} L(\boldsymbol{y}_d, \boldsymbol{y}_a, \boldsymbol{\lambda}, t, \boldsymbol{p}) V$. The regularity of the projected Hessian is independent of the choice of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$.

formulation of DAEOs in form (1); (ii) it automatically derives and solves the KKT system (3c) - (3e) for a fixed set of computationally active constraints I_a^k and provides the values of the switching function $\boldsymbol{\sigma}^k$ (cf. Eq. (3f)) to the integration algorithm and (iii) it triggers the optimizer to compute a new optimal solution point and the corresponding set of computationally active constraints I_a^{k+1} when a new stage is introduced.

With the maximal order κ of the employed numerical integrator, we require smoothness of the functions f, h, and $g_k, k = 1, \dots n_g$ according to the following assumption:

Assumption 2. *The function f is $(\kappa+1)$-times continuously differentiable. $h, g_k, k = 1, \dots n_g$ are $(\kappa+2)$-times continuously differentiable, where $\kappa \in \mathbb{N}$, $\kappa \geqslant 1$.*

In order to integrate that DAE numerically, we assume that Assumption 2 holds with sufficiently large κ.

Assumption 3. *When a Runge-Kutta, Rosenbrock-type, extrapolation or BDF method of order q is used to solve the NDAE* (3) *and its adjoint system, Assumption 2 holds with $\kappa = q$.*

In our numerical experiments will use the integrator IDAS of the SUNDIALS suite [14] which uses a variable order from 1 to 5, i.e., $\kappa = 5$.

Assumption 4. *We assume that the time derivatives of the triggering components of the switching functions $\sigma_{j(k)}^k[\cdot]$ at its roots exist and are nonzero:*

$$\frac{d}{dt}\sigma_{j(k)}^k[t]\bigg|_{t=t^k} \neq 0, \quad k = 1, \dots, K-1.$$

Assumption 4 is taken over unchanged from [12] and is required to avoid division by zero in the adjoint equations.

In order to solve the dynamic optimization problem (4) numerically, it is important to detect zero-crossings of the switching functions. According to Eq. (3f) the formulation of a switching function $\boldsymbol{\sigma}^k$ depends on I_a^k and may change between two stages. Figure 1 illustrates the procedures of the

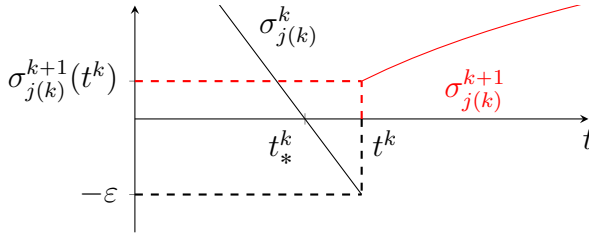


Figure 1: Illustration to explain root finding

root-polishing method used in this work and the ε-approach in Modelica used in our previous work [26], respectively. First we explain the root-polishing method.

The root-finding DAE solver detects a zero-crossing of the $j(k)$th component of $\boldsymbol{\sigma}^k$ from positive to negative at t^k_* and stops the integration[4]. Then, we virtually add a small positive increment ε to $\sigma^k_{j(k)}$ and continue the integration of stage k. Then the integrator will detect a zero-crossing of the modified $\sigma^k_{j(k)}$ and stop the integration at t^k. Now we change the stage from stage k to stage $k+1$. With the change of the active set, also the formulation of the switching function changes according to Eq. (3f). Hence, at the beginning of stage $k+1$, $\sigma^{k+1}_{j(k)}$ will have a strict positive value.

[4]Note that we exploit the capability of the root-finding DAE solver IDAS to set the direction of the zero-crossing, i.e., zero-crossings from negative to positive are ignored by IDAS.

The ε-approach in Modelica works similar. Here, we virtually add a positive increment ε to the switching condition. In this way the integration of the first stage will pass t_*^k and stop directly at t^k. At t^k, the stage is changed from stage k to stage $k+1$. Again, at the beginning of stage $k+1$, the switching function has a strict positive value.

Both approaches are implemented such that switching functions from other stages are not visible in the current stage, i.e., in the example illustrated in Figure 1, $\boldsymbol{\sigma}^k$ is only visible to the solver in the interval $[t^{k-1}, t^k]$, similarly $\boldsymbol{\sigma}^{k+1}$ is only visible to solver in the interval $[t^k, t^{k+1}]$.

An active set change occurs if either an active inequality becomes inactive (i.e., λ_j becomes zero) or vice versa. The total number of stages is unknown before simulation and depends on the parameter vector $\boldsymbol{p}$.

The extended numerical framework requires the user-supplied derivatives of the residuals of the DAE model. These derivatives are best obtained by algorithmic differentiation (AD). For an introduction to algorithmic differentiation, we refer to the book by Naumann [22]. In our implementation, we use an AD solution based on operator overloading in C++ and developed by the research group on Software and Tools for Computational Engineering (STCE) at RWTH Aachen University. The NLP solver SNOPT [8] is used to solve the dynamic optimization problem (4).

3. Illustrative examples

The following examples illustrate how the presented sequential method can be used for optimization of DAEOs. The first example deals with a flash unit comprising a single embedded NLP. The second case study calculates

an optimal start-up scenario for a rectification column with 20 trays and an equal number of embedded NLPs. In the last example, we aim to find optimal batch conditions for a microbial transformation process based on the DFBA modeling approach leading to a DAE with a single embedded quadratic program. We use our extended numerical framework with integration tolerance of 1×10^{-8}, event tolerance of 2×10^{-8} and maximum discontinuity tolerance of 1×10^{-7} [12] and the NLP solver SNOPT 7.2-4 [8] with optimality and feasibility tolerances of 1×10^{-6}. All calculations are performed on an Intel® Core™ i3-4160 with 3.6 GHz and 16 GB RAM running Windows 7 64-bit operating system.

3.1. Optimal batch vaporization

In this illustrative example, we consider an optimal batch vaporization of an equimolar binary mixture of acetone (1) and ethanol (2). The mixture is in vapor-liquid equilibrium that is found at the (global) minimum of the

Gibbs free energy. The dynamic model is given by

$$\dot{h}(t) = \frac{Q(t)}{M_V(t) + M_L(t)}, \quad h(t_0) = h_0, \tag{5a}$$

$$h(t) = \frac{M_V(t) \cdot h_V(T(t), p, \boldsymbol{y}(t)) + M_L(t) \cdot h_L(T(t), p, \boldsymbol{x}(t))}{M_V(t) + M_L(t)} \tag{5b}$$

$$\begin{aligned}(M_V(t), M_L(t), \boldsymbol{x}(t), \boldsymbol{y}(t)) \in \arg \min_{\hat{M}_V, \hat{M}_L, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}} \; & G^M = \hat{M}_V \sum_{i=1}^{C} \hat{y}_i \bar{G}_i^V(T(t), p, \hat{\boldsymbol{y}}) \\ & + \hat{M}_L \sum_{i=1}^{C} \hat{x}_i \bar{G}_i^L(T(t), p, \hat{\boldsymbol{x}})\end{aligned} \tag{5c}$$

$$\text{s.t.} \qquad M_i = \hat{M}_V \cdot \hat{y}_i + \hat{M}_L \cdot \hat{x}_i, \quad i = 1, \ldots, C, \tag{5d}$$

$$\sum_{i=1}^{C} M_i = \hat{M}_V + \hat{M}_L, \tag{5e}$$

$$\sum_{i=1}^{C} \hat{y}_i - \sum_{i=1}^{C} \hat{x}_i = 0, \tag{5f}$$

$$\hat{M}_V \geqslant 0, \quad \hat{M}_L \geqslant 0, \tag{5g}$$

$$\hat{y}_i > 0, \quad \hat{x}_i > 0, \quad i = 1, \ldots, C, \tag{5h}$$

where h is the molar enthalpy and the only differential variable. The algebraic variables are the temperature T, the molar amounts of vapor and liquid phase M_V and M_L, respectively, and the mole fractions of component i in the vapor and liquid phase y_i and x_i, respectively. Note that we adapt the formulation of the embedded optimization problem from the work by Sahlodin et al. [32]. The heat duty Q is the only input of the model and will be calculated as the solution of the upper-level optimization problem. h_V and h_L are the molar enthalpies of vapor and liquid phase, respectively, G^M is the Gibbs free energy, $\bar{G}_i^V$ and $\bar{G}_i^L$ are the partial molar Gibbs free energy of component i

in the vapor and the liquid phase, respectively, M_i is the total molar hold-up of component i in both phases that is constant in this example and C is the number of components (i.e., for the binary example $C = 2$). Equation (5a) is the energy balance equation and (5b) is the constitutive equation for the molar enthalpy h. Of the algebraic variables, all but the temperature are calculated as the solution of the embedded optimization problem (5c) - (5h) that minimizes the Gibbs free energy G^M subject to mass balance and non-negativity constraints. The molar enthalpies of vapor and liquid phase are calculated from constitutive equations of form $h_V(T, p, \boldsymbol{y})$ and $h_L(T, p, \boldsymbol{x})$, where T is the temperature, p is the pressure (the symbol of the pressure should not be confused with the parameter vector $\boldsymbol{p}$), and $\boldsymbol{y}$ and $\boldsymbol{x}$ are the vectors of mole fractions of vapor and liquid phase, respectively. Similarly, the partial molar Gibbs free energies $\bar{G}_i^V$ and $\bar{G}_i^L$ depend on temperature, pressure and mole fraction of the respective phase.

We now reformulate DAEO (5) using first-order optimality conditions of the embedded NLP as described in Section 2.1. In the two-phase region (i.e., $M_V > 0$ and $M_L > 0$), the well-known isopotential condition can be deduced from the KKT conditions [32]. In the single-phase region, however, the unmodified KKT conditions of DAEO (5) violate the LICQ , i.e., if either $M_V = 0$ (liquid only) or $M_L = 0$ (vapor only). In this situation, the KKT system becomes singular. In particular, the mole fractions of the non-existing phase can be chosen arbitrarily as long as they sum up to unity and thereby satisfy (5f). A common approach from literature to overcome this shortcoming uses a relaxation of the isopotential condition by an auxiliary variable β in the single phase region [10, 3, 32]. Thereby, the singularity is

avoided. The mole fractions of the nonexisting phase are determined by a tangent plane to the Gibbs free energy surface that is parallel to the one of the existing phase [25]. We use the described relaxation from literature to capture vanishing and reappearing phases in dynamic VLE systems [10, 3, 32]. A detailed derivation of the reformulation is omitted here but can be found in the cited literature. Using the notation $\boldsymbol{g}(\dots) = \mathbf{0}$ for the algebraic equations, we obtain the following nonsmooth equation system

$$\dot{h}(t) = \frac{Q(t)}{M_V(t) + M_L(t)}, \quad h(t_0) = h_0, \tag{6a}$$

$$g_i = y_i(t) - \beta(t) \cdot K_i(T(t), p, \boldsymbol{x}(t), \boldsymbol{y}(t)) \cdot x_i(t), \quad i = 1, \dots, C, \tag{6b}$$

$$g_{C+i} = M_i - (M_V(t) \cdot y_i(t) + M_L(t) \cdot x_i(t)), \quad i = 1, \dots, C, \tag{6c}$$

$$g_{2C+1} = h(t) - \frac{M_V(t) \cdot h_V(T(t), p, \boldsymbol{y}(t)) + M_L(t) \cdot h_L(T(t), p, \boldsymbol{x}(t))}{M_V(t) + M_L(t)} \tag{6d}$$

$$g_{2C+2} = \sum_{i=1}^{C} M_i - (M_V(t) + M_L(t)), \tag{6e}$$

$$g_{2C+3} = \sum_{i=1}^{C} y_i(t) - \sum_{i=1}^{C} x_i(t), \tag{6f}$$

$$g_{2C+4}^{\mathrm{L}} = \frac{M_V(t)}{M_V(t) + M_L(t)}, \quad g_{2C+4}^{\mathrm{VL}} = \beta(t) - 1,$$

$$g_{2C+4}^{\mathrm{V}} = \frac{M_V(t)}{M_V(t) + M_L(t)} - 1, \tag{6g}$$

$$\sigma^{\mathrm{L}\to\mathrm{VL}} = g_{2C+4}^{\mathrm{L}} - g_{2C+4}^{\mathrm{VL}}, \quad \sigma^{\mathrm{VL}\to\mathrm{L}} = g_{2C+4}^{\mathrm{VL}} - g_{2C+4}^{\mathrm{L}}, \tag{6h}$$

$$\sigma^{\mathrm{V}\to\mathrm{VL}} = g_{2C+4}^{\mathrm{V}} - g_{2C+4}^{\mathrm{VL}}, \quad \sigma^{\mathrm{VL}\to\mathrm{V}} = g_{2C+4}^{\mathrm{VL}} - g_{2C+4}^{\mathrm{V}}, \tag{6i}$$

where β is the auxiliary variable that is introduced to enable simulation in all possible phase regimes, K_i is the vapor-liquid distribution factor that

is a function of temperature, pressure and mole fractions of vapor and liquid phase, respectively. Note that no conclusion regarding global optimality of the embedded NLP can be drawn as we use local, first-order optimality conditions to reformulate the DAEO (5) into the nonsmooth DAE (6). Thermodynamically, the formulation may result in unstable phase splits referring to local but not global minima of the Gibbs free energy. Furthermore, the globally optimal solution may change from global to local optimum during dynamic simulation. Sahlodin et al. [32] advocate to use the mid function that returns the middle value of its three arguments to capture phase changes during dynamic simulation, i.e., mid $(\frac{M_V(t)}{M_V(t)+M_L(t)}, \beta - 1, \frac{M_V(t)}{M_V(t)+M_L(t)} - 1) = 0$ [32, Eqn. (39e)]. For our framework, a formulation similar to the mid function is given in Equations (6g) - (6i). Note that no direct switch from pure liquid to pure vapor is possible. The corresponding switching function has a constant value ($\sigma^{\mathrm{V}\leftarrow\mathrm{L}} = g^{\mathrm{V}}_{2C+4} - g^{\mathrm{L}}_{2C+4} = -1$) and is thus omitted. If vapor and liquid phase coexist (indicated by superscript VL), $\beta = 1$ and Equation (6b) becomes the well-known isopotential equation. A vanishing phase is captured by the respective switching condition $\sigma^{\mathrm{VL}\rightarrow\mathrm{L}}$ or $\sigma^{\mathrm{VL}\rightarrow\mathrm{V}}$, respectively. In the single-phase regime, the molar amount of the non-existing phase are equal to zero and the auxiliary variable becomes $\beta \neq 1$ and thereby relaxes the isopotential conditions. To summarize, a single algebraic equation (6g) changes for each considered phase regime. The switching between two phase regimes is detected by zero-crossings from positive to negative of the switching functions (6h) and (6i). For details regarding the reformulation, please refer to literature [10, 3, 32]. The molar enthalpy h is the only differential state of the system while the $2C + 4$ algebraic states are

$\boldsymbol{y} = [M_V, M_L, \boldsymbol{x}, \boldsymbol{y}, T, \beta]^T$. The total molar hold-ups of each component M_i and the pressure p are parameters and the heat duty Q is the only input to the model.

We aim to find the optimal heat duty Q to increase the temperature T from $T_0 = T(t_0)$ to the fixed setpoint temperature T_{set}. We choose T_0 corresponding to sub-cooled liquid and T_{set} to super-heated vapor to illustrate the handling of phase switches during optimization. The optimal heat duty profile is expected to be at the upper bound $Q_{\max}$ until the setpoint temperature is reached and zero afterwards. Therefore, we formulate a time-optimal control problem to find the minimum time required for the desired temperature increase at constant heat duty $Q = Q_{\max}$. We solve the optimization problem

$$\min_{t_f} t_f \tag{7a}$$

$$\text{s.t.} \quad T(t_f) = T_{\text{set}} \tag{7b}$$

$$Q = Q_{\max} \tag{7c}$$

$$\text{Nonsmooth DAE (6)} \tag{7d}$$

where equation (7b) states the endpoint constraint for the temperature. Note that this equality constraint of the upper-level optimization problem can be written as $T(t_F) - T_{\text{set}} \leqslant 0$ and $T_{\text{set}} - T(t_F) \leqslant 0$ to match the notation of the constraint operator in (4). We use $t_f = 8000$ s as initial guess. The results of the dynamic optimization are shown in Fig. 2. The minimum time required to heat the mixture to the desired setpoint temperature is $t_F \approx 6999$ s. The optimal solution comprises two active set changes (see Fig. 2, bottom). First,

the vapor phase appears (at $t^1 \approx 142$ s) before the liquid phase disappears (at $t^2 \approx 6932$ s). For this small example, the analytic solution is quite intuitive. The optimization framework solves it as expected while detecting both active set changes. As discussed, using the KKT-based reformulation does not guarantee global solution of the embedded optimization problem. An a-posteriori check on a fine grid, however, indicates that the global minimum of the embedded problem is found.

3.2. Optimal start-up of rectification column

The second illustrative example is based on a case study presented by Caspari et al. [5] and deals with the optimal start-up of a rectification column. The dynamic model formulation is adapted from the work of Raghunathan et al. [30]. A detailed description of the dynamic model is given in the work of Caspari et al. [5] and the equations for an equilibrium tray are given in the SI of this work. In the following, we give a short summary of the considered column configuration and the dynamic model equations with a particular focus on the differences of our modeling approach compared to the work by Caspari et al. [5].

We consider a cryogenic rectification column to separate a mixture of nitrogen, oxygen and argon. The column has a single feed stream with fixed location and a total condenser at the top. The column has 20 equilibrium trays, each comprising an embedded optimization problem as described in the previous example. The column model introduces two additional nonsmooth conditions per equilibrium tray for the calculation of leaving vapor and liquid flows, respectively [30]. The first is an overflow weir formulation that allows liquid flow from trays if the liquid hold-up is greater than a minimum hold-up

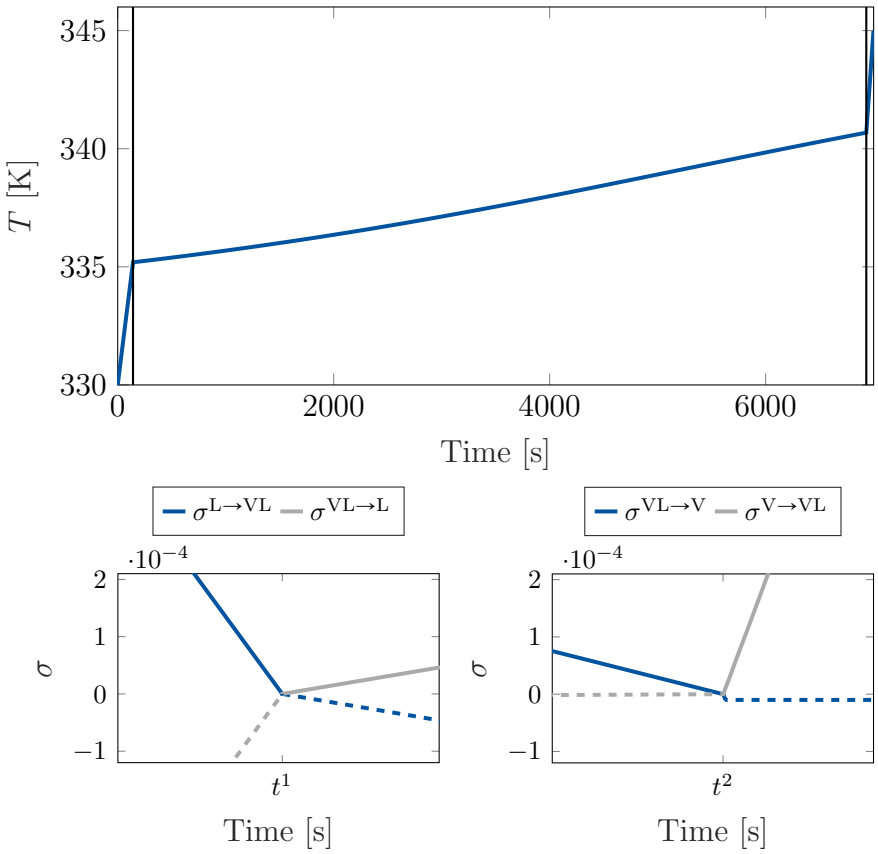


Figure 2: Temperature profile of the solution of time-optimal control problem (7) (top). The setpoint temperature $T_{\mathrm{set}} = 345$ K is reached after a minimum heating time of $t_f \approx 6999$ s. Vertical lines indicate active set changes: Vapor phase appears after $t^1 \approx 142$ s and liquid phase vanishes after $t^2 \approx 6932$ s. The bottom graphs show the values of the respective switching function triggering the switching event. Solid lines indicate switching functions that are visible to the solver.

[30]. A similar condition is formulated for vapor stream leaving a tray that exists if the vapor head is greater than zero [30]. In the previous literature, these conditions are reformulated using smoothed Fischer-Burmeister functions [5, 30]. In this work, we treat them as additional nonsmooth equations. In total, the model comprises 80 differential and 727 algebraic states.

The aim of the optimization problem is to find optimal start-up scenario

for the rectification column by minimizing the deviation of the product flow rate and the product purity from given set points, i.e.,

$$\min_{\boldsymbol{\xi},\dot{\boldsymbol{n}}_{\text{Feed}}} \int_{t_0}^{t_f} \left(w_n(\dot{n}_{\text{Product}}(t) - \dot{n}^{\text{set}}_{\text{Product}})^2 + w_x(x^{\text{N}_2}_{\text{Product}}(t) - x^{\text{N}_2,\text{set}}_{\text{Product}})^2 \right) dt \quad \text{(8a)}$$

$$\text{s.t.} \quad 0 \leqslant \xi_i \leqslant 0.8, \quad i = 1, \ldots, n_i \quad \text{(8b)}$$

$$80 \text{ mol s}^{-1} \leqslant \dot{n}_{\text{Feed},i} \leqslant 120 \text{ mol s}^{-1}, \quad i = 1, \ldots, n_i \quad \text{(8c)}$$

$$\text{Nonsmooth rectification column model,} \quad \text{(8d)}$$

where the objective function defines the integral of weighted squared deviations of the product flow rate $\dot{n}_{\text{Product}}$ and product purity $x^{\text{N}_2}_{\text{Product}}$ from their respective set points. We use the weights $w_n = 0.001$ and $w_x = 1$ to obtain same order of magnitude for both parts of the objective. The set points are $\dot{n}^{\text{set}}_{\text{Product}} = 50$ mol s^{-1} and $x^{\text{N}_2,\text{set}}_{\text{Product}} = 0.995$, respectively. The optimization variables are the parameters for the discretization of the split factor $\boldsymbol{\xi}$ and the feed flow rate $\dot{\boldsymbol{n}}_{\text{Feed}}$ with upper and lower bounds defined by constraints (8b) and (8c), respectively. In contrast to the work of Caspari et al. [5], who employed control grid adaption based on Haar wavelets, we choose a fixed piecewise constant discretization for the control variables with $N = 10$ equidistant intervals . The initial guess is $\xi_i = 0, i = 1, \ldots, N$, and $\dot{n}_{\text{Feed},i} = 80$ mol s$^{-1}, i = 1, \ldots, N$, on all intervals. The rectification column is initialized with the steady-state corresponding to the initial guess values of the optimization variables, i.e., with vapor at the dew point on all column trays.

The results of example (8) are shown in Figure 3. Despite the coarser discretization, the optimization variables in Figures 3a and 3b show a similar

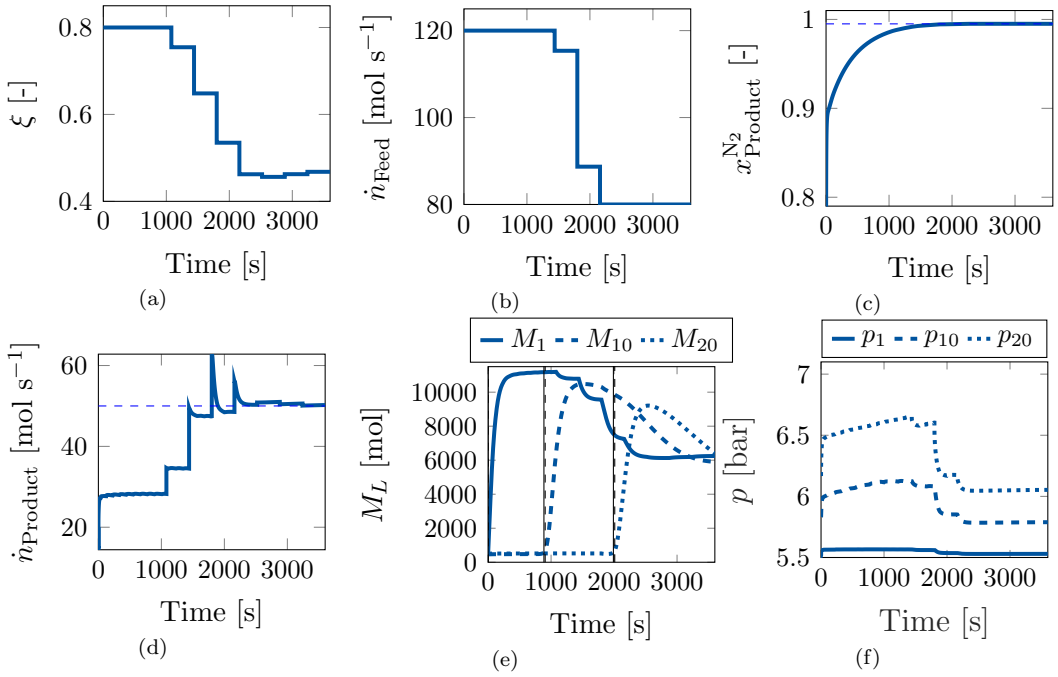


Figure 3: Results of rectification column optimization (8): optimization variables split factor ξ (a) and feed flow rate $\dot{n}_{\text{Feed}}$ (b), state variables product purity (c) and product flow rate (d) with dashed set points. (e) Liquid hold-up on trays. Vertical solid line refers to active set change, dashed line to activation of liquid outflow on the respective trays. (f) Pressure on trays.

profile as in the work of Caspari et al. [5, Figures 10a and 10b]. In the beginning, both optimization variables are at their upper bounds. The split factor is decreased after approx. 1000 s and settles at a value of around 0.46 towards the end. The feed flow rate is at its lower bound in the end. The product purity and product flow rate reach their respective set points after $t \approx 2400$ s (cf. Figures 3c and 3d). In the beginning of the start-up process, there is only vapor in the column. Starting with the top tray, VLE is gradually established on every column tray. The nonsmooth model tracks the appearance of the respective liquid phase by an active set change related

to the respective inequality constraint (solid vertical lines in Fig. 3e). After a short period of time, there is enough liquid hold-up on the corresponding column tray so that the liquid outflow also becomes active (vertical dashed lines). This sequence follows from the top to the bottom tray.

The results show that the sequential approach using the nonsmooth DAE integration framework based on adjoint sensitivity analysis and SNOPT solves relevant large-scale DAEO examples with NLPs embedded leading to equivalent results compared to literature. Note, however, that the cited work (refs. [5, 30]) applies smoothing technique to the complementarity constraints, while we solve the nonsmooth DAE system, i.e., we do not have to choose a smoothing parameter. Similar to the previous phase equilibrium example, there is again no guarantee to find the global minimum of the embedded NLP when using the KKT-based reformulation approach. For this example, an a-posteriori check again showed that the global solution is found.

3.3. Optimal batch fermentation using DFBA modeling approach

This example deals with optimal conditions for a batch fermentation process. We consider the growth of *Corynebacterium glutamicum* on a carbon source mixture of D-glucose and D-xylose based on a DFBA model from our previous work [28]. The metabolic model describes four different pathways for D-xylose uptake and consists of 50 intracellular metabolites, 59 metabolic fluxes, and six additional exchange fluxes. Maximization of the biomass growth rate is chosen as objective function for the embedded optimization problem. The dynamic model contains differential balance equations for the concentration of biomass (X), carbon sources D-glucose (GLC) and D-xylose (XYL), potential products succinate (SUCC), lactate (LAC) and acetate

(ACE), and for the overall reaction volume. The six exchange fluxes couple the metabolic model to the differential balance equations. The uptake rates for both carbon sources are modeled by Michaelis-Menten kinetics while the maximum oxygen uptake rate was not constrained to simulate aerobic growth conditions. More details on the metabolic model and its KKT embedding reformulation yielding a nonsmooth bioreactor model can be found in [26, 28].

In this illustrative example, we aim at finding optimal batch conditions for biomass growth by solving the following dynamic optimization problem

$$\min_{c_{\mathrm{GLC},0}, c_{\mathrm{XYL},0}, t_f} \quad -\frac{c_X(t_f)}{t_f} \tag{9a}$$

$$\text{s.t.} \quad 0 \leqslant 6 \cdot c_{\mathrm{GLC},0} + 5 \cdot c_{\mathrm{XYL},0} \leqslant 1 \text{ mol L}^{-1}, \tag{9b}$$

$$\text{Nonsmooth bioreactor model}, \tag{9c}$$

where $c_{\mathrm{GLC},0}$ and $c_{\mathrm{XYL},0}$ are the initial concentrations of D-glucose and D-xylose, respectively, t_f is the final batch time, the objective function aims to maximize the space-time yield with respect to biomass, constraint (10d) places an upper limit on the overall concentration of carbon atoms in the reaction medium at initial time, and the nonsmooth bioreactor model refers to the model described above. In summary, we aim to find the optimal mixture of carbon sources D-glucose and D-xylose for high biomass growth within short batch time.

In this illustrative example, we aim at finding optimal batch conditions

for biomass growth by solving the following dynamic optimization problem

$$\min_{\phi_{\mathrm{C5,0}}, t_f} \; -\frac{c_X(t_f)}{t_f} \tag{10a}$$

$$\text{s.t.} \quad c_{\mathrm{GLC}}(t_0) = \frac{(1-\phi_{\mathrm{C5,0}}) \cdot c_{\mathrm{total,0}}}{6}, \tag{10b}$$

$$c_{\mathrm{XYL}}(t_0) = \frac{\phi_{\mathrm{C5,0}} \cdot c_{\mathrm{total,0}}}{5}, \tag{10c}$$

$$c_{\mathrm{total,0}} = 1 \text{ mol L}^{-1}, \tag{10d}$$

$$\text{Nonsmooth bioreactor model}, \tag{10e}$$

where $\phi_{\mathrm{C5,0}}$ is the fraction of total carbon atoms stemming from D-xylose at initial time t_0 and t_f is the final batch time. The objective function aims to maximize the space-time yield with respect to biomass. $c_{\mathrm{GLC}}(t_0)$ and $c_{\mathrm{XYL}}(t_0)$ are the initial concentrations of D-glucose and D-xylose, respectively, constraint (10d) limits on the overall concentration of carbon atoms in the reaction medium at initial time, and the nonsmooth bioreactor model refers to the model described above. The bounds on the optimization variables are $\phi_{\mathrm{C5,0}} \in [0, 1]$ and $t_f \in [9.5, 13]$ h. Note that $\phi_{\mathrm{C5,0}} = 0$ refers to a pure D-glucose solution, while $\phi_{\mathrm{C5,0}} = 1$ indicates pure D-xylose. In summary, we aim to find the optimal mixture of carbon sources D-glucose and D-xylose for high biomass growth within short batch time.

Figure 4 shows the concentration profiles of relevant extracellular species leading to maximum space-time yield with respect to biomass under aerobic process conditions. The optimal values of the control variables are $\phi_{\mathrm{C5,0}} = 0.2$ and $t_f = 10.55$ h. In this example, the embedded optimization problem is a convex quadratic program so that the KKT-based approach provides an

exact reformulation of the embedded optimization problem and the solution refers to the global minimum. In this example, the LICQ holds and the projected Hessian of the Lagrangian function is regular as no violations are reported by the DAEO toolbox.

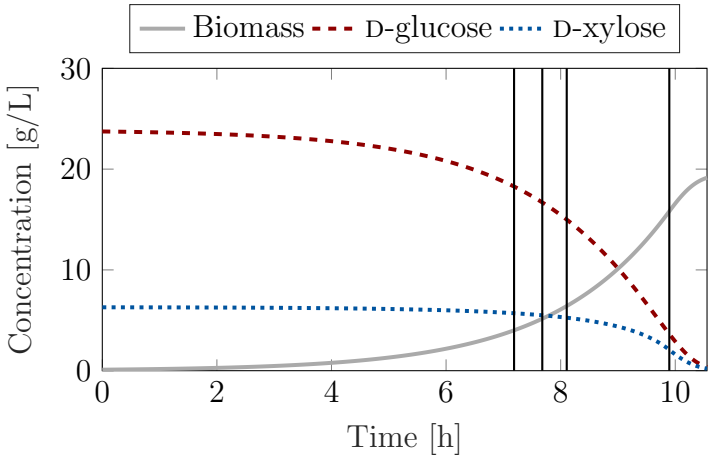


Figure 4: Concentration profiles of extracellular species for aerobic growth of *C. glutamicum*. Vertical lines indicate changes of the active set.

4. Conclusion and outlook

This work deals with the optimization of differential-algebraic equations with embedded optimization criteria. To this end, the embedded optimization problem is first substituted by its first-order optimality conditions. The resulting nonsmooth DAE system is solved by extending the software provided by Hannemann-Tamás et al. [12]. Adjoint sensitivity analysis provides the required gradients to solve the upper-level optimization problem via direct single-shooting. In contrast to existing works, the presented method allows the optimization of the nonsmooth DAE system without approxima-

tion as illustrated by solving three relevant optimization problems based on DAEO models.

The theory of this work covers the use of more general objective function formulations than the ones used in the examples. For example, it allows the consideration of parameter estimation problems, e.g., based on a least-squared objective. In a future work, the framework shall be used to estimate parameters related to the intracellular reaction network or to the uptake kinetics (such as Michaelis-Menten constant) of DFBA models to increase the model agreement when compared to experimental data and thereby also the applicability of this modeling approach. Another direction of research is the global optimization of the embedded optimization problem that is particularly important for embedded NLPs with multiple local optima. In the present work, we used local optimality conditions that do not allow a conclusion about global optimality so we relied on thorough check of the results. Further, the numerical framework may form the basis for more complex dynamic optimization examples such as model predictive control or optimal experimental design.

Availability and requirements

License: The software code for optimization of DAEOs is available under the terms of the GNU General Public License version 3.0 as published by the Free Software Foundation at `https://www.gnu.org/licenses/gpl-3.0.html`.

Link to repository: `http://permalink.avt.rwth-aachen.de/?id=646019`

Operating system(s):

Programming language: The models and the program for optimization of DAEOs are written in C++.

Other requirements: CPLEX for embedded linear and quadratic programs (tested with version 12.8). SNOPT or other NLP solver for upper-level optimization problem (we used SNOPT 7.2-4).

Acknowledgements This work was supported by the German Research Foundation (DFG) under grant MI 1851/3-2. We thank our colleagues Falco Jung for proof reading and Adrian Caspari for his help with the second case study. We also thank Stephan Noack, Eric von Lieres and Prof. Wiechert from IBG-1, FZ Jülich, for many fruitful discussions on the DFBA modeling approach.

Appendix A. Small linear DAEO

Here, we use a small example where the embedded optimization problem is linearly constrained to illustrate the need for the DAEO toolbox. For this case, a switching event, i.e., a change of the active set, requires the optimizer to solve the embedded optimization problem and choose a new set of computationally active constraints. This will be illustrate using a simple DAEO with one differential variable y_d, three algebraic variables $\boldsymbol{y}_a$ and no time-invariant parameters. We consider the time horizon [0,2.5]. The DAEO model is given by

$$\dot{y}_d(t) = 1, \quad \text{with} \quad y_d(t_0) = 0, \tag{A.1a}$$

$$\min_{\hat{y}_{a,1},\hat{y}_{a,2},\hat{y}_{a,3}} \; -(\hat{y}_{a,1} + \hat{y}_{a,2}) \tag{A.1b}$$

$$\text{s.t. } 0 = g_1(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = \hat{y}_{a,1} + \hat{y}_{a,2} + \hat{y}_{a,3} - y_d(t), \tag{A.1c}$$

$$0 \leqslant g_2(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = \hat{y}_{a,1}, \tag{A.1d}$$

$$0 \leqslant g_3(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = \hat{y}_{a,2}, \tag{A.1e}$$

$$0 \leqslant g_4(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = 1 - \hat{y}_{a,1}, \tag{A.1f}$$

$$0 \leqslant g_5(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = 1 - \hat{y}_{a,2}, \tag{A.1g}$$

$$0 \leqslant g_6(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = \hat{y}_{a,3}. \tag{A.1h}$$

Figure A.5 shows the feasible set of the embedded linear program of DAEO (A.1). The variables $\hat{y}_{a,1}$ and $\hat{y}_{a,2}$ are box-constrained by the constraints (A.1d) - (A.1g). An additional inequality constraint $\hat{y}_{a,1} + \hat{y}_{a,2} \leqslant y_d(t)$ is reformulated giving the equality constraint (A.1c) and inequality (A.1h) introducing the slack variable $\hat{y}_{a,3}$. This reformulation yields an embedded

LP that is typical for DFBA models (cf. [26, Eqns. (15)]) where only the right-hand-side of the equality constraints is time-dependent. This constraint depends on the differential variable $y_d(t)$ as indicated by the dashed lines for different time points. The cost vector of the linear part of the objective function $\boldsymbol{q}$ is orthogonal to constraint (A.1c).

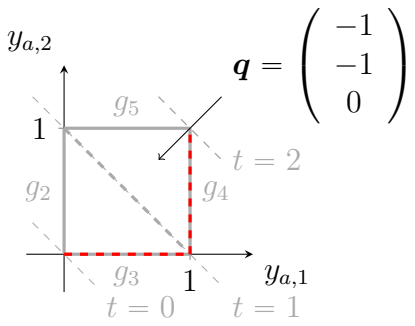


Figure A.5: Feasible set of the embedded LP in DAEO (A.1).

We now apply the KKT embedding solution approach with the Lagrangian function given by

$$\begin{aligned} L(y_d, \boldsymbol{y}_a, \boldsymbol{\lambda}, t) = &-(y_{a,1}(t) + y_{a,2}(t)) \\ &- \lambda_1(t) \cdot (y_{a,1}(t) + y_{a,2}(t) + y_{a,3}(t) - y_d(t)) \\ &- \lambda_2(t) \cdot y_{a,1}(t) - \lambda_3(t) \cdot y_{a,2}(t) - \lambda_4(t) \cdot (1 - y_{a,1}(t)) \\ &- \lambda_5(t) \cdot (1 - y_{a,2}(t)) - \lambda_6(t) \cdot y_{a,3}(t). \end{aligned}$$

Evaluation of the time-dependent KKT conditions yields the following non-

smooth algebraic equation system

$$
\begin{aligned}
0 &= L_{\boldsymbol{y}}(y_d, \boldsymbol{y}_a, \boldsymbol{\lambda}, t) \\
&= \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \lambda_1(t) \\
&\quad - \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_2(t) \\ \lambda_3(t) \\ \lambda_4(t) \\ \lambda_5(t) \\ \lambda_6(t) \end{pmatrix},
\end{aligned} \tag{A.2a}
$$

$$
0 = g_1(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}) = y_{a,1}(t) + y_{a,2}(t) + y_{a,3}(t) - y_d(t), \tag{A.2b}
$$

$$
0 = \begin{cases} g_2(\boldsymbol{y}_d(t), \boldsymbol{y}_a, t, \boldsymbol{p}) = y_{a,1}(t) & \text{if } 2 \in I_a \\ \lambda_2(t) & \text{else} \end{cases}, \tag{A.2c}
$$

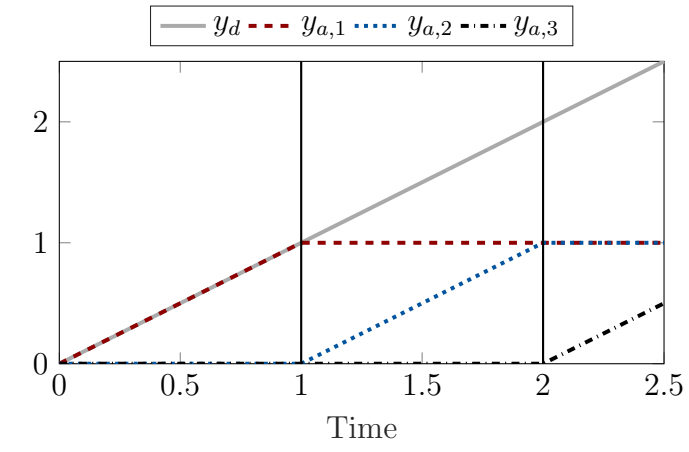
$$
0 = \begin{cases} g_3(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = y_{a,2}(t) & \text{if } 3 \in I_a \\ \lambda_3(t) & \text{else} \end{cases}, \tag{A.2d}
$$

$$
0 = \begin{cases} g_4(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = 1 - y_{a,1}(t) & \text{if } 4 \in I_a \\ \lambda_4(t) & \text{else} \end{cases}, \tag{A.2e}
$$

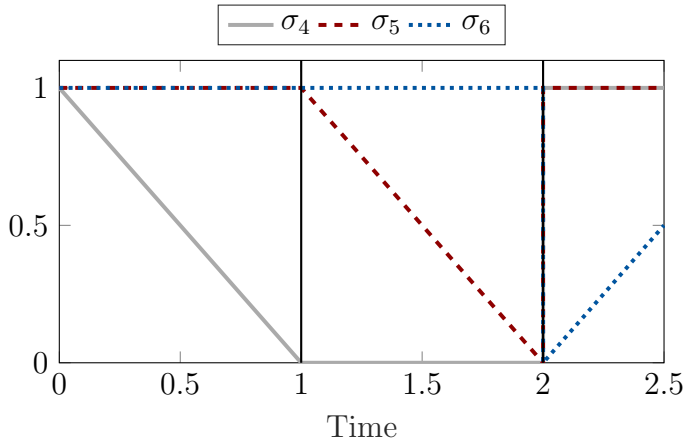
$$
0 = \begin{cases} g_5(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = 1 - y_{a,2}(t) & \text{if } 5 \in I_a \\ \lambda_5(t) & \text{else} \end{cases}, \tag{A.2f}
$$

$$
0 = \begin{cases} g_6(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}) = y_{a,3}(t) & \text{if } 6 \in I_a \\ \lambda_6(t) & \text{else} \end{cases}, \tag{A.2g}
$$

$$
\sigma_j^k = \begin{cases} \lambda_j(t) & \text{if } j \in I_a \\ g_j(\boldsymbol{y}_d(t), \boldsymbol{y}_a, t, \boldsymbol{p}) & \text{else} \end{cases}, \quad \text{for } j = 2, \ldots, 6. \tag{A.2h}
$$



(a)



(b)

Figure A.6: Trajectories of small linear DAEO (A.1): state variables (top); switching functions (bottom).

The solution of DAEO (A.1) using the KKT embedding solution method is shown in Figure A.6. At initial time $t = t_0$, the DAEO toolbox solves the embedded LP and returns the set of computationally active constraints for the first stage $I_a^1 = \{1, 3, 6\}$. The dynamic behavior of the algebraic variables is indicated by the red dashed line in Fig. A.5. The algebraic equation system

takes the following form

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} y_{a,1}(t) \\ y_{a,2}(t) \\ y_{a,3}(t) \\ \lambda_1(t) \\ \lambda_2(t) \\ \lambda_3(t) \\ \lambda_4(t) \\ \lambda_5(t) \\ \lambda_6(t) \end{pmatrix} = \begin{pmatrix} y_d(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{A.3}$$

Note that the order of the equations was changed to better visualize the structure of the linear algebraic equation system. For the chosen set of computationally active constraints, the following observations can be made: First, the Lagrange multiplier can be solved independently of $y_d(t)$ and $\boldsymbol{y}_a(t)$ and will be constant with respect to time. Second, the embedded LP has alternate optima, i.e., no unique solution, and the LP solver will choose among the possible solutions. The solution is degenerate because we have $\lambda_2 = 0$ for $2 \in I_a^1$. Remember that the ϵ-approach prevents the event detection algorithm from triggering an event. Lastly, LICQ holds, the projected Hessian has dimension zero and is therefore regular and the partial Jacobian $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}$ is regular, i.e., we have an index-1-system.

At $t = 1$, we have that $y_{a,1}(t = 1) = 1$ and the zero-crossing of switching function $\sigma_4^1 = (1 - y_{a,1}(t))$ triggers an event, i.e., the first switching time

point t^1 is found and a new stage is created. Only activating $g_4 = 0$ for the second stage makes the algebraic part of the DAE unsolvable because we would have $y_{a,1}(t) = 1$ from g_4, $y_{a,2}(t) = 0$ from g_2, $y_{a,3}(t) = 0$ from g_6, and g_1 would provide no additional information. As a consequence, this combination of active constraints violates LICQ and the DAE cannot be solved. At this point, the DAEO toolbox calls the optimizer to solve the LP again and return a new set of computationally active constraints. Remember that we use the ϵ-approach to ensure that a new set of computationally active constraints is found. In this case, $I_a^2 = \{1, 4, 6\}$. In the second stage, the solution of the algebraic variables $(y_{a,1}, y_{a,2})$ moves from (1,0) along g_4 to $(1, 1)$ as indicated by the red dashed line in Fig. A.5. The algebraic equation system for stage 2 is given by

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} y_{a,1}(t) \\ y_{a,2}(t) \\ y_{a,3}(t) \\ \lambda_1(t) \\ \lambda_2(t) \\ \lambda_3(t) \\ \lambda_4(t) \\ \lambda_5(t) \\ \lambda_6(t) \end{pmatrix} = \begin{pmatrix} y_d(t) \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \quad \text{(A.4)}$$

and similar observations can be made, most importantly, $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}$ is regular and the system can be solved.

The last change of the set of computationally active constraints occurs at

$t = 2$, where zero-crossing of $\sigma_5^2 = g_5$ is found. The new set of computationally active constraints for stage 3 is $I_a^2 = \{1, 4, 5\}$ and simulation continues until t_f.

This example shows that embedded optimization problems that are linearly constrained require the calculation of a new set of computationally active constraints by an optimizer. That is because the activation of an inequality constraint based on the respective switching function crossing zero makes the resulting linear algebraic equation system unsolvable because LICQ is violated. As a consequence, most linearly constrained DAEOs are not solvable by direct implementation of the nonsmooth DAE system.

Another important observation can be made for $y_d(t = 0) = 0$ where the optimal solution is $\boldsymbol{y}_a = (0, 0, 0)^T$. At this point, the set of computationally active constraints defining the algebraic equation system is non-unique. Any two of the constraints (A.1d), (A.1e) and (A.1h) can hold with equality, i.e., be part of the set of computationally active constraints I_a. The same is true for $y_d(t = 1) = 1$ and the corresponding optimal solution is $\boldsymbol{y}_a = (1, 1, 0)^T$. Here, either two of (A.1f), (A.1g) and (A.1h) can be active.

References

[1] Baker, L., Pierce, A., Luks, K., 1982. Gibbs energy analysis of phase equilibria. Society of Petroleum Engineers Journal 22, 731–742. 10.2118/9806-PA.

[2] Baumrucker, B.T., Biegler, L.T., 2009. MPEC strategies for optimization of a class of hybrid dynamic systems. Journal of Process Control 19, 1248–1256. 10.1016/j.jprocont.2009.02.006.

[3] Biegler, L.T., 2010. Nonlinear programming: Concepts, algorithms, and applications to chemical processes. Society for Industrial and Applied Mathematics and Mathematical Optimization Society, Philadelphia.

[4] Cao, Y., Li, S., Petzold, L., Serban, R., 2003. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. SIAM Journal on Scientific Computing 24, 1076–1089. 10.1137/S1064827501380630.

[5] Caspari, A., Lüken, L., Schäfer, P., Vaupel, Y., Mhamdi, A., Biegler, L., Mitsos, A., 2019. Dynamic optimization with complementarity constraints: Regularization for direct shooting. Optimization Online .

[6] Chang, L., Liu, X., Henson, M.A., 2016. Nonlinear model predictive control of fed-batch fermentations using dynamic flux balance models. Journal of Process Control 42, 137–149. 10.1016/j.jprocont.2016.04.012.

[7] Emenike, V.N., Schenkendorf, R., Krewer, U., 2018. Model-based optimization of biopharmaceutical manufacturing in pichia pastoris based

on dynamic flux balance analysis. Computers & Chemical Engineering 118, 1–13. 10.1016/j.compchemeng.2018.07.013.

[8] Gill, P.E., Murray, W., Saunders, M.A., 2005. SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Review 47, 99–131. 10.1137/S0036144504446096.

[9] Gomez, J.A., Höffner, K., Barton, P.I., 2014. DFBAlab: A fast and reliable MATLAB code for dynamic flux balance analysis. BMC bioinformatics 15, 409. 10.1186/s12859-014-0409-8.

[10] Gopal, V., Biegler, L.T., 1999. Smoothing methods for complementarity problems in process engineering. AIChE Journal 45, 1535–1547. 10.1002/aic.690450715.

[11] Hanly, T.J., Henson, M.A., 2011. Dynamic flux balance modeling of microbial co-cultures for efficient batch fermentation of glucose and xylose mixtures. Biotechnology and Bioengineering 108, 376–385. 10.1002/bit.22954.

[12] Hannemann-Tamás, R., Muñoz, D.A., Marquardt, W., 2015. Adjoint sensitivity analysis for nonsmooth differential-algebraic equation systems. SIAM Journal on Scientific Computing 37, A2380–A2402. 10.1137/140976315.

[13] Harwood, S.M., Höffner, K., Barton, P.I., 2016. Efficient solution of ordinary differential equations with a parametric lexicographic linear program embedded. Numerische Mathematik 133, 623–653. 10.1007/s00211-015-0760-3.

[14] Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S., 2005. Sundials: Suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. 31, 363–396. 10.1145/1089014.1089020.

[15] Hjersted, J.L., Henson, M.A., 2006. Optimization of fed-batch *Saccharomyces cerevisiae* fermentation using dynamic flux balance models. Biotechnology progress 22, 1239–1248. 10.1021/bp060059v.

[16] Höffner, K., Harwood, S.M., Barton, P.I., 2013. A reliable simulator for dynamic flux balance analysis. Biotechnology and Bioengineering 110, 792–802. 10.1002/bit.24748.

[17] Kojima, M., 1980. Strongly stable stationary solutions in nonlinear programs, in: Analysis and Computing of Fixed Points. Academic Press, New York, pp. 93–138.

[18] Leppävuori, J.T., Domach, M.M., Biegler, L.T., 2011. Parameter estimation in batch bioreactor simulation using metabolic models: Sequential solution with direct sensitivities. Industrial & Engineering Chemistry Research 50, 12080–12091. 10.1021/ie201020g.

[19] Lin, H., Antsaklis, P.J., 2014. Hybrid Dynamical Systems: An Introduction to Control and Verification. volume 1 of *Foundations and Trends in Systems and Control*. now publishers inc. 10.1561/2600000001.

[20] Mahadevan, R., Edwards, J.S., Doyle, F.J., 2002. Dynamic flux balance analysis of diauxic growth in *Escherichia coli*. Biophysical Journal 83, 1331–1340. 10.1016/S0006-3495(02)73903-9.

[21] Michelsen, M.L., 1982. The isothermal flash problem. Part I. Stability. Fluid Phase Equilibria 9, 1–19. 10.1016/0378-3812(82)85001-2.

[22] Naumann, U., 2012. The art of differentiating computer programs: An introduction to algorithmic differentiation. SIAM, Philadelphia, Pa.

[23] Nocedal, J., Wright, S.J., 2006. Numerical optimization. Second ed., Springer, New York.

[24] Palsson, B.Ø., 2008. Systems biology: Properties of reconstructed networks. Reprinted ed., Cambridge University Press, Cambridge.

[25] Ploch, T., Glass, M., Bremen, A.M., Hannemann-Tamás, R., Mitsos, A., 2019a. Modeling of dynamic systems with a variable number of phases in liquid-liquid equilibria. AIChE Journal 65, 571–581. 10.1002/aic.16447.

[26] Ploch, T., von Lieres, E., Wiechert, W., Mitsos, A., Hannemann-Tamás, R., 2020. Simulation of differential-algebraic equation systems with optimization criteria embedded in modelica. Computers & Chemical Engineering 140, 106920. https://doi.org/10.1016/j.compchemeng.2020.106920.

[27] Ploch, T., von Lieres, E., Wiechert, W., Mitsos, A., Hannemann-Tamás, R., 2020, accepted. Simulation of differential-algebraic equation systems with optimization criteria embedded in modelica. Computers and Chemical Engineering .

[28] Ploch, T., Zhao, X., Hüser, J., von Lieres, E., Hannemann-Tamás, R., Naumann, U., Wiechert, W., Mitsos, A., Noack, S., 2019b. Multiscale

dynamic modeling and simulation of a biorefinery. Biotechnology and Bioengineering 116, 2561–2574. 10.1002/bit.27099.

[29] Raghunathan, A.U., Pérez-Correa, J.R., Agosin, E., Biegler, L.T., 2006. Parameter estimation in metabolic flux balance models for batch fermentation—formulation & solution using differential variational inequalities (DVIs). Annals of Operations Research 148, 251–270. 10.1007/s10479-006-0086-8.

[30] Raghunathan, A.U., Soledad Diaz, M., Biegler, L.T., 2004. An MPEC formulation for dynamic optimization of distillation operations. Computers & Chemical Engineering 28, 2037–2052. 10.1016/j.compchemeng.2004.03.015.

[31] Ritschel, T.K., Capolei, A., Gaspar, J., Jørgensen, J.B., 2018. An algorithm for gradient-based dynamic optimization of UV flash processes. Computers & Chemical Engineering 114, 281–295. 10.1016/j.compchemeng.2017.10.007.

[32] Sahlodin, A.M., Watson, H.A.J., Barton, P.I., 2016. Nonsmooth model for dynamic simulation of phase changes. AIChE Journal 62, 3334–3351. 10.1002/aic.15378.

[33] Scott, F., Wilson, P., Conejeros, R., Vassiliadis, V.S., 2018. Simulation and optimization of dynamic flux balance analysis models using an interior point method reformulation. Computers & Chemical Engineering 119, 152–170. 10.1016/j.compchemeng.2018.08.041.

[34] Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106, 25–57. 10.1007/s10107-004-0559-y.

[35] Zhao, X., Noack, S., Wiechert, W., von Lieres, E., 2017. Dynamic flux balance analysis with nonlinear objective function. Journal of Mathematical Biology 147, 761. 10.1007/s00285-017-1127-4.