Simulation of differential-algebraic equation systems with optimization criteria embedded in Modelica

Tobias Ploch[a], Eric von Lieres[b], Wolfgang Wiechert[b], Alexander Mitsos[a,c,d], Ralf Hannemann-Tamás[a,*]

[a]*Process Systems Engineering (AVT.SVT), RWTH Aachen University, 52074 Aachen, Germany*
[b]*Institute of Bio- und Geosciences, IBG-1: Biotechnology, Forschungszentrum Jülich GmbH, 52425 Jülich, 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 equations with embedded optimization criteria (DAEO) are a class of mathematical models for underdetermined differential-algebraic equation (DAE) systems with less algebraic equations than algebraic variables. The algebraic variables may be calculated as the solution of an embedded (non)linear program, yielding a DAEO system. An example for DAEOs is the dynamic flux balance analysis (DFBA) approach, where the formulation of metabolic reaction networks leads to an underdetermined equation system for the intracellular fluxes that are assumed to behave optimally with respect to some cell-specific optimization criterion.

We present a toolbox that allows formulation of DAEOs in the object-oriented Modelica modeling language. The solution method is based on substituting the embedded optimization problem with its first-order Karush-

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

Kuhn-Tucker conditions to obtain a nonsmooth DAE system that can be simulated by a root-finding DAE solver. One nonlinear example and two examples based on DFBA demonstrate the performance of the toolbox.
Keywords: nonsmooth dynamic systems, Karush-Kuhn-Tucker conditions, dynamic flux balance analysis, Modelica

1. Introduction

In chemical engineering, dynamic models are important for an improved understanding of transient process behavior, in particular in response to disturbances. They also help to design startup and shutdown procedures and form the basis for model-based operation and control. In some cases, however, the system of differential-algebraic equations (DAE) may be underdetermined due to for instance lack of mechanistic knowledge, i.e., there may be more algebraic states than algebraic equations.

The simulation of such underdetermined DAEs is a non-trivial problem. One may interpret these systems as differential inclusions where the right-hand side of the differential equation is a set-valued function [41]. Under some assumptions (e.g., linear differential and algebraic equations), it might be possible to obtain a closed-form description of the set of trajectories. However, the solution of general nonlinear underdetermined DAEs is much more challenging. Methods based on Monte Carlo sampling or methods that approximate the reachable set [45] have to be employed.

Nevertheless, from the modeling perspective it is often favorable to impose one optimization condition which selects a unique or at least locally unique solution from the space of possible solutions. This approach leads to

a differential-algebraic equation system with optimization criteria embedded (DAEO). This problem class is relevant to a variety of chemical engineering applications.

For instance, in systems biology, a successful method to model the complex reactions inside a living organism is based on steady-state balance equations coupled with the stoichiometry of intracellular reactions and additional thermodynamic, kinetic or phenomenological constraints to narrow the possible metabolic flux distributions inside the cell. In most cases, the resulting (linear) equation system is underdetermined. The flux balance analysis (FBA) method selects one possible flux distribution by imposing an optimization criterion subject to these constraints [44, 28]. The dynamic flux balance analysis (DFBA) method couples the FBA approach to the dynamics of the reaction medium [23, 28]. DFBA became one of the dominating methodologies used to describe the transient behavior of large biochemical networks inside living cells, for example, to model the growth of *Escherichia coli* (*E. coli*) on D-glucose and acetate [23] or to investigate growth and ethanol productivity of a co-culture of *E. coli* and *Saccharomyces cerevisiae* on D-glucose/D-xylose mixtures [15]. The DFBA approach leads to a DAEO where the equality and inequality constraints of the embedded optimization problem are linear. The objective function may be linear (e.g., maximization of cell growth [44, 15]) or nonlinear (e.g., maximization of biomass yield per flux unit [47]).

Dynamic models of separation processes often assume thermodynamic equilibrium between coexisting phases (e.g., vapor-liquid equilibrium) that is found at the (global) minimum of Gibbs free energy [2, 25, 26]. Here,

the phase or chemical equilibrium assumption is made on purpose due to much faster mass exchange dynamics compared to the overall dynamics of the system. In contrast, the embedded optimization problem in the DFBA approach is motivated by a lack of knowledge regarding the intracellular flux distribution. Nevertheless, dynamic models based on phase equilibrium may be formulated as DAEOs where both objective function and constraints (e.g., material balance) are nonlinear. The applications range from modeling of organic aerosol particles [21] to dynamic simulation of separation processes based on different types of phase equilibria [13, 3, 37].

There are different approaches to solve DAEOs. The direct approach (DA) is commonly used if an embedded LP is considered [15, 18]. It calculates the algebraic variables by directly solving the optimization problem in each time step of the integration. The direct approach is used in some MATLAB implementations, for example, DyMMM [49] and as part of the COBRA toolbox [38]. While this approach is computationally fast enough for a certain class of DAEOs (in particular embedded LPs), it bears the risk of wrong simulation results caused by the interaction of optimization and integration algorithm. For example, an attempted integration step may lead to infeasibility of the embedded optimization problem [17]. A remedy is to use first-order optimality conditions (KKT conditions) to reformulate the DAEO into a nonsmooth DAE. This idea has been used to solve DAEO applications with NLP embedded [21] and resulted in Fortran [19] and MATLAB [12] implementations for solving models with LPs embedded.

We note that for efficient simulation of DAEOs taylored solution methods are required. These solution methods should exploit the fact that during

numerical integration of the models, a sequence of related linear, quadratic or nonlinear programs has to be solved. Actually, we have parametric optimization problems, with the time as varying parameter. A comprehensive overview about the properties of parametric nonlinear programs is given in [14]. If the embedded optimization problem is linear or quadratic, efficient solution methods could borrow ideas from discrete time linear model predictive control, e.g., from multi-parametric programming [8] or online active set strategies for QP solvers [9]. The latter will find a resemblance in our subsequently presented active-set strategy for solving DAEOs with linear or quadratic programs embedded. However, it is important to stress the difference between DAEOs and model-predictive control. In the former, an embedded optimization problem has to be solved, in the latter we have an *outer* optimization problem. The similarity arises because both are in fact parametric optimization problems.

In comparison to existing literature, this work does not focus on a particular application but aims to treat general DAEOs. Thereby, we build on the solution technique proposed by Landry et al. [21] using the first-order KKT conditions to reformulate general DAEOs into nonsmooth DAEs and provide implementations of the direct approach and the KKT embedding approach. We build on Modelica as an object-oriented modeling language that provides a structured and efficient approach to model systems with differential, algebraic and discrete equations. For solving the embedded optimization problem, we implemented an external toolbox that is interfaced to the Modelica-based simulation environment. Our point of view is that Modelica is better suited to model complex systems than MATLAB or Fortran. The

aim of Modelica is to unify and generalize previous object-oriented modeling languages and become a de facto standard [11]. Cellier and Kofman compare Modelica and MATLAB for the simulation of an electrical circuit and conclude that in MATLAB much more manual preprocessing is required [7]. From our personal experience, setting up a simulation with Fortran libraries is even more tedious.

By providing a DAEO toolbox for Modelica, we get an easy-to-use interface for model extension and embedding of DAEO formulations into large and complex models (e.g., DFBA-based bioreactor coupled with plant-wide biorefinery model [31]).

The structure of this work is as follows. In the next section, we formally introduce the DAEO formulation, describe the applied solution methods and discuss the specialized solution method for DFBA models and the implementation within the Modelica language. Subsequently, we demonstrate the applicability of our method on one nonlinear example and two linearly constrained examples based on DFBA. Finally, we close with concluding remarks. The DAEO software will be made available on the public git repository with the permalink `http://permalink.avt.rwth-aachen.de/?id=252659`.

2. Methods

2.1. DAEO formulation

In DAEOs, the algebraic variables are determined by an embedded nonlinear program (NLP), i.e.,

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

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

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

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

Here, $\boldsymbol{y}_d(t) \in \mathbb{R}^{n_d}$ and $\boldsymbol{y}_a(t) \in \mathbb{R}^{n_a}$ are the differential and algebraic variables, respectively. The differential equations are given by sufficiently smooth functions $\boldsymbol{f} : \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \times [0, t_f] \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_d}$. The vector $\boldsymbol{p} \in \mathbb{R}^{n_p}$ comprises all model parameters, t_f is the final time and $t \in [0, t_f]$ is the independent time variable. The algebraic variables are given as the solution point of an embedded NLP comprising Eqs. (1b)–(1d). The task is to find algebraic variables $\boldsymbol{y}_a$ that satisfy the constraints and further minimize an objective function $h : \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \times [0, t_f] \times \mathbb{R}^{n_p} \to \mathbb{R}$.

Let us consider a simple DAEO example with one differential variable y_d, one algebraic variable y_a and the time-invariant parameter vector $p^T = (p_1, p_2, p_3)^T = (1, 3, 1)^T$. The dynamics on the time horizon $[0, 1]$ are given

by the following model

$$\dot{y}_d(t) = 8\pi \cos(2\pi \cdot t), \qquad y_d(0) = 4, \tag{2a}$$

$$y_a(t) \in \min_{\hat{y}_a \in \mathbb{R}} \; h(t, y_d(t), \hat{y}_a, \boldsymbol{p}) = (-y_d(t) + 2\,\hat{y}_a + p_3)^2 \tag{2b}$$

$$\text{s.t.} \;\; 0 \leqslant g_1(t, y_d(t), \hat{y}_a, \boldsymbol{p}) = \hat{y}_a, \tag{2c}$$

$$0 \leqslant g_2(t, y_d(t), \hat{y}_a, \boldsymbol{p}) = p_2 - \hat{y}_a. \tag{2d}$$

In the following, we discuss solution approaches to solve DAEO (1) using Example (2) for illustration.

2.2. Direct approach

In the direct approach, an NLP solver directly solves the embedded optimization problem every time the integration algorithm needs to evaluate the algebraic part of the DAEO (1). The advantage of this approach is the simple implementation. For this reason, it is often applied, in particular, when considering an embedded LP [18, 49, 15]. In that case, the successive solution of slightly changing embedded optimization problems is usually obtained within a small number of iterations and the computation is often fast enough for dynamic simulation. However, the approach bears some limitations that may lead to wrong simulation results [17]. The failure is induced by the interaction of integrator and optimizer. For instance, an attempted Euler step of the integrator may lead to infeasibility of the embedded optimization problem [17]. An additional drawback is the lack of derivative information that the integrator needs for calculation of the Jacobian. Using the direct approach, this information needs to be obtained using numerical approxima-

tion methods that introduce a truncation error leading to another potential cause of error for the integration of DAEOs. Alternatively, post-optimal sensitivity analysis (see, e.g., [10]) may be applied to obtain derivative information. Given the numerous drawbacks of the direct solution method, its derivative calculation is not further investigated in this work. Instead, we present an approach that allows calculation of truncation-free derivatives by reformulating the DAEO into a nonsmooth DAE.

2.3. Reformulation by KKT embedding

The essential idea to solve this problem class is to substitute NLP (1b)–(1d) in DAEO (1) by its associated KKT conditions, an approach also followed by Landry et al. [21]. To set up the KKT necessary conditions of optimality, they introduce time-dependent Lagrange multipliers $\boldsymbol{\lambda}(t) \in \mathbb{R}^{n_g}$, associated with the constraints $\boldsymbol{g}$. Let

$$I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) := \{1 \leqslant k \leqslant n_g \mid g_k(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) = 0\} \subset \{1, 2, \ldots, n_g\} \tag{3}$$

denote the set of active constraints (shortly active set). Note that the equality constraints (1c) are always active. The Lagrangian function is defined as

$$L(\boldsymbol{y}_d, \boldsymbol{y}_a, \boldsymbol{\lambda}, t, \boldsymbol{p}) = h(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) - \sum_{k=1}^{n_g} \lambda_k \cdot g_k(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}), \tag{4a}$$

to formulate the KKT stationarity condition (see, e.g., [27])

$$0 = \nabla_{\boldsymbol{y}_a} L = \nabla_{\boldsymbol{y}_a} h - \sum_{k=1}^{n_g} \lambda_k \cdot \nabla_{\boldsymbol{y}_a} g_k. \tag{4b}$$

The algebraic variables $\boldsymbol{y}_a(t)$ and the non-zero-valued multipliers $\lambda_k(t)$ with $k \in I_a(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p})$ are determined by the first-order conditions in Eq. (4b) together with the active constraints $g_k(\dots) = 0$, $k \in I_a(\dots)$. Landry et al. [21] transform the DAEO (1) to the following parametric nonsmooth DAE:

$$\dot{\boldsymbol{y}}_d(t) = \boldsymbol{f}(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}), \quad t \in [0, t_f] \tag{5a}$$

$$0 = g_k(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}), \quad t \in [0, t_f], \quad k \in I_a(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}) \tag{5b}$$

$$0 = \lambda_k(t), \qquad t \in [0, t_f], \quad k \notin I_a(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), t, \boldsymbol{p}) \tag{5c}$$

$$0 = \nabla_{\boldsymbol{y}_a} L(\boldsymbol{y}_d(t), \boldsymbol{y}_a(t), \boldsymbol{\lambda}(t), t, \boldsymbol{p}), \quad t \in [0, t_f] \tag{5d}$$

$$\boldsymbol{y}_d(0) = \boldsymbol{y}_{d,0}(\boldsymbol{p}). \tag{5e}$$

The KKT-based transformation method yields an equation system that gives a stationary point of the original embedded optimization problem. For ease of notation, we introduce the algebraic variables $\boldsymbol{z} := (\boldsymbol{y}_a^T, \boldsymbol{\lambda}^T)^T$ with $\boldsymbol{z} \in \mathbb{R}^{n_a + n_g}$. There are two main issues that need to be discussed: First, a conclusion about global optimality can only be made for convex NLPs (5b) - (5d). However, in general, one has to provide measures to either check whether one tracks the global minimum or the local minimum. An example where the latter is desirable is the concept of metastability in thermodynamics describing an energy state of an isolated system. A metastable state is locally but not globally minimal with respect to the energy and satisfies some strong second-order sufficient condition of optimality [43]. Furthermore, the solution ($\boldsymbol{y}_a$, $\boldsymbol{\lambda}$) may be non-unique. If the embedded NLP (1b)–(1d) is convex in $\hat{\boldsymbol{y}}_a$ and has a strictly convex objective function for any $(\boldsymbol{y}_d(t), t, \boldsymbol{p})$ and satisfies a constraint qualification, e.g., the linear independence constraint

qualification (LICQ, cf. [27, Def. 12.4]), along the solution trajectory, then, apart from time points where the active set changes, the set of KKT points contains exactly one element that is the unique globally optimal minimum point. However, if the NLP is not convex, then the global optimal objective function value is still unique but there might be more than one minimum point, i.e., for given $(\boldsymbol{y}_d(t), t, \boldsymbol{p})$ the solution set of the embedded NLP might contain more than one element. The same situation may arise if the embedded NLP is convex but has not strictly convex objective function. While this case is of a rather theoretical nature for NLPs, embedded linear programs are a typical example for this scenario. In both cases, $\boldsymbol{y}_a(t)$ has to be chosen according to a heuristic selection strategy. For example, Höffner et al. [19] consider differential equations with linear programs embedded and propose to use lexicographic optimization, i.e., they formulate additional LPs to choose from the optimal solution set of the original LP.

The KKT embedding approach relates DAEOs directly to nonsmooth systems, since several constraints can be active and the activity of any constraint can change at any time. To this end, as we will show later, it suffices to consider only the slightly restricted class of nonsmooth systems of index-1

semi-explicit differential-algebraic equations of type

$$\dot{\boldsymbol{x}}^k(t) = \boldsymbol{f}^k(\boldsymbol{x}^k(t), \boldsymbol{y}^k(t), \boldsymbol{p}),\ t \in [t^{k-1}, t^k],\ k = 1, 2, \ldots, K, \tag{6a}$$

$$0 = \boldsymbol{g}^k(\boldsymbol{x}^k(t), \boldsymbol{y}^k(t), \boldsymbol{p}),\ t \in [t^{k-1}, t^k],\ k = 1, 2, \ldots, K, \tag{6b}$$

$$\boldsymbol{x}^1(t^0) = \boldsymbol{\psi}^0(\boldsymbol{p}), \tag{6c}$$

$$\boldsymbol{x}^{k+1}(t^k) = \boldsymbol{\psi}^k(\boldsymbol{x}^k(t), \boldsymbol{y}^k(t), \boldsymbol{p}),\ k = 1, 2, \ldots, K-1, \tag{6d}$$

$$0 = \boldsymbol{\sigma}^k(\boldsymbol{x}^k(t), \boldsymbol{y}^k(t), \boldsymbol{p}),\ k = 1, 2, \ldots, K-1. \tag{6e}$$

The notation is borrowed from [16]. Let us consider the nonsmooth system (6) in detail. The initial time is t^0, the final time is t^K and the implicitly computed switching times $t^1, \ldots, t^{K-1}$ divide the total time horizon $[t^0, t^K]$ into the intervals $[t^0, t^1]$, $[t^1, t^2]$, $\ldots$, $[t^{K-1}, t^K]$, also called *stage 1*, *stage 2*, $\ldots$, *stage* K, respectively. We allow the structure and the characteristics of the individual stages to implicitly depend on $\boldsymbol{p}$. Especially the number of stages K may depend implicitly on the parameters $\boldsymbol{p}$. The differential and algebraic states are denoted $\boldsymbol{x}^k$ and $\boldsymbol{y}^k$ and differential and algebraic equations are given by Eqs. (6a) and (6b). Equations (6c) and (6d) provide initial conditions for the differential states at the beginning of each *stage*. The switching time points are computed as zero crossings of the switching function $\boldsymbol{\sigma}^k$ given in Eq. (6e). When such a time point t^k is detected, the system switches to the subsequent *stage* $k+1$ that may comprise different states, equations and switching functions. In the context of DAEOs, a change of the active set of the embedded optimization problem will trigger such a switching event (a general introduction will be given shortly; Equations (11) and (12) illustrate the switching functions for the small nonlinear example (2)).

If in the DAEO (1), the embedded NLP is substituted by its associated KKT conditions, we yield the nonsmooth system (5). We will show that this system can be formulated in terms of the notation of Eq. (6). Assume, we have $K-1$ changes in the active sets, corresponding to the times $t^1 < \ldots, t^{K-1}$. Then the set $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$ is constant with respect to t, say $I_a(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}) \equiv I_a^k$ on the open intervals (t_{k-1}, t_k) for $k = 1, \ldots, K$. In particular, we have K stages, and with the notation of system (6), we have

$$\boldsymbol{x}^k(t) \equiv \boldsymbol{y}_d(t), \quad \boldsymbol{y}^k(t)^T \equiv \boldsymbol{z}^T = (\boldsymbol{y}_a(t)^T, \boldsymbol{\lambda}(t)^T) \quad \text{and} \quad \boldsymbol{f}^k = \boldsymbol{f}, \quad k = 1, \ldots, K.$$

Now let us fix $k \in \{1, \ldots, K-1\}$. Then at $t = t^k$ an inequality constraint, say $g_l(\cdot) \geqslant 0$ for some $l \in \{n_e + 1, \ldots, n_g\}$, either changes from inactive to active or vice versa. In the first case we have the identity $\sigma^k(\boldsymbol{y}_d, \boldsymbol{z}, t, \boldsymbol{p}) \equiv g_l(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p})$ while in the second case we have $\sigma^k(\boldsymbol{y}_d, \boldsymbol{z}, t, \boldsymbol{p}) \equiv \lambda_l$. We may identify the algebraic equations in stage k by

$$g^k(\boldsymbol{y}_d, \boldsymbol{z}, t, \boldsymbol{p}) \equiv \begin{pmatrix} (g_j(\boldsymbol{y}_d, \boldsymbol{y}_a, t, \boldsymbol{p}))_{j \in I_a^k} \\ (\lambda_j)_{j \notin I_a^k} \\ \nabla_{\boldsymbol{y}_a} L(\boldsymbol{y}_d, \boldsymbol{y}_a, \boldsymbol{\lambda}, t, \boldsymbol{p}) \end{pmatrix}, \quad t \in (t^{k-1}, t^k). \tag{7}$$

Further, we have $\boldsymbol{\psi}^0 \equiv \boldsymbol{y}_{d,0}$ and for $k = 1, \ldots, K-1$, $\boldsymbol{\psi}^k$ is the identity function, since the differential variables are continuous in DAEO systems. In conclusion, we have reformulated the DAEO (1) to the class on nonsmooth systems given by Eq. (6).

Let us now reformulate the simple example DAEO (2) into a nonsmooth DAE system of form (6). The Lagrangian of the embedded NLP (2b)–(2d)

is given by

$$L(y_d, y_a, \lambda_1, \lambda_2, \boldsymbol{p}) = (-y_d(t) + 2\, y_a(t) + p_3)^2 - \lambda_1(t) \cdot y_a(t) - \lambda_2(t) \cdot (p_2 - y_a(t))\,.$$

Then, evaluating the Karush-Kuhn-Tucker conditions for each $t \in [0, 1]$, yields the algebraic equation system with complementarity constraints

$$0 = L_y(t, y_d(t), y_a(t), \boldsymbol{p}) = 4 \cdot (-y_d(t) + 2y_a(t) + p_3) - \lambda_1(t) + \lambda_2(t), \tag{8}$$

$$0 = \begin{cases} y_a(t) & \text{if } y_a(t) \leqslant 0 \text{ and } \lambda_1(t) \geqslant 0 \\ \lambda_1(t) & \text{else} \end{cases}, \tag{9}$$

$$0 = \begin{cases} p_2 - y_a(t) & \text{if } p_2 - y_a(t) \leqslant 0 \text{ and } \lambda_2(t) \geqslant 0 \\ \lambda_2(t) & \text{else} \end{cases}. \tag{10}$$

The associated switching functions σ_1 and σ_2 are defined by

$$\sigma_1(y_a(t), \lambda_1(t)) = \begin{cases} \lambda_1(t) & \text{if } y_a(t) \leqslant 0 \text{ and } \lambda_1(t) \geqslant 0 \\ y_a(t) & \text{else} \end{cases}, \tag{11}$$

$$\sigma_2(y_a(t), \lambda_2(t), p_2) = \begin{cases} \lambda_2(t) & \text{if } p_2 - y_a(t) \leqslant 0 \text{ and } \lambda_2(t) \geqslant 0 \\ p_2 - y_a(t) & \text{else} \end{cases}. \tag{12}$$

Together with the differential equation (2a), the algebraic equations (8)–(10) and switching functions (11) and (12) constitute a well-determined nonsmooth system in the variables $y_d, y_a, \lambda_1, \lambda_2$. At switching times t^*, the differential state is continuous $y_d(t^*_+) = y_d(t^*_-)$.

Direct implementation of the nonsmooth DAE formulation is possible in most modeling languages (see Table A.3 for a Modelica representation of

Table 1: Manual direct implementation of nonsmooth DAE formulation of illustrative DAEO example (2) in Modelica

```
model SmallNonlinearExample_nDAE
    import Modelica.Constants.pi;
    constant Integer m=0 "Number of DAEO equalities";
    constant Integer q=2 "Number of DAEO inequalities";
    parameter Real p1=1;
    parameter Real p2=3;
    parameter Real p3=1;
    Real y_d(start=4, fixed=true) "Differential variable";
    Real y_a(start=1) "Algebraic variable";
    Real sigma[q] "DAEO switching function values";
    Real lambda[q] "Lagrangian multiplier";
    Real gineq[q] "Inequalities of embedded NLP";
    Boolean isactive_ineq[q](start={false,false});
equation
    der(y_d) = 8*pi*cos(time*(2*pi));
    gineq[1] = y_a;
    gineq[2] = p2 - y_a;
    4*(-y_d + 2*y_a + p3) - lambda[1] + lambda[2] = 0;
    for i in 1:q loop
        if (pre(isactive_ineq[i])) then
            gineq[i] = 0;
            sigma[i] = lambda[i];
        else
            lambda[i] = 0;
            sigma[i] = gineq[i];
        end if;
    end for;
    when (min(sigma) < 0) then
        isactive_ineq = Solver.changeActiveSet(
            sigma,
            pre(isactive_ineq));
    end when;
end SmallNonlinearExample_nDAE;
```

example (2)). However, this approach has some disadvantages. First, the user needs to put much work into the manual implementation which may work for smaller problems but get tedious and error-prone for larger cases. In addition, the correct solution of the system depends on the robustness of the simulation tool. In particular, there may be multiple KKT points satisfying the algebraic equations and the solution may jump between them during simulation. Another important aspect is the calculation of the correct initial active set. Note that it was specified for the small illustrative example. While this might be possible for small problems, it becomes intractable as the size of the embedded NLP increases. An additional difficulty arises for embedded linear programs in case of degenerated solutions (i.e., the strict complementarity condition does not hold and consequently $\lambda_i = 0$ for some index $i \in I_a$ [27]). In that case, the algebraic equation systems become singular and no unique solution can be computed.

To overcome these drawbacks, we use an external toolbox to solve the KKT-based algebraic system. In the next section, we describe how we use homotopy continuation in the external toolbox to avoid jumps between different KKT points. To determine the (initial) active set I_a, the embedded (non)linear program is directly optimized. In Section 3, the solution technique is specialized for the interesting class of DFBA models (i.e., embedded optimization problems that are linearly constrained with linear objective function).

2.4. Tracking solution points of embedded NLPs by homotopy continuation

By substituting the embedded NLP (1b)–(1d) by its KKT conditions for a fixed active set, we get an algebraic equation system. To ensure that the

local solution points of the embedded NLP (1b)–(1d) do not jump between to consecutive time steps, we track them by numerical continuation. An excellent introduction to numerical continuation methods is [1]. For a fixed active set I_a, the unknowns are denoted $\boldsymbol{z}$. For a fixed time $s_1 \in (t_{k-1}, t_k)$, we have to solve nonlinear system (7) which has the form

$$\mathbf{0} = \boldsymbol{F}(\boldsymbol{z}, \boldsymbol{q}) := g^k(\boldsymbol{y}_d(t), \boldsymbol{z}, t, \boldsymbol{p}), \quad \text{where} \quad (\boldsymbol{y}_d(t), t, \boldsymbol{p}) =: \boldsymbol{q} \in \mathbb{R}^{n_q} \tag{13}$$

with $\boldsymbol{F} : \mathbb{R}^{n_z} \times \mathbb{R}^{n_q} \to \mathbb{R}^{n_z}$, which can be assumed to be sufficiently smooth for our purposes. By varying q in Eq. (13) we see that $\boldsymbol{F}(\boldsymbol{z}, t) = \mathbf{0}$ defines a solution manifold that might have more than one connected components. We have to ensure that during numerical integration we stay on the same connected component as the time t progresses.

Let the DAE integrator take a step of size h from $t = s_1$ to $s_2 = s_1 + h$. We can assume that the active set does not change. For the purpose of illustration we consider a simple DAEO with one differential variable x and one algebraic variable y (that is computed as the solution of an embedded nonlinear program). An implicit Euler scheme that always converges after two Newton-type iterations will serve as the numerical integrator. We assume that we already computed accurate approximation of $x_1^{(2)} \approx x(s_1)$ and $y_1^{(2)} \approx y(s_1)$ at $t = s_1$. The upper subscript in parentheses denotes the iteration index of the Newton-type iterations (remember that we always take two iterations to converge). After the first Newton-type iteration, that implicit Euler scheme computes a not very accurate approximation $x_2^{(1)}$ of $x(s_2)$. The corresponding algebraic variable $y_2^{(1)}$ could be computed by any nonlinear equation solver. However, an arbitrary nonlinear equation solver

cannot ensure that we stay on the same connected component of the solution manifold. Therefore, we will use homotopy continuation which will ensure that we do stay on the same connected component. Let us now discuss this first Newton-iteration in detail.

With the notation of Eq. (13) we have $\boldsymbol{z}_1 = y_1^{(2)}$ and $\boldsymbol{q}_1 = x_1^{(2)}$ which satisfy the equation $\boldsymbol{F}(\boldsymbol{z}_1, \boldsymbol{q}_1) = \boldsymbol{0}$. After the first Newton-iteration we have $\boldsymbol{q}_2 = x_2^{(1)}$ and have to compute $\boldsymbol{z}_2 = y_2^{(1)}$ by solving $\boldsymbol{F}(\boldsymbol{z}, \boldsymbol{q}_1) = \boldsymbol{0}$ for $\boldsymbol{z}$. To stay on the same connected component of the solution manifold we employ the homotopy continuation approach

$$\boldsymbol{F}(\boldsymbol{z}(\lambda), (1-\lambda)\boldsymbol{q}_1 + \lambda\, \boldsymbol{q}_2) = \boldsymbol{0}, \quad \lambda \in [0, 1]. \tag{14}$$

We start with $\lambda = 0$ and use a homotopy continuation solver that will take some intermediate steps in λ to solve Eq. (14) until $\lambda = 1$. This way we ensure that we stay on the same connected component of the solution manifold. The situation is sketched in Figure 1. To compute $y_2^{(2)}$ (corresponding to $x_2^{(2)}$) we proceed analogously.

Our homotopy continuation solver is based on "Program 1. A Simple PC Continuation method" of the book of Allgower and Georg [1]. The original code was written in Fortran 77, but we provide a reimplementation using modern C++ programming techniques. The jacobian of $\boldsymbol{F}$ in (14) is computed by the algorithmic differentiation library dco/c++ [22]. The number and locations of the intermediate points are computed adaptively using a steplength control algorithm. For details we refer to [1, page 266ff.]. The source code of the continuation solver is also published in the project repository.

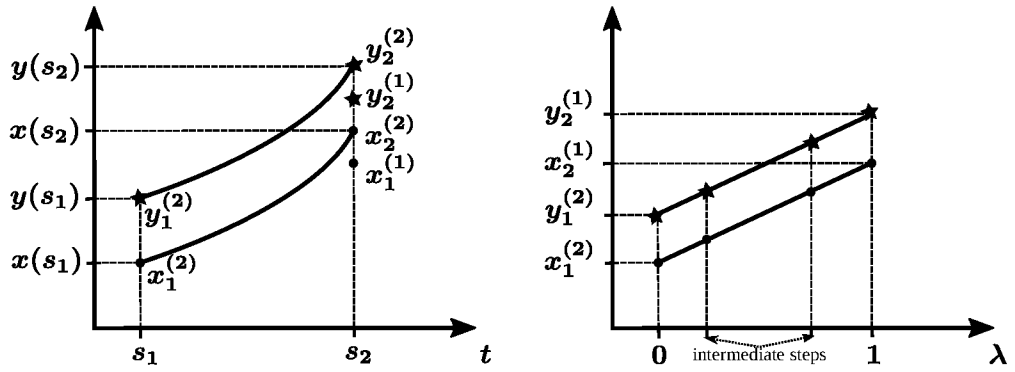

Figure 1: Illustration of the homotopy continuation of Equation (14). On the left graph we see the exact trajectories of $x(t)$ and $y(t)$ for $t \in [s_1, s_2]$. The stars "$\star$" denote the points which have to be solved for in Equation (13). The right graph illustrates the steps taken by a homotopy continuation solver to compute $(y_2^{(1)}, x_2^{(1)})$ starting from $(y_1^{(2)}, x_1^{(2)})$.

2.5. Solvability of nonsmooth DAEs

After discussing non-unique solutions of the embedded optimization problem, we will now focus on the solvability of the nonsmooth DAE system (6). For the discussion, we divide the total time horizon into intervals $[t^{k-1}, t^k]$. On these intervals, the DAE system is smooth because the intervals (or *stages*) are chosen accordingly and the nonsmoothness is captured by switching function (6e). The solvability of smooth DAE systems is investigated by many authors (see, e.g., [4, 33, 20] for an overview), primarily based on differential geometry [32, 35, 36] or derivative array approach [5]. Based on the latter, Kunkel and Mehrmann [20] show that a solution for semi-explicit DAE systems with differential index of 1 exists under mild conditions [20, Theorems 4.13 and 4.18]. In our case, the differential index of problem (6) on the smooth interval $[t^{k-1}, t^k]$ depends on the properties of the solution of the embedded optimization problem. The KKT-based reformulation of embedded NLPs yields an index-1-system if LICQ holds and the second-order

sufficient condition (SOSC, cf. [27, Theorem 12.6]) is fulfilled for the embedded optimization problem [48]. For the KKT conditions of linear-constrained problems (LP and QP embedded), LICQ can be replaced by the condition of linear constraints [27].

Stechlinski et al. [42] extend the theory on solvability to nonsmooth DAEs using projections of the generalized Clarke Jacobian. Another approach for solving nonsmooth DAEs is regularization in order to obtain a smooth DAE. When applying the KKT-based reformulation of the original DAEO system, the nonsmoothness results from the complementarity condition (i.e., $\lambda_k \cdot g_k = 0, k = 1, \ldots, n_g$). Caspari et al. [6] apply regularization to these nonsmooth complementarity constraints to obtain a smooth DAE system and show that the conditions for well-posedness of the original nonsmooth DAE are sufficient of well-posedness of the regularized DAE.

2.6. Solving DAEOs in Modelica

In order to solve this problem class in Modelica-based simulation software, we implemented an external DAEO toolbox that is interfaced to Modelica via the *ExternalObject* class and thereby allows formulation of DAEO (1) in Modelica language. The *constructor* function of the *ExternalObject* parses a textual representation of the embedded optimization problem and constructs an optimization model for the respective solver. Two different approaches are available to solve the embedded optimization problem depending on the chosen solution strategy. The direct approach returns the solution y_a as a function of the time-varying input variables by directly solving the embedded optimization problem. The KKT embedding approach additionally returns the values of switching function $\boldsymbol{\sigma}$. In the first step of this approach, the

toolbox uses an optimizer to solve the embedded problem in order to determine the active set. Subsequently, the KKT conditions (4a) - (4b) are derived and solved during dynamic simulation. An active set change is indicated by zero-crossing of the switching function σ. Returning the values to the integrator allows efficient tracking of these active set changes by its event detection algorithm. At such an event, a new active set is calculated and simulation continues:

```
when min(sigma) < -evtTol then
    updateActiveSet(daeo, daeoIn);
end when;
```

We use a small positive value for evtTol instead of zero to ensure that the optimizer finds a novel active set. This idea is borrowed from the principle of the *discontinuity tolerance* in [**?**]. The value of evtTol is set to 2×10^{-9} unless noted otherwise.

3. Specialized solution method for DFBA models with linear objective

In the context of DFBA models, the constraints of the embedded optimization problem (1c) and (1d) are linear and only a small part of the formulation is time-dependent. In this section, we specialize the KKT embedding solution technique for this modeling class to enable fast and reliable simulation. In particular, we propose a regularization to avoid non-unique solutions and exploit the structure of the model formulation. In case of a linear objective function (i.e., $h = \boldsymbol{c}^T \hat{\boldsymbol{y}}_a$ with objective gradient $\boldsymbol{c}$), the embedded optimization problem of DFBA models comprises a linear program

and any solution vector ($\boldsymbol{y}_a$, $\boldsymbol{\lambda}$) satisfying the KKT conditions is a global minimizer. However, the solution vector is generally non-unique and a selection strategy for choosing one solution is required. As mentioned before, Höffner et al. [19] use a lexicographic optimization approach. They formulate additional objective functions in a priority list. After solving the first LP, the optimal solution value is added as constraint and the next LP is solved. The differential equations in (1) usually depend on a small subset of the solution vector y_a, namely the exchange fluxes. By selecting each exchange flux as objective function in their priority list, they obtain unique exchange fluxes. The choice of order changes the model behavior and is thus an important modeling assumption. As a possible drawback, the lexicographic optimization approach required solving multiple LPs which may be time-consuming. Following a different idea, Scott et al. [19] reformulate the linear program using a logarithmic barrier function to avoid inequality constraints in their formulation. Thereby, they circumvent the non-uniqueness as they obtain a unique solution that is on the interior of the feasible set of the original problem. In this work, we add a quadratic regularization term to the objective function transforming the LP into a quadratic program (QP). The objective function becomes $h = \boldsymbol{c}^T\hat{\boldsymbol{y}}_a + \frac{1}{2}\hat{\boldsymbol{y}}_a^T\boldsymbol{G}\hat{\boldsymbol{y}}_a$. We use a diagonal matrix $\boldsymbol{G}$ with $G_{ii} = \epsilon \; \forall \; i \in \mathbb{R}^{n_a}$ where $\epsilon > 0$ is a small, positive value. With this choice, $\boldsymbol{G}$ is positive definite and the embedded LP is transformed into a strictly convex QP with a unique global solution [27]. After the KKT-based reformulation, the regularization yields continuity of the algebraic variables $y_a(t)$ of the nonsmooth DAE system over time [30, Chapter 1, Theorem 2]. The unique solution vector is obtained by solving a single QP compared to

a potentially high number of LPs that need to be solved sequentially in the lexicographic approach. For a sufficiently small ϵ, the optimal solution of the QP is also a the solution of the original LP [24].

From the set of possible solutions of the LP in the context of DFBA, the reformulated QP selects the solution with minimal overall intracellular flux which is also a popular choice for the objective function [39]. While the method described in section 2.3 also applies for QPs, we exploit the linearity of the constraints using a slightly different solution technique for the DFBA examples to improve computational performance. The embedded optimization problem of the DFBA approach can be written as

$$\min_{\hat{\boldsymbol{y}}_a \in \mathbb{R}^{n_a}} \boldsymbol{c}^T \hat{\boldsymbol{y}}_a + \frac{1}{2} \hat{\boldsymbol{y}}_a^T G \hat{\boldsymbol{y}}_a \tag{15a}$$

$$\text{s.t. } \mathbf{0} = \boldsymbol{A}\hat{\boldsymbol{y}}_a - \boldsymbol{b}(\boldsymbol{y}_d(t), \hat{\boldsymbol{y}}_a, t, \boldsymbol{p}), \tag{15b}$$

$$\mathbf{0} \leqslant \boldsymbol{D}\hat{\boldsymbol{y}}_a - \boldsymbol{d}, \tag{15c}$$

where $\boldsymbol{c} \in \mathbb{R}^{n_a}$ is the objective gradient, $\boldsymbol{G} \in \mathbb{R}^{n_a \times n_a}$ a positive definite matrix representing the nonlinear term of the objective function, the linear equality and inequality constraints are defined by the matrices $\boldsymbol{A} \in \mathbb{R}^{n_e \times n_a}$ and $\boldsymbol{D} \in \mathbb{R}^{n_q \times n_a}$ and vectors $\boldsymbol{b} \in \mathbb{R}^{n_e}$ and $\boldsymbol{d} \in \mathbb{R}^{n_q}$, where $n_q = n_g - n_e$ is the number of inequality constraints (15c). Note that the DFBA approach couples the differential part and the embedded optimization problem exclusively via the time-varying vector $\boldsymbol{b}$ in (15b) while all other matrices and vectors are time-invariant. In the following, we denote this time-dependency by $\boldsymbol{b}(t)$.

We split the vector of Lagrange multipliers $\boldsymbol{\lambda}$ into $\boldsymbol{\mu} \in \mathbb{R}^{n_e}$ and $\boldsymbol{s} \in \mathbb{R}^{n_q}$ referring to the equality constraints (15b) and inequality constraints (15c),

respectively. The Lagrangian function of (15) is then defined as

$$L(\boldsymbol{y}_a, \boldsymbol{\mu}, \boldsymbol{s}) = \boldsymbol{c}^T \boldsymbol{y}_a + \frac{1}{2} \boldsymbol{y}_a^T \boldsymbol{G} \boldsymbol{y}_a - \boldsymbol{\mu}(\boldsymbol{A}\boldsymbol{y}_a - \boldsymbol{b}(t)) - \boldsymbol{s}(\boldsymbol{D}\boldsymbol{y}_a - \boldsymbol{d}), \tag{16}$$

and the KKT stationarity conditions is given by

$$\boldsymbol{0} = \nabla_{\boldsymbol{y}_a} L = \boldsymbol{c} + \boldsymbol{G}\boldsymbol{y}_a - \boldsymbol{A}^T \boldsymbol{\mu} - \boldsymbol{D}^T \boldsymbol{s}. \tag{17}$$

Consequently, the algebraic equations of nonsmooth DAE system (6) in stage k are given by

$$\boldsymbol{g}^k(\boldsymbol{y}_d, (\boldsymbol{y}_a^T, \boldsymbol{\mu}^T, \boldsymbol{s}^T)^T, t, \boldsymbol{p}) = \begin{pmatrix} \boldsymbol{c} + \boldsymbol{G}\boldsymbol{y}_a - \boldsymbol{A}^T \boldsymbol{\mu} - \boldsymbol{D}^T \boldsymbol{s} \\ \boldsymbol{A}\boldsymbol{y}_a - \boldsymbol{b}(t) \\ (D_{j,*}\boldsymbol{y}_a - d_j)_{j \in I_a^k} \\ (s_j)_{j \notin I_a^k} \end{pmatrix}, \quad t \in (t_{k-1}, t_k),$$

where $D_{j,*}$ denotes the jth row of the matrix $\boldsymbol{D}$. The switching function is defined by

$$\boldsymbol{\sigma}^k(\boldsymbol{y}_d, (\boldsymbol{y}_a^T, \boldsymbol{\mu}^T, \boldsymbol{s}^T)^T, t, \boldsymbol{p}) = \begin{pmatrix} (D_{j,*}\boldsymbol{y}_a - d_j)_{j \notin I_a^k} \\ (s_j)_{j \in I_a^k} \end{pmatrix}, \quad t \in (t_{k-1}, t_k).$$

The linear equations $\boldsymbol{g}^k(\cdot) = \boldsymbol{0}$ can be written in matrix notation to examine

the structure of the equation system:

$$\begin{pmatrix} \boldsymbol{A} & & \\ \boldsymbol{D_N} & & \\ \boldsymbol{G} & -\boldsymbol{A}^T & -\boldsymbol{D}^T \\ & & \boldsymbol{S_B} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{y}_a \\ \boldsymbol{\mu} \\ \boldsymbol{s} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}(t) \\ \boldsymbol{d_N} \\ -\boldsymbol{c} \\ \boldsymbol{0} \end{pmatrix} \tag{18}$$

with $\boldsymbol{D_N} \equiv D_{j,*} \ \forall \ j \in I_a$, $\boldsymbol{d_N} \equiv d_j \ \forall \ j \in I_a$ and $\boldsymbol{S_B} \equiv [I_{j,*}]_{j \notin I_a}$, where $I_{j,*}$ is the jth row of identity matrix $\boldsymbol{I} \in \mathbb{R}^{n_q \times n_q}$. Let us first take a step back and consider the original DFBA formulation without quadratic term in the objective function (i.e., $\boldsymbol{G}$ is a zero matrix). In that case, the solution is on the vertex of the feasible set and the number of equality and active inequality constraints equals the number of algebraic variables. Consequently, two observations can be made. First, $\boldsymbol{y}_a$ can be calculated independently of $\boldsymbol{\mu}$ and $\boldsymbol{s}$. Second, only the former is a function of time while $\boldsymbol{\mu}$ and $\boldsymbol{s}$ are constant for a given active set I_a. The solution of the proposed QP is not necessarily on the vertices of the feasible region, i.e., there may be less active inequalities than required for the aforementioned independent calculation of $\boldsymbol{y}_a$. If that is the case, the entire linear equation system (18) needs to be solved. Also, the Lagrange multipliers are no longer constant for a given active set. These observations are exploited within the presented toolbox. These refinements of the KKT embedding approach lead to improved CPU time for solving DFBA models and may be applied to other DAEOs with a similar model structure.

4. Illustrative examples

The following examples demonstrate the modeling of DAEOs and their solution using the nonsmooth system approach described in Section 2.3 and its advantages in terms of required CPU time and the possibility of provide analytic derivatives (using the implicit function theorem) compared to the direct approach (cf. Section 2.2). The first example comprises the numerical results for the small nonlinear DAEO (2). The following two examples are based on the DFBA approach and discuss the different DAEO formulations that is an embedded LP and QP, respectively.

The Modelica-based examples are simulated in Dymola 2020 using Visual Studio 2015 compiler. The implicit integration algorithm DASSL (that is based on the work of Petzold [29, 4]) is used with a tolerance of 1×10^{-12}. The solution method for nonlinear problems uses IPOPT (version 3.12.13) [46] to calculate the initial active set. The linear examples are also simulated using the DAE reformulation of DFBAlab with event tolerance of 2×10^{-9} and MATLAB solver ode15s for integration. Both implementations use CPLEX 12.8 for solving the embedded optimization problem with optimality and feasibility tolerance of 1×10^{-9}. The resulting trajectories are compared by means of the root mean squared error (RMSE) defined by

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (c_{i,\mathrm{DFBAlab}} - c_{i,\mathrm{DAEOtoolbox}})^2}{N}}, \tag{19}$$

where c_i is the concentration of species i corresponding to the differential states of the system and N is the number of time points. 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.

4.1. Nonlinear example

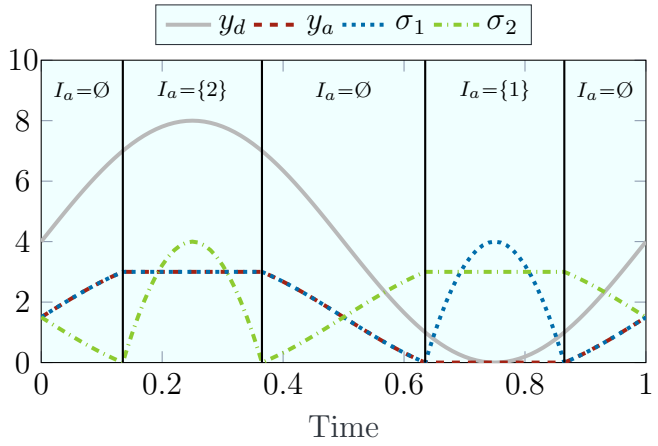

Figure 2: Trajectories of nonlinear DAEO example (2). Vertical lines indicate changes of the active set.

This section shows the simulation results of DAEO (2). The KKT embedding approach is used to solve the nonsmooth DAE reformulation (given by Eqns. (2a),(8)–(12)). The resulting trajectories are shown in Figure 2. Note that identical results are obtained for this example by direct Modelica implementation of the reformulated nonsmooth DAE system (as shown in Table A.3). The simulation starts with an empty active set $I_a = \emptyset$, i.e., $g_i > 0 \ \forall \ i$. During the simulation horizon, four active set changes are detected as indicated by the vertical lines. The first one occurs at $t \approx 0.14$ when σ_2 becomes 0, i.e., $g_2 = 0$ becomes active and the switching function changes from $\sigma_2^- = g_2$ to $\sigma_2^+ = \lambda_2$. At $t \approx 0.37$, the inequality g_2 becomes inactive as indicated by its Lagrange multiplier (remember that $\sigma_2^- = \lambda_2$) becoming 0. The two remaining changes of the active set are caused by inequality g_1

becoming active (at $t \approx 0.63$) and inactive again. Possible difficulties with the nonsmooth system approach are an eventual degeneracy of the embedded NLP and zero-touching of the switching functions which have to be checked.

4.2. DFBA example 1: Corynebacterium glutamicum

The first DFBA example is based on the metabolic network model of *Corynebacterium glutamicum* [31]. It comprises a focused model of the central carbon metabolism of *C. glutamicum* with four different pathways for potential D-xylose utilization. Differential variables are the extracellular concentration of biomass (X), D-glucose (GLC), D-xylose (XYL) and potential (by-)products succinate (SUC), lactate (LAC) and acetate (AC). In total, the model consists of 50 intracellular metabolites, 59 metabolic and 6 exchange fluxes. The uptake rates for carbon sources D-glucose and D-xylose are modeled via Michaelis-Menten kinetics with an upper limit for the overall uptake rate to account for transport limitations. To model aerobic growth conditions, the oxygen uptake rate is not computed by an algebraic equation but as the solution of the embedded optimization problem within fixed upper and lower bounds. Maximization of biomass growth is chosen as optimization criterion. Initial conditions are 0.1 g L^{-1} of biomass, 25 g L^{-1} of D-glucose and 5 g L^{-1} of D-xylose.

The resulting concentration profiles and important exchange fluxes are shown in Figure 3. Under the chosen conditions, only biomass growth without any (by-)product formation is observed. The concentration profiles obtained by using DFBAlab and the DAEO toolbox are nearly identical with small RMSE values of 1.42×10^{-4}, 9.27×10^{-4} and 3.76×10^{-4} for biomass, D-glucose and D-xylose concentration, respectively. Even the switching time

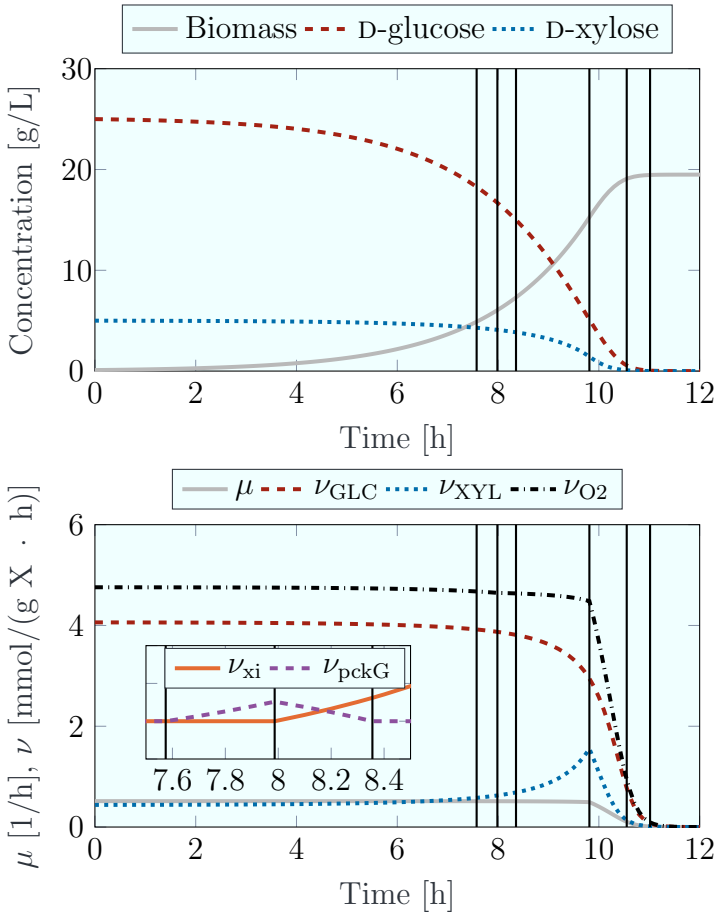

Figure 3: Aerobic growth of *C. glutamicum*: concentration profiles of extracellular species (top); exchange fluxes corresponding to species in the bioreactor (bottom). Magnification shows two intracellular fluxes whose constraints are activated/deactivated at the respective active set change. Vertical lines indicate changes of the active set. The RMSE (compared to DFBAlab) for biomass, D-glucose and D-xylose concentration are 1.42×10^{-4}, 9.27×10^{-4} and 3.76×10^{-4}, respectively.

points t^k are very similar with an RMSE value of 1.6×10^{-3}. Small deviations are probably caused by use of different integration algorithms (DASSL vs. ode15s). For the chosen aerobic conditions (no constraint on oxygen uptake rate) unique extracellular fluxes are obtained. Consequently, identical trajectories with identical switching time points are obtained using DFBAlab,

LP embedded and QP embedded approach.

4.3. DFBA example 2: Escherichia coli

In the third example, we consider the metabolic network model *i*JR904 of *Escherichia coli* K-12 [34]. It consists of 625 unique metabolites, 931 intracellular and 144 exchange fluxes. The dynamic case study is based on the study by Hanly et al. [15] that has been used by various authors to demonstrate their algorithms [19, 12, 40]. The differential variables are the concentrations of biomass, D-glucose, D-xylose and ethanol. Uptake rates are again modeled via Michaelis-Menten kinetics [15, Eqs. (3)-(5)] with parameters from literature [15, Tab. 1]. The uptake kinetics for carbon sources D-glucose and D-xylose contain an inhibition term reflecting growth rate suppression at high ethanol concentrations. The D-xylose uptake rate additionally comprises a second inhibition term with respect to high D-glucose concentrations [15]. The oxygen concentration is kept constant at 0.24 mmol L^{-1}. Initial concentrations are 0.03 g L^{-1} biomass, 15.5 g L^{-1} D-glucose and 8 g L^{-1} D-xylose. Simulation is terminated when carbon sources are depleted. The example is solved using the presented DAEO toolbox, both with a linear and a quadratic objective function. For both cases, maximization of biomass growth is chosen for the linear part of the objective function. The approach with an embedded QP additionally comprises a diagonal regularization matrix as described above. The simulation results of both approaches are compared to DFBAlab serving as benchmark.

The trajectories of the differential states are shown in Figure 4. The concentration profiles show biomass growth on the primary carbon source D-glucose until its depletion at $t \approx 7$ h. At that time point, a larger number

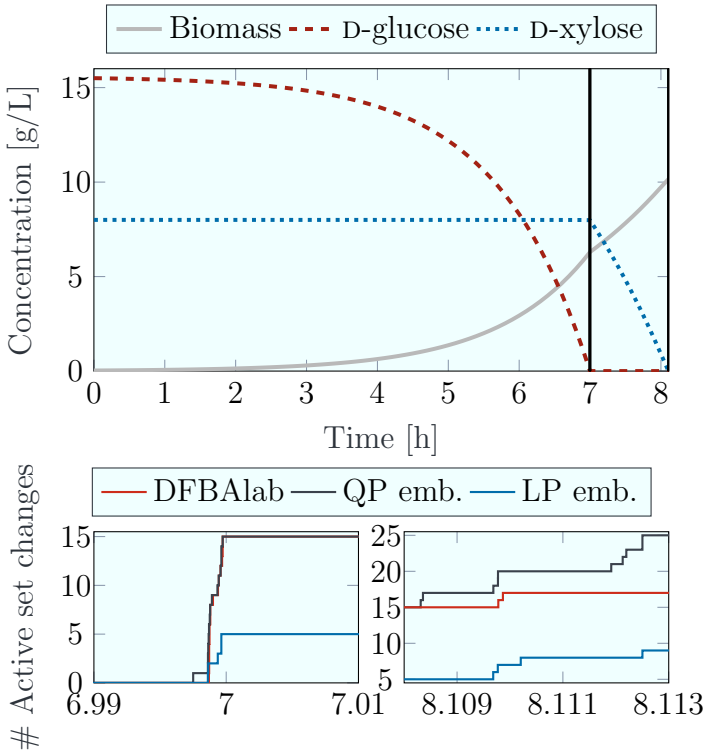

Figure 4: Aerobic growth of *E. coli*: concentration profiles of extracellular species (top) and magnification of two time intervals with large number of active set changes (bottom).

of successive active set changes occurs indicating a switch to growth on D-xylose. After approximately 8.1 h, both carbon sources are depleted and simulation is stopped. In this example, the concentration curves are again in good agreement. All approaches detect active set changes at similar time points. The approaches may identify different active sets. Consequently, a different number of active set changes is detected during simulation (cf. Fig. 4, bottom).

Table 2: Comparison of CPU time for the presented examples using different approaches for solving DAEO (1).

	Mean CPU time (s)[a]	
	Direct	KKT embedding
Nonlinear example [b]	-	0.26
C. glutamicum, LP	0.23	0.02 (0.02)[c]
C. glutamicum, QP	1.36	0.04 (0.03)[c]
E. coli, LP[d]	2.5	0.31 (0.31)[c]
E. coli, QP[d]	8.2	1.13 (1.07)[c]

[a] The mean CPU time refers the average integration time from 10 simulations.
[b] For the nonlinear example, an event tolerance of 1×10^{-6} was used.
[c] The CPU time in brackets is obtained using analytical derivatives.
[d] For the simulations of the *E. coli* example, an event tolerance of 1×10^{-8} was used.

4.4. Discussion

Table 2 summarizes the CPU times for solving the examples presented above using the different approaches. For all examples, the direct approach successfully solves the DAEO. However, the KKT embedding method requires less CPU time. This result can be explained by different reasons. First, the KKT-based reformulation yields a nonsmooth DAE system for which sophisticated solvers exist as opposed to solving the DAEO in the direct approach. In addition, solving an algebraic equation system is generally faster than solving an optimization problem. On the other hand, the KKT-based method yields a system with complementarity constraints that are not

naturally handled by the DAE solver. Returning the values of the switching function to the integrator allows the integration algorithm to explicitly detect changes of the active set (e.g., when an inactive inequality becomes active or vice versa).

For both DFBA examples, the computation time is shorter for the LP embedded approach compared to QP embedded, independent of the chosen solution method (i.e., for both direct and KKT embedding approach). The difference in the computational time is mainly caused by the higher effort required to solve a QP compared to LP.

An additional advantage of the KKT embedding approach is the possibility of providing analytical derivatives of the algebraic reformulation of the embedded optimization problem. The computational times using analytical derivatives are slightly lower compared to numerical derivatives. Overall, it is the fastest approach for the examples shown.

5. Conclusion and outlook

This work presents and compares different solution methods for differential-algebraic equation systems with embedded optimization criterion. The implemented DAEO toolbox allows model formulation in the modeling language Modelica. The direct solution method directly optimizes the embedded (non)linear program in each step and successfully solves the considered examples. The KKT embedding approach reformulates the DAEO into a nonsmooth DAE by using first-order optimality conditions. The nonsmoothness arises if an inequality constraint changes from inactive to active or vice versa. The time-points of such active set changes are determined by the

event detection algorithm as zero-crossings of a suitable switching function. This way, the integrator can handle the nonsmoothness in an effective manner leading to faster simulation times compared to the direct approach. In addition, providing analytical derivatives leads to a further improvement.

The presented solution methods enable simple expansion of DAEO models into large-scale applications (e.g., dynamic modeling of a bioreactor in a plant-wide biorefinery, see [31]). In a further application, the implementation of the KKT embedding method with its analytical derivatives allows for dynamic optimization of DAEOs. The presented solution approach uses local optimization software to determine the initial active set and is further based on local optimality conditions. While this is adequate for LPs and convex QPs, it may lead to undesired local solutions for nonconvex NLPs. In that case, the global solution of the DAEO needs to be adressed in future research.

Availability and requirements

License: The Modelica models and the code of the presented DAEO toolbox are 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=252659`

Operating system(s): In principle any operating system that supports a Modelica-based simulation environment and CPLEX. We successfully tested with Dymola 2018 and 2020 64-bit on Windows and Linux and with OpenModelica 1.14.1 on MacOS and Linux. Using OpenModelica on Windows does not work due to incompatability of the OpenModelica C++ compiler with CPLEX.

Programming language: The models are written in Modelica. If the user wants to extend the DAEO toolbox, C++ is required.

Other requirements: CPLEX for embedded linear and quadratic programs (tested with version 12.8)

Acknowledgements This work was supported by the German Research Foundation (DFG) under grant MI 1851/3-1.

Appendix A. Modelica code of nonlinear DAEO example

Table A.3: Implementation of illustrative DAEO example (2) in Modelica

```
model SmallNonlinearExample_DAEO
    import Modelica.Constants.pi;
    parameter Real p1=1, p2=3, p3=1;
    constant Integer n=2 "number of DAEO variables";
    constant Integer p=3 "number of DAEO inputs";
    constant Integer m=1 "number of DAEO equalities";
    constant Integer q=2 "number of DAEO inequalities";
    constant String params[p]={"x","p2","p3"} "DAEO inputs";
    Real daeoIn[p] "DAEO input values";
    constant String variables[n]={"y","z"} "DAEO variables";
    constant String objective="(-x + 2 * y + p3)^2"
    "DAEO objective function (to be minimized)";
    constant String equalities[m]={"y - z"}
    "DAEO equality constraints";
    constant String inequalities[q]={"y","p2-y"}
    "DAEO inequality constraints";
    parameter Solver.NonlinearDaeoObject daeo=
        Solver.NonlinearDaeoObject(params,
        variables,
        objective,
        equalities,
        inequalities) "Nonlinear external DAEO object";
    parameter Real tolEvent=1e-6;
    Real y_d(start=4, fixed=true) "Differential variable";
    Real y_a(start=1) "Algebraic variable";
    Real sigma[q] "DAEO switching function values";
    Real daeoOut[n] "DAEO solution vector";
    Integer info[2];
    Integer eventCounter(start=0, fixed=true);
equation
    der(y_d) = 8*pi*cos(time*(2*pi));
    daeoIn[1] = y_d;
    daeoIn[2] = p2;
    daeoIn[3] = p3;
    (daeoOut,sigma,info) = Solver.solveNonlinearActiveSet(
        daeo,
        daeoIn,
        n,
        q);
    when min(sigma) < -tolEvent then
        Solver.updateNonlinearActiveSet(daeo, daeoIn);
        eventCounter = pre(eventCounter) + 1;
    end when;
    y_a = daeoOut[1];
end SmallNonlinearExample_DAEO;
```

References

[1] Allgower, E.L., Georg, K., 2003. Introduction to numerical continuation methods. volume 45 of *Classics in applied mathematics*. Society for Industrial and Applied Mathematics, Philadelphia.

[2] 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.

[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] Brenan, K.E., La Campbell, S.V., Petzold, L.R., 1996. Numerical solution of initial-value problems in differential-algebraic equations. volume 14 of *Classics in applied mathematics*. Society for Industrial and Applied Mathematics, Philadelphia.

[5] Campbell, S.L., Griepentrog, E., 1995. Solvability of general differential algebraic equations. SIAM Journal on Scientific Computing 16, 257–270. 10.1137/0916017.

[6] 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 http://www.optimization-online.org/DB_HTML/2019/10/7398.html.

[7] Cellier, F.E., Kofman, E., 2006. Continuous System Simulation. Springer.

[8] Faisca, N.P., Dua, V., Pistikopoulos, E.N., 2011. Multiparametric Linear and Quadratic Programming. John Wiley & Sons, Ltd. chapter 1. pp. 1–23. 10.1002/9783527631216.ch1, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527631216.ch1.

[9] Ferreau, H., Bock, H., Diehl, M., 2008. An online active set strategy to overcome the limitations of explicit mpc. International Journal of Robust and Nonlinear Control 18, 816–830.

[10] Fiacco, A.V., 1983. Introduction to sensitivity and stability analysis in nonlinear programming. volume 165 of *Mathematics in science and engineering*. Academic Press, New York.

[11] Fritzson, P., Bunus, P., 2002. Modelica - a general object-oriented language for continuous and discrete-event system modeling and simulation, in: Proceedings 35th Annual Simulation Symposium, IEEE. pp. 365–380.

[12] 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.

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

[14] Guddat, J., Vazquez, F.G., Jongen, H.T., 1990. Parametric Optimization: Singularities, Pathfollowing and Jumps. Vieweg+Teubner.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Kunkel, P., Mehrmann, V., 2006. Differential-algebraic equations: Analysis and numerical solution. European Mathematical Society, Zürich.

[21] Landry, C., Caboussat, A., Hairer, E., 2009. Solving optimization-constrained differential equations with discontinuity points, with appli-

cation to atmospheric chemistry. SIAM Journal on Scientific Computing 31, 3806–3826. 10.1137/080740611.

[22] Lotz, J., Leppkes, K., Naumann, U., 2012. dco/c++ – Derivative Code by Overloading in C++. Technical Report. RWTH Aachen, Department of Computer Science. ISSN 0935–3232, Aachener Informatik Berichte, AIB-2011-06.

[23] 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.

[24] Mangasarian, O.L., Meyer, R.R., 1979. Nonlinear perturbation of linear programs. SIAM Journal on Control and Optimization 17, 745–752. 10.1137/0317052.

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

[26] Michelsen, M.L., Mollerup, J.M., 2007. Thermodynamic models: Fundamentals & computational aspects. Second ed., Tie-Line Publications, Holte.

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

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

[29] Petzold, L.R., 1982. Description of DASSL: a differential/algebraic system solver. Technical Report. Sandia National Laboratories, Livermore.

[30] Pistikopoulos, E.N., 2007. Multi-parametric programming: Theory, algorithms, and applications. volume 1 of *Process systems engineering*. Wiley-VCH, Weinheim.

[31] Ploch, T., Zhao, X., Hüser, J., von Lieres, E., Hannemann-Tamás, R., Naumann, U., Wiechert, W., Mitsos, A., Noack, S., 2019. Multiscale dynamic modeling and simulation of a biorefinery. Biotechnology and Bioengineering 116, 2561–2574. 10.1002/bit.27099.

[32] Rabier, P.J., Rheinboldt, W.C., 1991. A general existence and uniqueness theory for implicit differential-algebraic equations. Technical Report. Institute for Computational Mathematics and Applications, Pittsburgh.

[33] Rabier, P.J., Rheinboldt, W.C., 2002. Theoretical and numerical analysis of differential-algebraic equations. Handbook of Numerical Analysis 8, 183–540. 10.1016/S1570-8659(02)08004-3.

[34] Reed, J.L., Vo, T.D., Schilling, C.H., Palsson, B.Ø., 2003. An expanded genome-scale model of *Escherichia coli* K-12 (*i*JR904 GSM/GPR). Genome biology 4, R54. 10.1186/gb-2003-4-9-r54.

[35] Reich, S., 1991. On an existence and uniqueness theory for nonlinear differential-algebraic equations. Circuits, Systems, and Signal Processing 10, 343–359. 10.1007/BF01187550.

[36] Rheinboldt, W.C., 1991. On the existence and uniqueness of solutions of nonlinear semi-implicit differential-algebraic equations. Nonlinear Analysis: Theory, Methods & Applications 16, 647–661. 10.1016/0362-546X(91)90172-W.

[37] 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.

[38] Schellenberger, J., Que, R., Fleming, R.M.T., Thiele, I., Orth, J.D., Feist, A.M., Zielinski, D.C., Bordbar, A., Lewis, N.E., Rahmanian, S., Kang, J., Hyduke, D.R., Palsson, B.Ø., 2011. Quantitative prediction of cellular metabolism with constraint-based models: The COBRA toolbox v2.0. Nature protocols 6, 1290–1307. 10.1038/nprot.2011.308.

[39] Schuetz, R., Kuepfer, L., Sauer, U., 2007. Systematic evaluation of objective functions for predicting intracellular fluxes in *Escherichia coli*. Molecular systems biology 3, 119. 10.1038/msb4100162.

[40] 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.

[41] Smirnov, G.V., 2002. Introduction to the theory of differential inclusions. volume 41 of *Graduate studies in mathematics*. American Mathematical Society, Providence.

[42] Stechlinski, P.G., Barton, P.I., 2017. Dependence of solutions of nonsmooth differential-algebraic equations on parameters. Journal of Differential Equations 262, 2254–2285. 10.1016/j.jde.2016.10.041.

[43] Tester, J.W., Modell, M., 1997. Thermodynamics and its applications. 3rd ed., Prentice Hall PTR, Upper Saddle River, N.J.

[44] Varma, A., Palsson, B.Ø., 1994. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type *Escherichia coli* W3110. Applied and environmental microbiology 60, 3724–3731.

[45] Villanueva, M.E., Houska, B., Chachuat, B., 2014. On the stability of set-valued integration for parametric nonlinear ODEs, in: 24th European Symposium on Computer Aided Process Engineering. Elsevier. volume 33 of *Computer Aided Chemical Engineering*, pp. 595–600. 10.1016/B978-0-444-63456-6.50100-9.

[46] 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.

[47] 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.

[48] Zhao, X., Ploch, T., Mitsos, A., Noack, S., Wiechert, W., von Lieres, E., submitted. Analyze the local well-posedness of optimization-constrained differential equations by local optimality conditions .

[49] Zhuang, K., Izallalen, M., Mouser, P., Richter, H., Risso, C., Mahadevan, R., Lovley, D.R., 2011. Genome-scale dynamic modeling of the competition between *Rhodoferax* and *Geobacter* in anoxic subsurface environments. The ISME journal 5, 305–316. 10.1038/ismej.2010.117.