AIChE

Generation of Discrete First- and Second-Order Sensitivities for Single Shooting

Tilman Barz, Robert Kraus, Liming Zhu, Günter Wozny, and Harvey Arellano-Garcia
Chair of Process Dynamics and Operation, Berlin Institute of Technology, Berlin, D-10623, Germany

DOI 10.1002/aic.13720
Published online January 24, 2012 in Wiley Online Library (wileyonlinelibrary.com).

In this work, an approach to the forward generation of discrete first- and second-order sensitivities is proposed. For this purpose, an algorithm has been developed, which can basically be applied to general implicit differential-algebraic equation (DAE) systems. Moreover, the approach has been tailored to both the generation of directional derivatives and sensitivities with respect to discontinuous control trajectories. The implementation of the method is discussed here for the orthogonal collocation method based on Legendre–Gauss–Radau points and considering the linear implicit DAE type, which arises in problems related to chemical engineering. Lastly, the approach has been applied to three case studies of different complexities. The corresponding performance for the generation of Jacobian and Hessian information is discussed in detail. © 2012 American Institute of Chemical Engineers *AIChE J,* 58: 3110–3122, 2012
Keywords: dynamic optimization, quasi-sequential approach, orthogonal collocation, forward generation of first- and second- order sensitivities

Introduction

The generation of first- and second-order sensitivity information is in particular interesting for the solution of dynamic optimization problems when applying the so-called sequential or control parametrization approach, where the solution of the optimization problem is decoupled from the solution of the embedded dynamic system. Here, the dynamic model equations are solved using an integration method of choice, and the results at discrete points in time are then passed as arguments to an optimization routine, where only independent variables are considered.[1] In doing so, the original continuous time problem is transformed into a discrete nonlinear programming problem (NLP). For the efficient solution of this NLP problem, the gradient information is provided based on sensitivities of the dynamic system solution. Generally, the so-called forward and the adjoint mode are considered for the generation of exact sensitivity information.

The forward mode represents merely the derivation of the dynamic equation system with respect to (w.r.t.) independent parameters. For the computation of first- and second-order sensitivities, Ref. 2 applied this approach to general implicit differential-algebraic equation (DAE) systems, which yields an augmentation of the original equation system by a linear matrix differential equation system. The augmented dynamic system is then solved using a standard integrator and yields time evolution of Jacobian and Hessian of the integrated state variables. The forward mode implies the solution of a number of systems scaling linearly and quadratically with the number of parameters, when generating Jacobians and Hessians.

Correspondence concerning this article should be addressed to T. Barz at tilman.barz@tu-berlin.de.

Adjoint sensitivity analysis is especially designed to find the sensitivity of one or a small number of scalar functions for the solution of the dynamic system w.r.t. to a large number of parameters.[3] Usually, when one result of the integration method is needed (e.g., sensitivity of a function value defined only at a specific time point), the adjoint mode is the method of choice for the generation of first- and second-order sensitivities. It is based basically on the calculation of the intermediate quantity, called the adjoint variable, as the solution of a linear system called adjoint system.[3] In the adjoint method, the time for a Jacobian evaluation is a small multiple of the forward increase and nearly independent of parameter number. Moreover, it increases linearly for the Hessian generation. Using a state-of-the-art NLP solver, the second-order derivative information is needed as Hessian of the Lagrangian, which results from the superposition or linear combination of function values and its Hessian for different points in time. Ref. 4 introduced recently the concept of composite adjoints for path-constrained optimal control problems, where the superposition of so-called single adjoints is performed by solving one adjoint system only.

In contrast to the use of full Hessian information for the solution of an NLP problem, directional derivatives, also called restricted or projected second-order derivatives, have been considered in several works (e.g., to be used by a truncated Newton method to calculate search directions). In the forward mode, these Hessian matrix-vector products show the same computational complexity as for the evaluation of first-order sensitivities.[5] Thus, it increases only linearly with the number of free parameters. In the same way, for the reverse generation of second-order directional derivatives, the computational effort is reduced significantly to a nearly constant average cost,[4] or it has a weak dependence on the number of parameters.[6] Examples in chemical engineering with a relative large numbers of free optimization variables

are given, for example, in Refs. 4 and 6–8. All of them report fast and robust solutions using second-order directional derivatives together with tailored optimization methods.

Moreover, to improve the efficiency of the sensitivity generation, different approaches have been proposed. One is the exploitation of the structure of the original dynamic system augmented with equations for sensitivity generation while using a state-of-the-art integration method, see for example Refs. 7, 9, and 10. However, in contrast to the so-called continuous sensitivities, another well-studied approach yields discrete sensitivities. This is the so-called staggered integration method,[11] which follows the same idea as the internal numeric differentiation.[12] It is based on the differentiation of the numeric solution of the original system, which results from the application of a specific integration method. Here, after solving the original problem, all adaptive elements of the integrator are frozen, and thus, the same step lengths are used for the computation of both the original and the sensitivity equation system. In doing so, the differentiation of large and complex integration code with iterations and error control can be avoided. The fact is exploited that the sensitivity equation system is linear and shares the same Jacobian matrices with the original system.[3] Using implicit integration methods for the integration of large-scale systems, the reuse of the factorized Jacobian for the sensitivity equations becomes the most interesting part, because this factorization corresponds to the main computational cost.[13–15]

Most of the solvers with capability of providing sensitivity information are based on the backward differentiation formula (BDF) method. In particular, for the DASSL code, different variants of forward generation of continuous and discrete first-order sensitivities have been studied extensively.[9,11,16,17] Different methods are implemented in DASPK 3.0 package for the solution of general DAE system with index up to two.[16] An extension to a parallel computation of sensitivities for each independent parameter value,[16,18] as well as results using matrix free or inexact Newton method for the solution of all linear problems have also been reported.[19] The generation of first- and second-order directional derivatives using adjoint derivative analysis and the BDF method (here an extension of DASPK) has been introduced in Refs. 6 and 20. A different implementation for index one linear implicit systems have been reported in Refs. 13 and 21. A comparison of the adjoint mode for explicit and implicit integration methods can be found in Ref. 14. An application based on the explicit Runge–Kutta method for the solution of linear implicit systems with a constant mass matrix has been presented in Ref. 22.

Generally, chemical engineering problems are inherently stiff and nonlinear and show often discontinuities mainly imposed by switching points in the control trajectory. In the presence of discontinuities, frequent restarts of multistep integration algorithms represent additional work, because they have to revert to first-order at each restart. In contrast, one-step methods have the capability of self-starting in high orders, which is here of particular interest.[10,15] Discrete second-order adjoint schemes for one-step methods of implicit Runge–Kutta and Rosenbrock type have been presented in Ref. 23. Additional examples of one-step integration methods are applications of a semiimplicit Runge–Kutta method,[15] where discrete first-order sensitivities are generated by a staggered approach for explicit index one ordinary differential equation systems and for one-step extrapolation of the linearly implicit Euler method, where first-order derivatives are generated by a simultaneous state and sensitivity integration approach to linearly implicit DAE systems of index one.[10] Furthermore, a different implementation based on orthogonal collocation for the staggered forward integration of discrete first-order sensitivities of index one DAEs has also been proposed.[24]

In this work, orthogonal collocation, also known as a pseudospectral method, is used for the approximate solution of DAEs. In chemical engineering, collocation methods have been widely used in the simultaneous optimization approach for state and control parametrization,[25,26] or in the sequential optimization approach for the solution of the underlying state integration problem.[27–29] We refer here to collocation methods based on Legendre–Gauss–Radau (LGR) points, a quadrature formula, where one node of the basic mesh is included in the set of collocation points. For problems that are sufficiently smooth, global LGR schemes can be applied, where a single interval is used together with high-order polynomials.[30] In contrast, in a local collocation the parametrization is done by an element-wise discretization in time with each time element using a low-order LGR scheme.[31] Using this so-called Orthogonal Collocation on finite elements (OCFE) method in sequential optimization, an adaptive selection of the number of elements (step-size control) is usually applied.[32] The state-of-the art implementation using LGR points is the RADAU5 software for the solution of linear implicit DAEs.[33] A discussion on their equivalence to particular implicit Runge–Kutta methods with highest-order accuracy and excellent stability properties for index one and higher systems can be found, for example in Ref. 27.

This contribution describes an approach to the forward generation of discrete first- and second-order sensitivities. Special focus lies on the derivation of efficient formulas for sensitivity information w.r.t. discontinuous control trajectories, which are usually applied in single shooting (see section "Problem Formulation"). For this purpose, an algorithm is derived for the generation of first- and second-order discrete sensitivities for one-step integration methods considering general implicit DAE systems (section "Forward Generation of Discrete First- and Second-Order Sensitivities"). The algorithm is then adapted to the generation of directional derivatives. In the following, implementation details are discussed for orthogonal collocation method based on LGR points together with a staggered integration approach while considering the simpler linear implicit DAE type, which arises in chemical engineering problems (section "Algorithmic Implementation"). Finally, the algorithm is applied to three different case studies. Performance estimates for the generation of first- and second-order sensitivities in form of full Jacobian and Hessian information are presented in detail (section "Performance Evaluation").

Problem Formulation

A fully implicit DAE system is considered

$$0 = g(\dot{x}(t), x(t), u(t), p, t) \;, \quad x(t_0) = x_0 \tag{1}$$

where $x \in \mathbb{R}^{Nx}$ denotes the dependent states, $u \in \mathbb{R}^{Nu}$ the independent controls or decisions, $p \in \mathbb{R}^{Np}$ the independent parameters, and t the time. However, whereas Eq. 1 represents the general implicit DAE type, most of the arising dynamic models in chemical engineering can also be expressed by the

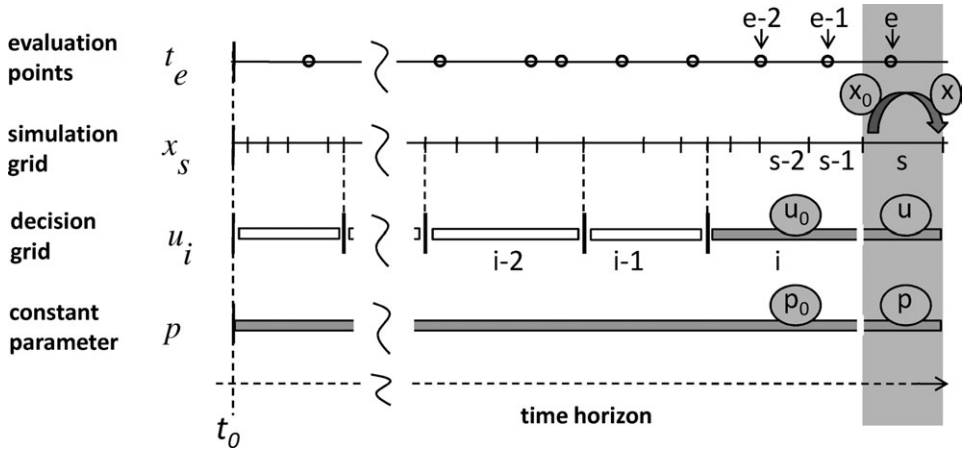

Figure 1. Discretization grids and evaluation points used for the numeric solution of a dynamic optimization problem.

The discrete variables (marked by a circle) illustrate discrete state and decomposed decisions and parameters, which are used in the step-wise integration of Eqs. 1 and 2.

type shown in Eq. 2.

$$\frac{dg^{\text{lhs}}(x(t),p)}{dt} = g^{\text{rhs}}(x(t),u(t),p,t)$$
$$0 = h(x(t),p,t)\ , \quad x(t_0) = x_0 \quad (2)$$

Equation 2 corresponds to a linear implicit DAE system, which is defined in Eq. 3 as follows

$$M(x(t),p)\cdot\dot{x}(t) = \hat{g}(x(t),u(t),p,t)\ , \quad x(t_0) = x_0 \quad (3)$$

where $M(\cdot) = \partial g^{\text{lhs}}(\cdot)/\partial x$ represents the so-called mass matrix, where zero rows have to be added in the presence of algebraic model equations $h(x(t), p, t)$. As the generation of this matrix for some cases may not be trivial (see, e.g., the process models in case study distillation of fatty Acids (DiCol) and high-performance liquid Chromatography (HPLC) in section "Performance evaluation"), the form in Eq. 2 is preferable. In this work, both the general implicit DAE (Eq. 1) and the linear implicit DAE type (Eq. 2) are considered, showing how the sensitivity generation can be simplified for the latter case.

For a complete specification of a simulation problem, the initial state values x_0, the limits t_0, t_f of the simulation time horizon, the model parameter values p, and the time dependent decisions $u(t)$ have to be specified (see Figure 1). In doing so, $u(t)$ is usually transformed using a finite element discretization (called here decision grid) with zeroth- or higher-order approximations within each element. In this work, a zeroth-order approximation is considered, which leads then to piece-wise constant decisions over time. It should be though noted that in addition to the discrete values, the time length of each interval can also be considered in the discrete decisions u_i with $i = 1, \cdots$, Nd. Finally, evaluation points are defined by discrete points in time, at which the state variables $x(t = t_e)$ with $e = 1, \cdots$, Ne are evaluated. Applying a step-wise integration for the solution of the defined problem, the so-called simulation grid with discrete states x_s with $s = 1, \cdots$, Ns is formed, for which the element length results from the adaptive step-size control and the discontinuities in the decision grid.

Local and global sensitivities for single shooting

It can be seen from Figure 1 that in contrast to the parameters p, which are active over the entire simulation horizon, the influence of the discrete decisions u_i on the solution of the integration problem is restricted to a finite element length within the simulation time horizon. In comparison to multiple shooting and the simultaneous optimization approach, dependencies of the state variables on all discrete decisions (in past and current decision intervals) have to be considered while using single shooting. Thus, we distinguish between local and global sensitivities when referring to dependencies of states on the discrete decisions u_j, u_k in a certain decision interval x_i. An illustrative example using the single shooting method for the generation of optimal control profiles and with local and global sensitivities can be found, for example in Ref. 34.

Equation 4 shows the general definition of first- and second-order sensitivities for the problems defined in Eqs. 1 and 2.

$$S'_{i,j} = \frac{\partial x_i}{\partial u_j} \in \mathbb{R}^{\text{Nx}\times\text{Nu}}\ ; \quad S''_{i,j,k} = \frac{\partial^2 x_i}{\partial u_j \partial u_k} \in \mathbb{R}^{\text{Nx}\cdot\text{Nu}\times\text{Nu}}$$
$$\text{with} \quad i,j,k = 1,\cdots,\text{Nd} \quad (4)$$

The term "local sensitivities" (SL) is used to denote dependencies of state variables x_i on decisions u_j, u_k, which are located in the same decision interval, with $i = j = k$ (see Eq. 4). These decisions act direct locally on the solution of x_i. In contrast, all prior decisions u_j, u_k with $j, k = 1, \cdots, i-1$ do not have a direct influence on x_i. Accordingly, these sensitivities are denoted here as "global sensitivities" (SG), with $i > j$ and $j = k$. Finally, we define "mixed sensitivities" (SM) of second order, in which one can further distinguish between "mixed (local–global) sensitivities" with $i = j$ and $j > k$ and "mixed (global) sensitivities" with $i > j$, $i > k$, and $j \neq k$. It has to be noted that dependencies of state variables x_i on decisions u_j, u_k in future decision intervals are zero, with $i < j$ and $i < k$. Moreover, for mixed second-order sensitivities, it follows that $\partial^2 x_i/\partial u_j \partial u_k = \partial^2 x_i/\partial u_k \partial u_j$. These considerations

are used to reduce the computational effort while evaluating sensitivity information (see section "Algorithmic Implementation").*

The complete set of first-order sensitivities, which have to be evaluated in all decision intervals from $S'_{i=1} \in \mathbb{R}^{Nx \times Nu}$ to $S'_{i=Nd} \in \mathbb{R}^{Nx \times Nu \cdot Nd}$, are given in Eq. 5

$$\begin{aligned} S'_{i=1} &= [\mathrm{SL}'_{1,1}] \\ S'_{i=2} &= [\mathrm{SG}'_{2,1},\ \mathrm{SL}'_{2,2}] \\ S'_{i=3} &= [\mathrm{SG}'_{3,1},\ \mathrm{SG}'_{3,2},\ \mathrm{SL}'_{3,3}] \\ &\vdots \\ S'_{i=Nd} &= [\mathrm{SG}'_{Nd,1},\ \mathrm{SG}'_{Nd,2},\ \mathrm{SG}'_{Nd,3},\ \cdots,\ \mathrm{SL}'_{Nd,Nd}] \end{aligned} \tag{5}$$

In the same way, the complete set of second-order sensitivities from $S''_{i=1} \in \mathbb{R}^{Nx \cdot Nu \times Nu}$ to $S''_{i=Nd} \in \mathbb{R}^{Nx \cdot Nu \cdot Nd \times Nu \cdot Nd}$ is given in Eq. 6.

$$\begin{aligned} S''_{i=1} &= \left[\mathrm{SL}''_{1,1,1}\right] \\ S''_{i=2} &= \begin{bmatrix} \mathrm{SG}''_{2,1,1} & \mathrm{SM}''_{2,1,2} \\ \mathrm{SM}''_{2,2,1} & \mathrm{SL}''_{2,2,2} \end{bmatrix} \\ S''_{i=3} &= \begin{bmatrix} \mathrm{SG}''_{3,1,1} & \mathrm{SM}''_{3,1,2} & \mathrm{SM}''_{3,1,3} \\ \mathrm{SM}''_{3,2,1} & \mathrm{SG}''_{3,2,2} & \mathrm{SM}''_{3,2,3} \\ \mathrm{SM}''_{3,3,1} & \mathrm{SM}''_{3,3,2} & \mathrm{SL}''_{3,3,3} \end{bmatrix} \\ &\vdots \\ S''_{i=Nd} &= \begin{bmatrix} \mathrm{SG}''_{Nd,1,1} & \mathrm{SM}''_{Nd,1,2} & \mathrm{SM}''_{Nd,1,3} & \cdots & \mathrm{SM}''_{Nd,1,Nd} \\ \mathrm{SM}''_{Nd,2,1} & \mathrm{SG}''_{Nd,2,2} & \mathrm{SM}''_{Nd,2,3} & \cdots & \mathrm{SM}''_{Nd,2,Nd} \\ \mathrm{SM}''_{Nd,3,1} & \mathrm{SM}''_{Nd,3,2} & \mathrm{SG}''_{Nd,3,3} & \cdots & \mathrm{SM}''_{Nd,3,Nd} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{SM}''_{Nd,Nd,1} & \mathrm{SM}''_{Nd,Nd,2} & \mathrm{SM}''_{Nd,Nd,3} & \cdots & \mathrm{SL}''_{Nd,Nd,Nd} \end{bmatrix} \end{aligned} \tag{6}$$

In a previous work, it has already been demonstrated that the generation of global sensitivities is basically done in a similar way as the generation of local sensitivities.[34] Thus, in the following, we first focus on the generation of local sensitivities for the general DAE system described in Eq. 1 and discuss afterward adaptation and simplifications so as to compute global and mixed sensitivities and to apply the derived formulas to the DAE type defined in Eq. 2.

Forward Generation of Discrete First- and Second-Order Sensitivities

In the general one-step integration method, the approximate solution x_s is obtained by an integration method with the known initial value x_{s-1} (solution of the previous step), the local discrete decisions u_d, and the parameter values p for each integration step s (see Figure 1). The repeated computation for the last and current step is

$$\begin{aligned} x_s &\approx x_{s-1} + \int_{t_{s-1}}^{t_s} f(x, u_d, p, t); \\ x_{s-1} &\approx x_{s-2} + \int_{t_{s-2}}^{t_{s-1}} f(x, u_d, p, t); \quad \cdots \end{aligned} \tag{7}$$

with $\Delta t_s = t_s - t_{s-1}$ being the current step size. Because of the recursive implementation in Eq. 7, the following discussion is only restricted to the current step s. Using the so-called staggered state and sensitivity integration method, all adaptive elements from the integration are "frozen" after each successful step. The sensitivities are then generated based on the available "frozen" information. Thus, to account for the step-wise nature of this procedure, decisions and the parameters are decomposed into piece-wise constant contributions in $u_{d,s}$, $u_{d,s-1}$, and p_s, p_{s-1}, respectively. In doing so, formally independent variables are introduced, which have actually the same value.

$$u_{d,s},\ u_{d,s-1} = u; \qquad p_s,\ p_{s-1} = p \tag{8}$$

Moreover, while decomposed decisions $u_{d,s}$ and parameter p_s contribute only to the current element, $u_{d,s-1}$ and p_{s-1} denote past contributions of the last element.

For ease of discussion, in the remainder of this section, we focus only on sensitivities w.r.t. to decision variables. Furthermore, a simplified notation for the discrete variables x, the decomposed parameters p, and the decisions u within the current interval in defined in Eq. 9.

$$\begin{array}{llll} \mathrm{x}_0 := x_{s-1}; & \mathrm{u}_0 := u_{d,s-1}; & \mathrm{p}_0 := p_{s-1}; & \Delta\mathrm{t}_0 := \Delta t_{d,s-1} \\ \mathrm{x} := x_s; & \mathrm{u} := u_{d,s}; & \mathrm{p} := p_s; & \Delta\mathrm{t} := \Delta t_{d,s} \end{array} \tag{9}$$

Generation of local sensitivities

Throughout this section, the problem of generating local sensitivities $\mathrm{SL}'_{i,i}$, $\mathrm{SL}''_{i,i,i}$ w.r.t. u is first considered along with the solution of the state profile.† The consideration of the individual contributions u_0 and u in Eq. 9 corresponds to the step-wise integration of the model equations. Thus, their influence on the solution within the current simulation subinterval is considered separately. For the implicit one-step integration method, x is calculated in the current element by the solution of an implicit equation system‡

$$0_{Nx} = \mathrm{g}(\mathrm{x}, \mathrm{x}_0, \mathrm{u}, \mathrm{p}, t, \Delta\mathrm{t}) \tag{10}$$

Considering the dependencies on the decomposed contributions u_0 and u (see Eq. 9), we can write§

$$0_{Nx} = \mathrm{g}(\ \mathrm{x}_0(\mathrm{u}_0),\ \mathrm{x}(\mathrm{x}_0(\mathrm{u}_0),\ \mathrm{u}),\ \mathrm{u}\ ) \tag{11}$$

Adopting the rule of differential calculus for functions of several variables, that is, the independent contributions u_0, u, we obtain

$$d^{\{n\}}\mathrm{x} = \left(\frac{\partial}{\partial \mathrm{u}_0} d\mathrm{u}_0 + \frac{\partial}{\partial \mathrm{u}} d\mathrm{u}\right)^{\{n\}} \mathrm{x} \tag{12}$$

*From the above discussion it follows also, that dependencies of states on the parameters always follow the definition for local sensitivities.

†For simplicity the indices i in this section are dropped.

‡Without loss of generality, we assume that the discrete system has the dimension Nx, which is the case, for example, when using first-order discretization.

§Nonitalic symbols are used to indicate discretized variables/quantities (see also Figure 1).

The general expression for the sensitivities of the discrete states in the current element w.r.t. both contributions of the decomposed decisions (in the last and current element) with $du = d\mathrm{u}_0 = d\mathrm{u}$ (see Eq. 8) is defined as follows

$$\mathrm{SL}^{\{n\}} = \frac{d^{\{n\}}\mathrm{x}}{du^{\{n\}}} = \left(\frac{\partial}{\partial \mathrm{u}_0} + \frac{\partial}{\partial \mathrm{u}}\right)^{\{n\}} \mathrm{x} \tag{13}$$

The derivation of the discretized equation system (Eq. 11) w.r.t. the independent contributions u_0, u and their addition results in

$$\frac{\partial \mathrm{g}}{\partial \mathrm{x}} \underbrace{\left(\frac{\partial \mathrm{x}}{\partial \mathrm{x}_0}\frac{\partial \mathrm{x}_0}{\partial \mathrm{u}_0} + \frac{\partial \mathrm{x}}{\partial \mathrm{u}}\right)}_{\mathrm{SL}' = d\mathrm{x}/du} + \frac{\partial \mathrm{g}}{\partial \mathrm{x}_0} \underbrace{\frac{\partial \mathrm{x}_0}{\partial \mathrm{u}_0}}_{\mathrm{SL}'_0 = d\mathrm{x}_0/d\mathrm{u}_0} + \frac{\partial \mathrm{g}}{\partial \mathrm{u}} = 0_{\mathrm{Nx}\times\mathrm{Nu}} \tag{14}$$

From the comparison with Eq. 13, the term SL' in Eq. 14 corresponds to the first-order sensitivities. Accordingly, first-order sensitivities can be calculated step wise by solving a linear matrix equation system with the coefficient matrix $J_x = \partial \mathrm{g}/\partial \mathrm{x}$. The right-hand side of Eq. 14 is defined by the partial derivatives w.r.t. the initial states and current contributions J_{x_0} and J_u, respectively.

$$J_x \cdot \mathrm{SL}' = -\left(J_{x_0} \cdot \mathrm{SL}'_0 + J_u\right) \tag{15}$$

For a step-wise computation of second-order sensitivities, Eq. 14 is differentiated again w.r.t. u_0 and u. In the same way as for the first-order sensitivities and by considering the dependencies in Eqs. 11 and 13, the addition of the resulting terms for SL' in Eq. 14 yields (see Appendix A)

$$\mathrm{SL}'' = \frac{d^2\mathrm{x}}{du^2} = \frac{(\partial \mathrm{SL}')}{\partial \mathrm{u}_0} + \frac{\partial(\mathrm{SL}')}{\partial \mathrm{u}} \tag{16}$$

Thus, SL'' in Eq. 16 corresponds to the second-order sensitivities. Moreover, by adding the results for all terms in Eq. 14 and using the definitions for SL' and SL'' we then obtain[¶]

$$\begin{aligned}
&\left[\frac{\partial \mathrm{g}}{\partial \mathrm{x}} \otimes I_{\mathrm{Nu}}\right] \cdot \mathrm{SL}'' + \left[I_{\mathrm{Nx}} \otimes (\mathrm{SL}')^{\mathrm{T}}\right] \cdot \left[\frac{\partial}{\partial \mathrm{u}_0}\left(\frac{\partial \mathrm{g}}{\partial \mathrm{x}}\right) + \frac{\partial}{\partial \mathrm{u}}\left(\frac{\partial \mathrm{g}}{\partial \mathrm{x}}\right)\right] \\
&\quad + \left[\frac{\partial \mathrm{g}}{\partial \mathrm{x}_0} \otimes I_{\mathrm{Nu}}\right] \cdot \left[\frac{\partial}{\partial \mathrm{u}_0}\left(\frac{\partial \mathrm{x}_0}{\partial \mathrm{u}_0}\right) + \underbrace{\frac{\partial}{\partial \mathrm{u}}\left(\frac{\partial \mathrm{x}_0}{\partial \mathrm{u}_0}\right)}_{=0}\right] \\
&\quad + \left[I_{\mathrm{Nx}} \otimes \left(\mathrm{SL}'_0\right)^{\mathrm{T}}\right] \cdot \left[\frac{\partial}{\partial \mathrm{u}_0}\left(\frac{\partial \mathrm{g}}{\partial \mathrm{x}_0}\right) + \frac{\partial}{\partial \mathrm{u}}\left(\frac{\partial \mathrm{g}}{\partial \mathrm{x}_0}\right)\right] \\
&\quad + \left[\frac{\partial}{\partial \mathrm{u}_0}\left(\frac{\partial \mathrm{g}}{\partial \mathrm{u}}\right) + \frac{\partial}{\partial \mathrm{u}}\left(\frac{\partial \mathrm{g}}{\partial \mathrm{u}}\right)\right] = 0_{\mathrm{Nx}\cdot\mathrm{Nu}\times\mathrm{Nu}}
\end{aligned} \tag{17}$$

An explicit derivation of all terms in Eq. 17 and some reformulations gives

$$\begin{aligned}
&\left[\frac{\partial \mathrm{g}}{\partial \mathrm{x}} \otimes I_{\mathrm{Nu}}\right] \cdot \mathrm{SL}'' + \left[I_{\mathrm{Nx}} \otimes (\mathrm{SL}')^{\mathrm{T}}\right] \\
&\cdot \left[\frac{\partial^2 \mathrm{g}}{\partial \mathrm{x}\partial \mathrm{u}} + \frac{\partial^2 \mathrm{g}}{\partial \mathrm{x}\partial \mathrm{x}_0} \cdot \mathrm{SL}'_0 + \frac{\partial^2 \mathrm{g}}{\partial \mathrm{x}^2} \cdot \mathrm{SL}'\right] + \left[\frac{\partial \mathrm{g}}{\partial \mathrm{x}_0} \otimes I_{\mathrm{Nu}}\right] \cdot \mathrm{SL}''_0 \\
&+ \left[I_{\mathrm{Nx}} \otimes \left(\mathrm{SL}'_0\right)^{\mathrm{T}}\right] \cdot \left[\frac{\partial^2 \mathrm{g}}{\partial \mathrm{x}_0\partial \mathrm{u}} + \frac{\partial^2 \mathrm{g}}{\partial \mathrm{x}_0^2} \cdot \mathrm{SL}'_0 + \frac{\partial^2 \mathrm{g}}{\partial \mathrm{x}_0\partial \mathrm{x}} \cdot \mathrm{SL}'\right] \\
&\quad + \left[\frac{\partial^2 \mathrm{g}}{\partial \mathrm{u}^2} + \frac{\partial^2 \mathrm{g}}{\partial \mathrm{u}\partial \mathrm{x}_0} \cdot \mathrm{SL}'_0 + \frac{\partial^2 \mathrm{g}}{\partial \mathrm{u}\partial \mathrm{x}} \cdot \mathrm{SL}'\right] = 0_{\mathrm{Nx}\cdot\mathrm{Nu}\times\mathrm{Nu}}
\end{aligned} \tag{18}$$

From Eq. 18, it can be followed that the second-order sensitivities are obtained again by the solution of a linear matrix equation system with the coefficient matrix $[J_x \otimes I_{\mathrm{Nu}}]$. Thus, it can be formulated in the same way as in Eq. 15 by using the following short notation.

$$\begin{aligned}
[J_x \otimes I_{\mathrm{Nu}}] \cdot \mathrm{SL}'' = -\Big(&\left[I_{\mathrm{Nx}} \otimes (\mathrm{SL}')^{\mathrm{T}}\right] \\
&\cdot \left[H_{xu} + H_{xx_0} \cdot \mathrm{SL}'_0 + H_{xx} \cdot \mathrm{SL}'\right] + [J_{x_0} \otimes I_{\mathrm{Nu}}] \cdot \mathrm{SL}''_0 \\
&+ \left[I_{\mathrm{Nx}} \otimes (\mathrm{SL}'_0)^{\mathrm{T}}\right] \cdot \left[H_{x_0u} + H_{x_0x_0} \cdot \mathrm{SL}'_0 + H_{x_0x} \cdot \mathrm{SL}'\right] \\
&+ \left[H_{uu} + H_{ux_0} \cdot \mathrm{SL}'_0 + H_{ux} \cdot \mathrm{SL}'\right]\Big)
\end{aligned} \tag{19}$$

The local sensitivities SL'_0, SL''_0 are initialized with zero, when starting the integration in a new decision interval d. In Eq. 19, the first-order solutions from the current and past integration element SL' and SL'_0 are used.

For the evaluation of Eqs. 15 and 19, a linear matrix equation system has to be solved. In principle this can be done through reuse of the factorization of the iteration matrix J_x, which is used to solve the implicit equation system in Eq. 10 (simulation problem). The algorithmic implementation is discussed in section "Algorithmic Implementation".

Generation of global, mixed (local–global), and mixed (global) sensitivities

In contrast to local sensitivities $\mathrm{SL}'_{i,i}$, $\mathrm{SL}''_{i,i,i}$, where past u_0 and current contributions of u are considered (see Eq. 13), global sensitivities $\mathrm{SG}'_{i,j}$, $\mathrm{SG}''_{i,j,j}$, with $i > j$ represent only derivatives w.r.t. contributions of u_0 (see also section "Problem Formulation").** The derivation of the discretized equation system in Eq. 11 w.r.t. the independent contributions of u_0 yields

$$J_x \cdot \mathrm{SG}' = -\left(J_{x_0} \cdot \mathrm{SG}'_0\right) \tag{20}$$

In analogy to this, the derivation of Eq. 20 w.r.t. u_0 yields the formula for second-order global and mixed (global) sensitivities

$$\begin{aligned}
[J_x \otimes I_{\mathrm{Nu}}] \cdot \mathrm{SG}'' = -\Big(&\left[I_{\mathrm{Nx}} \otimes (\mathrm{SG}')^{\mathrm{T}}\right] \\
&\cdot \left[H_{xx_0} \cdot \mathrm{SG}'_0 + H_{xx} \cdot \mathrm{SG}'\right] + [J_{x_0} \otimes I_{\mathrm{Nu}}] \cdot \mathrm{SG}''_0 \\
&+ \left[I_{\mathrm{Nx}} \otimes (\mathrm{SG}'_0)^{\mathrm{T}}\right] \cdot \left[H_{x_0x_0} \cdot \mathrm{SG}'_0 + H_{x_0x} \cdot \mathrm{SG}'\right]\Big)
\end{aligned} \tag{21}$$

In contrast to local sensitivities, which are initialized with zero, when stepping into a new decision interval i, global sensitivities SG'_0, SG''_0 are initialized by their corresponding values of the past local sensitivities $\mathrm{SL}'_{j,j}$, $\mathrm{SL}''_{j,j,j}$, with $j = i - 1$, or alternatively, by values of the past global sensitivities $\mathrm{SG}'_{j,k}$, $\mathrm{SG}''_{j,k,k}$, with $j = i - 1$, $j > k$ (notation according to Eq. 6). Using Eq. 21, the computation of second-order mixed (global) sensitivities is done in the same way.

Moreover, to obtain a formula for the step-wise integration of second-order mixed (local–global) sensitivities $\mathrm{SM}''_{i,i,j}$, with $i > j$, the following formula is finally used**

[¶]For the corresponding notations and results regarding second-order matrix derivatives, the interested reader is referred to Ref. 35.

**For the sake of simplicity, all indices i,j, and k are dropped in this section.

$$[J_x \otimes I_{\mathrm{Nu}}] \cdot \mathrm{SM}'' = -\Big(\Big[I_{\mathrm{Nx}} \otimes (\mathrm{SL}')^{\mathrm{T}}\Big] \cdot \big[H_{xx_0} \cdot \mathrm{SG}'_0 + H_{xx} \cdot \mathrm{SG}'\big] + [J_{x_0} \otimes I_{\mathrm{Nu}}] \cdot \mathrm{SM}''_0 + \Big[I_{\mathrm{Nx}} \otimes (\mathrm{SL}'_0)^{\mathrm{T}}\Big] \cdot \big[H_{x_0x_0} \cdot \mathrm{SG}'_0 + H_{x_0x} \cdot \mathrm{SG}'\big] + \big[H_{ux_0} \cdot \mathrm{SG}'_0 + H_{ux} \cdot \mathrm{SG}'\big] \Big) \quad (22)$$

The global sensitivities SG'_0 are then initialized by their corresponding values of the past local sensitivities $\mathrm{SL}'_{j,j}$, with $j = i - 1$, or alternatively, by values of the past global sensitivities $\mathrm{SG}'_{j,k}$, with $j = i - 1$, $j > k$. SL'_0, SM''_0 are initialized with zero.

Reformulation to obtain directional derivatives

In the last section above, a computation scheme for the generation of first- and second-order sensitivities has been derived, which is based on the explicit computation of Jacobian and Hessian matrix information. Full first- and second-order sensitivity information SL′, SG′, SL″, SG″, SM″ is obtained by the solution of a linear matrix equation system.

However, the influence of individual parameters and certain parameter space directions can be computed individually by solving a corresponding linear equation system. To do so, a reformulation is proposed, which is aimed at computing directional derivatives. The corresponding calculation rules allow to make use of AD tools, which provide directional derivative information in an efficient manner.[36] It will be shown that the use of AD techniques is especially interesting for the evaluation of the right-hand side of the linear equation systems of these calculation rules.

Generally, a first-order directional derivative $d' \in \mathbb{R}^{\mathrm{Nx}}$ is defined by $d' = \mathrm{SL}' \cdot \delta^1_u$, where $\delta^1_u \in \mathbb{R}^{\mathrm{Nu}}$ represents a specific direction. In Eq. 15, all terms are postmultiplied as $\{\cdots\} \cdot \delta^1_u$, and thus, the following linear equation system is obtained for the iterative step-wise calculation of local directional derivatives $dl'(\delta^1_u)$

$$J_x \cdot dl'(\delta^1_u) = -\big(J_{x_0} \cdot dl'_0(\delta^1_u) - J_u \cdot \delta^1_u\big) \quad (23)$$

Similarly, $dl'' \in \mathbb{R}^{\mathrm{Nx}}$ is used to indicate a second-order local directional derivative, which is defined by $dl'' = [I_{\mathrm{Nx}} \otimes (\delta^1_u)^{\mathrm{T}}] \cdot \mathrm{SL}'' \cdot \delta^2_u$ while using an additional direction $\delta^2_u \in \mathbb{R}^{\mathrm{Nu}}$. In Eq. 19, all terms are postmultiplied as $\{\cdots\} \cdot \delta^2_u$, and premultiplied as $[I_{\mathrm{Nx}} \otimes (\delta^1_u)^T] \cdot \{\cdots\}$. Consequently, the following linear equation system is obtained for the calculation of $dl''(\delta^1_u, \delta^2_u)$ (see Appendix B)

$$J_x \cdot dl''(\delta^1_u, \delta^2_u) = -\Big(\Big[I_{\mathrm{Nx}} \otimes \big(dl'(\delta^1_u)\big)^{\mathrm{T}}\Big] \cdot \big[H_{xu} \cdot \delta^2_u + H_{xx_0} \cdot dl'_0(\delta^2_u) + H_{xx} \cdot dl'(\delta^2_u)\big] + J_{x_0} \cdot dl''_0(\delta^1_u, \delta^2_u) + \Big[I_{\mathrm{Nx}} \otimes \big(dl'_0(\delta^1_u)\big)^{\mathrm{T}}\Big] \cdot \big[H_{x_0u} \cdot \delta^2_u + H_{x_0x_0} \cdot dl'_0(\delta^2_u) + H_{x_0x} \cdot dl'(\delta^2_u)\big] + \Big[I_{\mathrm{Nx}} \otimes (\delta^1_u)^{\mathrm{T}}\Big] \cdot \big[H_{uu} \cdot \delta^2_u + H_{ux_0} \cdot dl'_0(\delta^2_u) + H_{ux} \cdot dl'(\delta^2_u)\big] \Big) \quad (24)$$

The first integration step in a new decision interval d is initialized setting the local sensitivities $dl'_0(\delta^1_u)$, $dl'_0(\delta^2_u)$, $dl''_0(\delta^1_u, \delta^2_u)$ to zero in the same way as in Eqs. 15 and 19. Moreover, the second-order outcome depends on the solution of Eq. 23 in both the current and the past interval.

Furthermore, derivatives in arbitrary directions can be calculated, using Eqs. 23 and 24 while including those elementary directions, which can be used to compute single elements of the first- and second-order sensitivity matrices SL′, SL″.

It can be seen from Eqs. 15, 20 and Eqs. 19, 21, 22, that the difference in computing local and global sensitivities [as well as mixed (global and local–global) sensitivities] SL′, SG′ and SL″, SG″, (SM″) consists basically in the consideration or neglection of the following matrices: J_u and H_{xu}, H_{ux}, H_{x_0u}, H_{ux_0}, H_{uu}. Taking these simplifications into account, global and mixed directional derivatives can be computed using the Eqs. 23 and 24.

Algorithmic implementation

In this section, specific adaptations of the proposed algorithms are described, which allow the efficient use of the software for both the computation of higher-order directional derivatives and the solution of linear implicit equation systems while using an implicit one-step integration method.

Generalized approach for the calculation of directional derivatives

The different directional derivative terms (Jacobian vector products) on the right-hand side of Eq. 23 can be rewritten using a generalized Jacobian as shown in Eq. 25.

$$J_x \cdot dl'(\delta^1_u) = -[J_x \quad J_{x_0} \quad J_u] \cdot \begin{bmatrix} 0_{\mathrm{Nx}} \\ dl'_0(\delta^1_u) \\ \delta^1_u \end{bmatrix} \quad (25)$$

Likewise, the right-hand side of Eq. 24 can be formulated using the generalized Jacobian and Hessian as well as the general direction vectors, which result from the solution of Eq. 25. In Eq. 26, the definition of the second-order tensor T is given component-wise for each single discrete equation, g_i, with $i \in 1, \cdots, \mathrm{Nx}$ (see Eq. (11)) to keep the illustration simple.

$$J_x \cdot dl''(\delta^1_u, \delta^2_u) = -[T_1,\ T_2,\ \cdots,\ T_{\mathrm{Nx}}]^{\mathrm{T}} - J_{x_0} \cdot dl''_0(\delta^1_u, \delta^2_u) \quad (26)$$

$$\text{with} \quad T_i = \big[dl'(\delta^2_u),\ dl'_0(\delta^2_u),\ \delta^2_u\big] \cdot \begin{bmatrix} H_{xx} & H_{xx_0} & H_{xu} \\ H_{x_0x} & H_{x_0x_0} & H_{x_0u} \\ H_{ux} & H_{ux_0} & H_{uu} \end{bmatrix}_i \cdot \begin{bmatrix} dl'(\delta^1_u) \\ dl'_0(\delta^1_u) \\ \delta^1_u \end{bmatrix}$$

Specific adaptations for the integration of linear implicit equation systems using the OCFE method

All formula have been derived, so far, for the case of a fully implicit equation system (Eq. 1), which is solved using an implicit one-step integration approach.

However, for a linear implicit model (see Eq. 2) and the OCFE integration method, some simplifications can be carried out.[‡] Using the definitions in Eq. 9, the discrete equation system for the current integration step s (see Eq. 11) can then be written as a linear combination of functions $g^{\ominus}$ and $g^{\odot}$, which are computed using either x_0 or x. This is shown in Eq. 27a, where the dependencies on the free

decisions are considered (compare with Eq. 11). In the same way, the consideration of the parameters yields Eq. 27b.

$$0_{\mathrm{Nx}} = \mathrm{g}(\mathrm{x}_0(\mathrm{u}_0),\ \mathrm{x}(\mathrm{x}_0(\mathrm{u}_0),\ \mathrm{u}),\ \mathrm{u}) = \mathrm{g}^{\ominus}(\mathrm{x}_0(\mathrm{u}_0)) + \mathrm{g}^{\odot}(\mathrm{x}(\mathrm{x}_0(\mathrm{u}_0),\ \mathrm{u}),\ \mathrm{u}) \quad (27a)$$

$$0_{\mathrm{Nx}} = \mathrm{g}(\mathrm{x}_0(\mathrm{p}_0),\ \mathrm{x}(\mathrm{x}_0(\mathrm{p}_0),\ \mathrm{p}),\ \mathrm{p}) = \mathrm{g}^{\ominus}(\mathrm{x}_0(\mathrm{p}_0),\ \mathrm{p}) + \mathrm{g}^{\odot}(\mathrm{x}(\mathrm{x}_0(\mathrm{p}_0),\ \mathrm{p}),\ \mathrm{p}) \quad (27b)$$

A difference can be noticed while comparing the dependencies on the discrete decisions and parameters in Eqs 27a and 27b, which results from the fact that the decision variables are generally not connected to the differential variables in the process model (see 2). In other words, $\mathrm{g}^{\ominus} \neq f(\mathrm{u})$. In any case, from the linear combination of $\mathrm{g}^{\ominus}$ and $\mathrm{g}^{\odot}$, it follows then directly that the computation of second-order derivatives can be simplified significantly, as the matrices H_{xx_0}, H_{x_0x} vanish. Moreover, all function values as well as derivative terms can then be evaluated separately for any discrete point in time (e.g., x_0, x), which is in particular interesting when writing a general computer code, in which the model equations are implemented in routines for the evaluation of g^{lhs}, g^{rhs} and h, as previously defined in Eq. 2. Accordingly, the integrator can then be interfaced with these routines as well as routines for the evaluation of their corresponding derivatives.

Thus, we get for the step-wise integration of local directional derivatives w.r.t. parameters

$$J_x \cdot dl'(\delta_p^1) = -[J_{x_0}\ J_p]^{\ominus} \cdot \begin{bmatrix} dl_0'(\delta_p^1) \\ \delta_p^1 \end{bmatrix} - [J_x\ J_p]^{\odot} \cdot \begin{bmatrix} 0_{\mathrm{Nx}} \\ \delta_p^1 \end{bmatrix} \quad (28)$$

and

$$J_x \cdot dl''(\delta_p^1, \delta_p^2) = -[T_1,\ T_2,\ \cdots,\ T_{\mathrm{Nx}}]^{\mathrm{T}} - J_{x_0}^{\ominus} \cdot dl_0''(\delta_p^1, \delta_p^2)$$
$$T_i = \left[dl_0'(\delta_p^2),\ \delta_p^2\right] \cdot \begin{bmatrix} H_{x_0x_0} & H_{x_0p} \\ H_{px_0} & H_{pp} \end{bmatrix}_i^{\ominus} \cdot \begin{bmatrix} dl_0'(\delta_p^1) \\ \delta_p^1 \end{bmatrix} + \left[dl'(\delta_p^2),\ \delta_p^2\right] \cdot \begin{bmatrix} H_{xx} & H_{xp} \\ H_{px} & H_{pp} \end{bmatrix}_i^{\odot} \cdot \begin{bmatrix} dl'(\delta_p^1) \\ \delta_p^1 \end{bmatrix} \quad (29)$$

Likewise, but considering the differences in Eqs. 27a and 27b, we obtain for the step-wise integration of local directional derivatives w.r.t. decisions

$$J_x \cdot dl'(\delta_u^1) = -[J_{x_0}\ J_u]^{\ominus} \cdot \begin{bmatrix} dl_0'(\delta_u^1) \\ 0_{\mathrm{Nu}} \end{bmatrix} - [J_x\ J_u]^{\odot} \cdot \begin{bmatrix} 0_{\mathrm{Nx}} \\ \delta_u^1 \end{bmatrix} \quad (30)$$

and

$$J_x \cdot dl''(\delta_u^1, \delta_u^2) = -[T_1,\ T_2,\ \cdots,\ T_{\mathrm{Nx}}]^{\mathrm{T}} - J_{x_0}^{\ominus} \cdot dl_0''(\delta_u^1, \delta_u^2)$$
$$T_i = \left[dl_0'(\delta_u^2),\ 0_{\mathrm{Nu}}\right] \cdot \begin{bmatrix} H_{x_0x_0} & H_{x_0u} \\ H_{ux_0} & H_{uu} \end{bmatrix}_i^{\ominus} \cdot \begin{bmatrix} dl_0'(\delta_u^1) \\ 0_{\mathrm{Nu}} \end{bmatrix} + \left[dl'(\delta_u^2),\ \delta_u^2\right] \cdot \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix}_i^{\odot} \cdot \begin{bmatrix} dl'(\delta_u^1) \\ \delta_u^1 \end{bmatrix} \quad (31)$$

It should be noticed that global directional derivatives are calculated by Eqs. 30 and 31 by substituting the local directions $dl'(\delta_u^1)$, $dl'(\delta_u^2)$, $dl''(\delta_u^1, \delta_u^2)$ by their respective global counterpart, and by setting δ_u^1, δ_u^2 to zero. Moreover, a combination of local and global sensitivities yields the so-called mixed (local–global) directional derivatives.

Computer code

The algorithm has been implemented in the solver called sDACl, a sparse DAE solver based on the OCFE method.[34] Integration is performed using an orthogonal collocation based discretization along with the element-wise solution of the discretized nonlinear equation system. The dynamic linear implicit model is defined by the user supplied functions g^{lhs}, g^{rhs}, h, see Eq. 2. ADOLC is used for the evaluation of the Jacobian and first-as well as second-order directional derivatives of these user functions. The Jacobian is assumed to have a sparse partly unordered structure. When choosing a discretization scheme of an order higher than one, the size of the discrete equation system (Eq. 10) depends on the order of the polynomial approximation Nc given by Nc · Nx. Accordingly, the dimension of the discrete iteration matrix J_x results in Nc · Nx × Nc · Nx.

For the solution of the nonlinear simulation problem in Eq. 10 and the linear sensitivity integration problems in Eqs. 28–31, the general sparse nonlinear equation solver NLEQ1S[37] as well as the linear equation solver UMFPACK[38] have shown to provide sound results. Furthermore, before the solver is called, sDACl needs to perform a cold start with some preliminary steps. They are among others: verification of the user supplied information, recording of the user functions by ADOLC, memory allocation, extraction of sparsity pattern of the Jacobian for the user functions as well as of the discrete iteration matrix, preordering of nonzero elements and storage of the element placement for discretization, and symbolic factorization of the iteration matrix by UMFPACK. Provided that there is no structural change in the user supplied functions, the cold start has to be performed only once and the problem defined in Eq. 2 can then be solved for different solution parameters using the warm start option.

Moreover, a step size control (time derivative analysis[32]) is implemented to guarantee the accuracy of the state profile. After each successful step of the state integration, the sensitivities are evaluated for each direction solving the linear equation systems Eqs. 28–31. For this purpose, the LU factorization of the iteration matrix J_x from the current state integration step is reused. The simultaneous evaluation of derivatives in various directions speeds up then the computation time by using AD tools.[39] When integrating sensitivities in multiple directions, this fact is exploited for the construction of the right-hand side matrices in Eqs. 28–31. Additionally, when full second-order information is needed, S'' can be constructed efficiently by exploiting its symmetry, and thus, generating only the diagonal and upper diagonal elements. The general algorithm for the integration and staggered generation of sensitivities w.r.t. discrete decisions u_i, with $i = 1, \cdots, \mathrm{Nd}$ is shown in Algorithm 1.

Performance Evaluation

Case studies

Sequencing Batch Reactor – SBR. The first case study considers a sequencing batch reactor (SBR) process for waste water treatment. The equation system is derived from a state-of-the-art activated sludge model extended for two-

Algorithms 1: Step-wise integration of model equations and generation of first- and second-order sensitivities (as defined in Eqs. 5 and 6) using the staggered approach

```
/*Initialize sDACl solver                                          */
Set initial values for x0, u, p, t, Δt
Warm up sDACl (see section "Algorithmic Implementation")
/*Solution over all decision intervals                             */
for i = 1 to Nd do
    Update u for current decision interval i
    Initialize SL'_{i,i} = 0 and SL''_{i,i,i} = 0
    for j = 1 to i − 1 do
        Initialize SM''_{i,j,i} = 0
    while (end of decision interval is not reached) do
        /* Integrate model for current time step                    */
        repeat
            Discretize equation system using OCFE
            Solve for x (Eq. 10)
            Step size control update Δt
        until (integration tolerance is sufficient)
        Store LU-decomposition of iteration matrix matrix J_x
        /* Generate first-order sensitivities                       */
        Calculate SL'_{i,i} (Eq. 30 ∀ δ_u^1)
        for j = 1 to i − 1 do
            Calculate SG'_{i,j} (modified Eq. 30 ∀ δ_u^1)
        /* Generate second-order sensitivities                      */
        Calculate SL''_{i,i,i} (Eq. 31 ∀ δ_u^1,δ_u^2)
        for j = 1 to i do
            if j < i then
                Calculate SG''_{i,j,j} (modified Eq. 31 ∀ δ_u^1,δ_u^2)
            for k = 1 to j − 1 do
                Calculate SM''_{i,k,j} (modified Eq.31 ∀ δ_u^1,δ_u^2)
                Set symmetric elements SM''_{i,j,k} = SM''_{i,k,j}
        Update for next time step x0 = x, ···
    /* Initialize next decision interval                            */
    for j = 1 to i do
        Initialize SG'_{i+1,j} = SL'_{i,j} and SG''_{i+1,j,j} = SG''_{i,j,j}
    for j = 1 to i − 1 do
        for k = 2 to i do
            SM''_{i+1,j,k} = SM''_{i,j,k}
```

step nitrification–denitrification process.[40] It is an explicit index one DAE system of the type

$$\frac{dx(t)}{dt} = g^{\mathrm{rhs}}(x(t),u(t),p)$$
$$0 = h(x(t),p,t)\ , \quad x(t_0) = x_0 \qquad (32)$$

The process control variable is the air flux fed to the tank, which is optionally turned on and off so as to induce either aerobic or anoxic conditions. Figure 2 shows oxygen, nitrite, and nitrate concentrations for the scenario considered here.

This case study represents a small and stiff dynamic system. Due to the sharp changes in the concentrations, a relatively large effort is spent for the selection of an adequate step size for the integration. Table 1 provides the problem type and dimensions.

Distillation of Fatty Acids – DiCol. In the second case study, a packed distillation column operated at vacuum is considered for the separation of fatty acids. The detailed and experimentally validated dynamic model has been found to soundly represent different industrial scenarios.[41] It considers eight different organic carbon compounds, complex static and dynamic hold-up relations, energy balances, constant pressure drop, and vapour–liquid equilibrium relations. The detailed modeling of the organic carbon compounds includes physical property relations taken from the industry-standard DIPPR data base. The column model is an index two implicit DAE system of the type

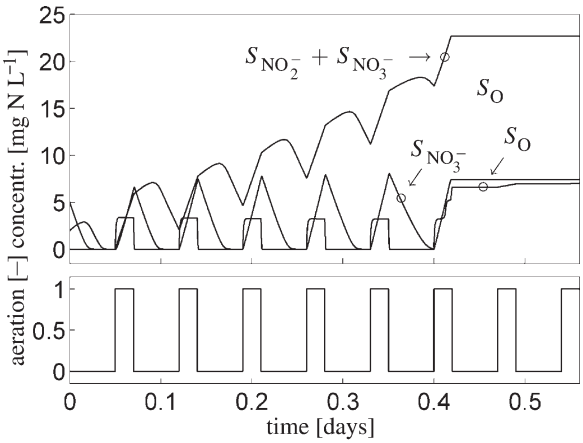

Figure 2. Case study SBR: operation scenario with the air flux fed to the tank turned on and off and resulting evolution of oxygen, nitrite, and nitrate concentrations.

$$\frac{dg^{\mathrm{lhs}}(x(t))}{dt} = g^{\mathrm{rhs}}(x(t),u(t),p)$$
$$0 = h(x(t),p,t)\ , \quad x(t_0) = x_0 \qquad (33)$$

The case study considers a product changeover in a minimum time. The manipulated operating parameters are: reboiler duty and reflux ratio. The interested reader is referred to Ref. 34, for details regarding the process model and the solution of the related dynamic optimization problem.

This case study represents a highly nonlinear model with a high number of operations for the evaluation of the problem functions. Table 1 provides details regarding the problem type and dimensions.

High Performance Liquid Chromatography – HPLC. The third case study considers a high-performance liquid chromatography (HPLC) separation process. A widely used HPLC model represents the so-called equilibrium transport dispersive model, which includes different mass transfer mechanisms, and it can be used with general nonlinear and multicomponent adsorption isotherms. Two fluid phases are considered, a bulk phase that is moving in axial column direction, and a stationary particle phase, where adsorption

Table 1. Problem Type and Dimensions for the Three Case Studies

Case Study	SBR	DiCol	HPLC
Problem type	Explicit DAE	Linear implicit DAE	Linear implicit DE
Problem index	1	2	0
#States	19	420	1,206
#Differential equations	11	179	1,206
#Algebraic equations	8	241	—
Problem function*: g, h			
#Operations	152/331	6,904/39,987	14,683/14,032
Problem Jacobian*: J_x			
#Nonzero elements	11/76	510/3,156	4,212/4,212
Sparsity	3.05%/21.05%	0.29%/1.79%	0.29%/0.29%
Iteration matrix**			
Dimension	57×57	$1{,}260\times1{,}260$	$3{,}618\times3{,}618$
#Nonzero elements	324	13,065	41,526
Sparsity	9.97%	0.82%	0.32%

*lhs/rhs.
**Discretization of third order, Nc = 3.

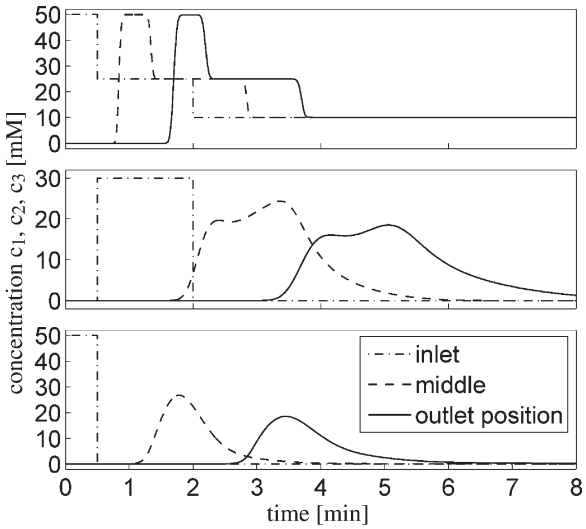

Figure 3. Simulation of the HPLC for three different species.

Column middle and outlet concentrations are depicted for step changes of the column inlet concentrations.

takes place. The resulting mathematical model is a two-dimensional (axial direction and time) partial differential equation system for the bulk phase coupled with an implicit differential equation system for the particle phase. The model is defined for each species involved in the separation process. After discretization in space using the Galerkin finite element method with second-order polynoms, an index zero implicit differential equation system is obtained[42]

$$\frac{dg^{\mathrm{lhs}}(x(t),p)}{dt} = g^{\mathrm{rhs}}(x(t),u(t),p), \quad x(t_0) = x_0 \tag{34}$$

where some model parameters (adsorption parameters) are connected to the differential states. The separation of a three component mixture is simulated. The geometric column data and physical model parameters are adopted from Ref. 34. As axial concentration profiles can be very steep in particular near the column entry, the accuracy and stability of the numeric solution depend strongly on the number of elements used for the discretization in axial direction. Here, 100 elements are used. In Figure 3, simulation results are given for concentration step changes within the feed. The results at the column outlet as well as in the middle position of the column are depicted for all three components.

This case study represents a large, sparse, and highly stiff dynamic system. Table 1 provides details regarding the problem type and dimensions.

Computation time

All computations were carried out with double precision using the GNU C/C++ compiler version 4.5.2 for 32 bit Linux platform with an Intel Core 2 Duo (U9400 @1.4 GHz) computer with 3-GB RAM. Parallel programming has not been considered.

Sensitivities w.r.t. Parameters. As discussed in section "Problem Formulation", local sensitivities have to be computed over the entire simulation time horizon for the evaluation of the complete set of first- and second-order sensitivities w.r.t. parameters. Accordingly, the total number of elements in the matrices S' and S'' increases linearly and quadratically with the number of parameters, respectively. As each element in S' and S'' represents one single direction, which is integrated along with the states, the additional computational effort compared with one state integration increases also linearly and quadratically for the evaluation of first- and second-order sensitivities. In Figure 4, the computational effort referred to one state integration is depicted for an increasing number of parameters for all three case studies. Although the selected case studies are of a very different problem type (see Table 1), they confirm the general dependency on the number of parameters as discussed above. The differences in the relative effort for 10–20 parameters (Figure 4, bottom) can be explained by a relative large overhead in the evaluation of the Jacobian and various directional derivatives of the user functions. The time needed for the evaluation using ADOLC can generally take up to $\leq$50% of the overall computation time, and thus, the complexity of the derivatives can then influence the result dramatically. Additionally, the model used for case study SBR is not a large one. The staggered integration approach benefits mostly from systems where the time for the lower and upper triangular matrix (LU) factorizations dominates over the total computation time. This is the case for large-scale process models. Table 2 shows detailed solution statistics for the three cases considered when sensitivities w.r.t. 10 and 12 parameters are evaluated.

Sensitivities w.r.t. Discrete Decisions. The complete set of first- and second-order local, global, and mixed sensitivities w.r.t. discrete decision variables over the entire simulation time horizon are shown in Eqs. 5 and 6. The total number of elements (nels) in Eq. 5: $\mathrm{SL}'_{i,i}$, $\mathrm{SG}'_{i,j}$ and Eq. 6: $\mathrm{SL}''_{i,i,i}$, $\mathrm{SG}''_{i,j,j}$, $\mathrm{SM}''_{i,j,k}$ increases quadratically and cubically with the number of decision intervals Nd used for control profile discretization, respectively

$$\begin{aligned} \mathrm{nels}S' &= \frac{\mathrm{Nd}^2 + \mathrm{Nd}}{2} \\ \mathrm{nels}S'' &= \frac{2\cdot \mathrm{Nd}^3 + 3\cdot \mathrm{Nd}^2 + \mathrm{Nd}}{6} \end{aligned} \tag{35}$$

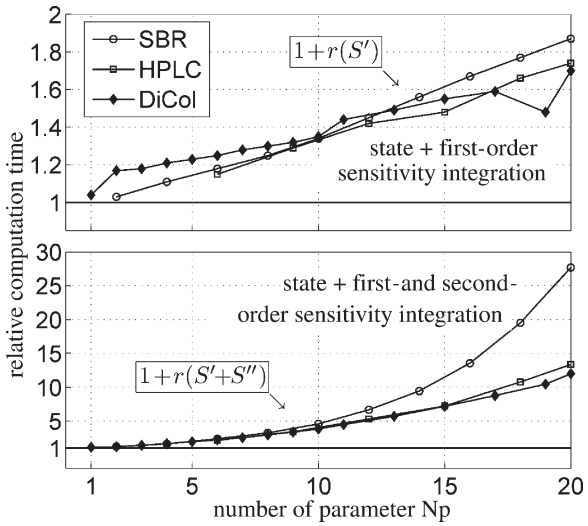

Figure 4. Relative computation time w.r.t. one state integration for an increasing number of model parameters.

Results are given for the evaluation of both one state and first-order sensitivity integration (top) and one state and first plus second-order sensitivity integration (bottom).

Table 2. Solution Statistics for the Three Case Studies for the Integration of States, First- and Second-Order Sensitivities

Sensitivities w.r.t.	Constant Parameters			Discrete Decisions		
Case Study	SBR (Np = 10)	DiCol (Np = 10)	HPLC (Np = 12)	SBR (Nu = 10)	DiCol (Nu = 10)	HPLC (Nu = 12)
State integration*						
#Steps	472/543	36/36	219/221	257/292	52/52	163/163
Sensitivity integration**						
#Local steps	4,720/25,960	360/1,980	2,628/17,082	257/257	52/52	163/163
#Global steps	−/−	−/−	−/−	780/780	76/76	401/401
#Mixed steps	−/−	−/−	−/−	−/1,986	−/252	−/1,212
Iteration matrix						
#LU decompositions	1,413	109	874	707	173	641
#Linear system solutions	32,093	2,449	20,584	4,767	681	2,981
Function evaluations***						
#Calls to g, h	5,872/5,870	441/439	3,291/3,289	3,000/2,998	681/679	2,418/2,416
#Calls to J_x	4,241/4,241	331/331	2,626/2,626	2,122/2,122	523/523	1,927/1,927
#Calls to $J \cdot d_x$	30,680/−	2,340/−	19,710/−	4,050/−	498/−	2,328/−
#Calls to $J \cdot d$	20,768/15,576	1,584/1,188	11,388/8,541	−/1,542	−/312	−/978
#Calls to $d^T \cdot H \cdot d$	124,608/93,456	9,504/7,128	79,716/59,787	12,092/9,069	1,520/1,140	7,104/5,328
Computation time						
State integration	0.74 s	3.56 s	25.76 s	0.34 s	5.39 s	17.84 s
+ First-order sens.	0.99 s	4.80 s	36.69 s	0.39 s	5.77 s	19.04 s
+ Second-order sens.	3.38 s	13.64 s	136.86 s	0.63 s	7.46 s	27.34 s

*Successfull/ trials.
**First and Second order.
***lhs/rhs.

To keep the discussion simple, we refer to the case where the number of continuous decision variables is Nu = 1 and assume an equidistant spacing of all decisions u_i over the simulation time horizon. The additional effort in comparison to the state integration can be estimated according to Eq. 36.

$$r(S') \propto \frac{\text{nels}S'}{\text{Nd}}$$
$$r(S' + S'') \propto \frac{\text{nels}S' + \text{nels}S''}{\text{Nd}} \tag{36}$$

It can be seen that an increase in the number of discrete decisions means a weak linear ($r(S')$) and quadratic ($r(S' + S'')$) increase in the computational effort, which is due to the fact that global and mixed global sensitivities do not need to be integrated over the entire simulation time horizon. This is confirmed by those results in Figure 5, where the computational effort referred to one state integration is shown for an increasing number of discrete decisions for all three case studies. However, it should be noted that this results hold only for an equidistant spacing and differ for a different spacing of the decision intervals, that is, their individual time lengths.

Comparison with other approaches

For a general performance evaluation of the studied OCFE integration method applied to single shooting optimization, the interested reader is referred to Ref. 29. In comparison to an implementation of the well-known multiple shooting method, the proposed approach performs superior in particular for large-scale dynamic optimization problems (large number of free decision variables), although it should be noted that tests have only be performed for small-scale dynamic equation systems. However, the adaptation of the approach in Ref. 29 for the efficient handling of large-scale dynamic systems by sparse matrix calculus has been already presented elsewhere,[34] and thus, it is likely that a similar performance can be expected for large-scale dynamic systems also.

From the three numerical case studies presented in this work, it can be observed that in comparison to the effort for one state integration, the additional effort for the generation of one single directional derivative (first or second order) represents only a small fraction of around 5% of the simulation time for large dynamic systems. This fraction depends mainly on two different values. First, it corresponds to the required number of Newton type iterations performed in each state integration step, which means then that the approach benefits basically from highly nonlinear systems. Second, this fraction depends on the number of steps, which are rejected in the step-size control, in other words, the approach is in particular efficient for systems with difficult dynamics.

This relatively small computational effort has also been reported for different implicit integration schemes.[14] In

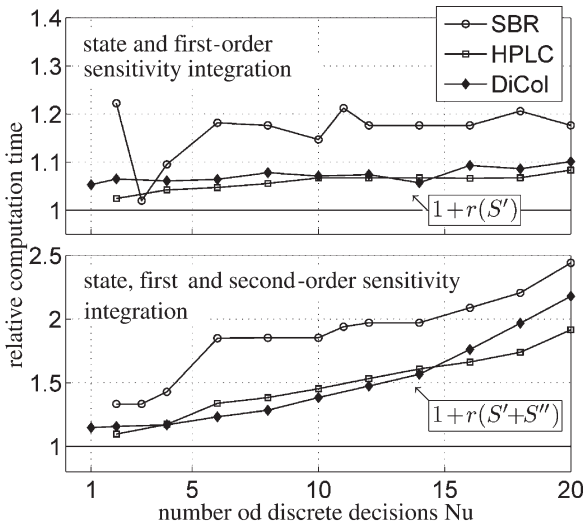

Figure 5. Relative computation time w.r.t. one state integration for an increasing number of elements used for discretization of the control profile.

Results are given for the evaluation of one state and first-order sensitivity integration (top) as well as for one state and first plus second-order sensitivity integration (bottom).

contrast to a fourth-order explicit Runge–Kutta scheme, where the time for solving a first-order adjoint and a directional second-order sensitivity is equivalent to that of four and 13 simulation times, respectively, using an implicit integration method, second-order directional derivatives are cheaply obtained by an additional computing time of 15% only.[14] Different performance results are reported for applications of the adjoint approach to the BDF integration method. A theoretical bound of maximum five times the cost of a system simulation for a discrete first-order directional gradient has generally been assumed.[13,21] For a different implementation, the computational cost for one directional second-order derivative has been reported to be between two and four simulation times.[6] Similarly, for one-step methods of implicit Runge–Kutta and Rosenbrock type, the cost of a first-order adjoint computation is slightly over two times, wheareas the cost of a second-order adjoint calculation is three and a half times the cost of a simulation.[23]

In contrast to the aforementioned references, in this contribution, the generation of complete Jacobian and Hessian information has been considered. The efficiency of this implementation for second-order derivatives deteriorates with an increasing number of free parameters (in the presented case studies higher than 15) in comparison to the results from other authors using the adjoint mode because of the quadratic increase in computation time. However, for the generation of sensitivities w.r.t. discrete decision variables (individual elements of a control trajectory), the computational effort can be reduced to a weak quadratic increase only. This is mainly due to the fact that discrete decision variables do not have an influence on the entire simulation horizon. It leads to very small computing times for a complete Jacobian and Hessian generation while not exceeding 2.5 times the time for one state integration (up to 20 discrete decision variables).

The results obtained can directly be contrasted with other approaches where Jacobians are computed by forward sensitivity analysis and Hessians by second-order discrete adjoint equations as presented in Ref. 22. Here, the computation time for a Jacobian and Hessian evaluation scales essentially linear w.r.t. the number of free parameters and a full Hessian evaluation can be expected to be over four to seven times longer than that for a Jacobian evaluation.[22] It can thus be concluded that the presented approach performs especially well in comparison to the discussed other approaches when generating sensitivities w.r.t. comparatively low-dimensional discrete decisions.

Conclusions

In this work, an approach to efficient forward generation of first- and second-order sensitivities for general and linear implicit equation systems is proposed. In contrast to the approach to forward generation of continuous sensitivities,[2] the generation of discrete sensitivities is discussed here for one-step integration methods. The implementation is shown for an implicit integration method, namely the OCFE method based on LGR points, while adopting the so-called staggered state and sensitivity integration approach. Using a high-order one-step method with exact nonlinear system solutions in each integration step, large step sizes together with accurate approximate solutions of the state profiles are obtained.[29,32] The staggered state and sensitivity generation scheme benefits mainly from this and also from the reuse of the factorized system Jacobian as well as the intermediate data. To emphasize this issue, numerical experiments have been carried out for three different case studies ranging from small and stiff to large, sparse, and complex application examples. In comparison to the effort for one state integration, the additional effort for the generation of one single directional derivative (first or second order) represents only a small fraction of around 5% of the simulation time for large dynamic systems.

However, the efficiency of the forward generation for second-order derivatives deteriorates with an increasing number of free parameters because of the quadratic increase in computation time. In contrast to this, the computational effort can be reduced to a weak quadratic increase for the generation of sensitivities w.r.t. discrete decision variables. Thus, it leads to very small computing times for a complete Jacobian and Hessian generation in comparison to the time needed for a state integration.

The here considered discrete sensitivities, correspond to the exact sensitivities of the solution of the time-discretized system. In Ref. 18, numerical experiments confirm that there is indeed little to gain by attempting to apply a local error test to the sensitivity values. However, the question whether it is preferable to use the continuous or the discrete gradient in an optimization procedure is still an open issue, and its answer depends undoubtedly on the underlying problem to be solved.[43]

Generally, the solution of NLP problems using full second-order information has not been considered competitive with standard Hessian free optimization methods when using the sequential approach[4,7,8,14,20,23] Therefore, the general focus is given to the efficient computation of Hessian-vector products of the Lagrangian. In this work, the proposed approach can be directly used for the generation of Hessian-vector products, where the vector specifies a direction in the parameter or the discrete decision space. Instead of the observed quadratic increase for the generation of a full Hessian, this leads then to a linear increase in computing time, which corresponds approximately to the time needed for a Jacobian generation.

To focus first on the general performance of the algorithm, parallel computing has not been exploited yet. It can anyhow be extended with very little modification of the source code to the parallel computation of directional derivatives using, for example, the standard Message Passing Interface (see also Refs. 16 and 18).

Moreover, the proposed algorithm is based on the evaluation of: Jacobians, first-, and second-order directional derivatives of the left- and right-hand side of the model equations. The operator overloading automatic differentiation (AD) software ADOLC has been used for their evaluation. Generally, the overall performance has shown to be inferior to the implementation using symbolically derived code (compare the results in Ref. 34). As discussed in Ref. 13, it is promising for an efficient implementation to substitute ADOLC by a different AD-based model derivative generation such as source code transformation together with a coding of the right-hand side optimized for AD or providing symbolically the derived source code.

Acknowledgments

This work is part of the Collaborative Research Center SFB/TR 63 InPROMPT "Integrated Chemical Processes in Liquid Multiphase Systems" coordinated by the Berlin Institute of Technology and funded

by the German Research Foundation. T.B. gratefully acknowledges the financial support of BMBF (Federal Ministry of Education and Research of Germany), support code: 03WOPAL4. The authors thank Dr.-Ing. Moritz Wendt (InfraServ GmbH & Co. Knapsack KG) for the cooperation and support regarding the second case study.

Literature Cited

1. Biegler LT, Grossmann IE. Retrospective on optimization. *Comput Chem Eng*. 2004;28(8):1169–1192.
2. Vassiliadis VS, Canto EB, Banga JR. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chem Eng Sci*. 1999;54(17):3851–3860.
3. Cao Y, Li S, Petzold L, Serban R. Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution. *SIAM J Sci Comput*. 2003;24(3):1076–1089.
4. Hannemann R, Marquardt W. Continuous and discrete composite adjoints for the Hessian of the Lagrangian in shooting algorithms for dynamic optimization. *SIAM J Sci Comput*. 2010;31:4675–4695.
5. Balsa Canto E, Banga JR, Alonso AA, Vassiliadis VS. Restricted second order information for the solution of optimal control problems using control vector parameterization. *J Process Control*. 2002; 12(2):243–255.
6. Özyurt DB, Barton PI. Cheap second order directional derivatives of stiff ODE embedded functionals. *SIAM J Sci Comput*. 2005;26(5): 1725–1743.
7. Balsa-Canto E, Banga JR, Alonso AA, Vassiliadis VS. Dynamic optimization of distributed parameter systems using second-order directional derivatives. *Ind Eng Chem Res*. 2004;43(21):6756–6765.
8. Vetukuri SRR, Biegler LT, Walther A. An inexact trust-region algorithm for the optimization of periodic adsorption processes. *Ind Eng Chem Res*. 2010;49(23):12004–12013.
9. Maly T, Petzold LR. Numerical methods and software for sensitivity analysis of differentialalgebraic systems. *Appl Numer Math*. 1996; 20(1):57–82.
10. Schlegel M, Marquardt W, Ehrig R, Nowak U. Sensitivity analysis of linearly-implicit differential-algebraic systems by one-step extrapolation. *Appl Numer Math*. 2004;48(1):83–102.
11. Caracotsios M, Stewart W. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. *Comput Chem Eng*. 1985;9(4):359–365.
12. Bock H. *Numerical treatment of inverse problems in chemical reaction kinetics*. In: Ebert K, Deuflhard P, Jäger W, editors. *Modelling of Chemical Reaction Systems, Springer Series in Chemical Physics*, Vol. 18. Heidelberg: Springer, 1981; 102–125.
13. Albersmeyer J, Bock HG. Efficient Sensitivity Generation for Large Scale Dynamic Optimization. Tech. Rep. SPP1253-01-02, Heidelberg, Germany: Interdisciplinary Center for Scientific Computing. http://www.am.uni-erlangen.de/home/spp1253/wiki/images/7/7f/Preprint-spp1253-01-02.pdf. 2009. Last accessed on: January 9, 2012.
14. Cioaca A, Alexe M, Sandu DA. Second order adjoints for solving PDE-constrained optimization problems. *Optimization Methods Software* 2010:1–31, DOI: 101080/105567882011610455.
15. Kristensen MR, Jørgensen JB, Thomsen PG, Jørgensen SB. An ESDIRK method with sensitivity analysis capabilities. *Comput Chem Eng*. 2004;28(12):2695–2707.
16. Li S, Petzold L. Software and algorithms for sensitivity analysis of large-scale differential algebraic systems. *J Comput Appl Math*. 2000;125(1–2):131–145.
17. Feehery WF, Tolsma JE, Barton PI. Efficient sensitivity analysis of large-scale differentialalgebraic systems. *Appl Numer Math*. 1997; 25(1):41–54.
18. Keeping BR, Pantelides CC. A distributed memory parallel algorithm for the efficient computation of sensitivities of differential-algebraic systems. *Math Comput Simul*. 1998;44(6):545–558.
19. Tocci MD. Sensitivity analysis of large-scale time dependent PDEs. *Appl Numer Math*. 2001;37(1–2):109–125.
20. Özyurt DB, Barton PI. Large-scale dynamic optimization using the directional second-order adjoint method. *Ind Eng Chem Res*. 2005; 44(6):1804–1811.
21. Wirsching L, Albersmeyer J, Kuehl P, Diehl M, Bock HG. An adjoint-based numerical method for fast nonlinear model predictive control. In: *Proceedings of the 17th IFAC World Congress, Seoul, Korea*, Elsevier: Oxford, UK, 2008:1934–1939.
22. Hannemann R, Marquardt W, Naumann U, Gendler B. Discrete first- and second-order adjoints and automatic differentiation for the sensitivity analysis of dynamic models. *Proc Comput Sci*. 2010;1(1): 297–305.
23. Sandu A, Zhang L. Discrete second order adjoints in atmospheric chemical transport modeling. *J Comput Phys*. 2008;227(12): 5949–5983.
24. Wang FS. A modified collocation method for solving differential-algebraic equations. *Appl Math Comput*. 2000;116(3):257–278.
25. Biegler LT. An overview of simultaneous strategies for dynamic optimization. *Chem Eng Process*. 2007;46(11):1043–1053.
26. Fornberg B. *A Practical Guide to Pseudospectral Methods*. Cambridge, New York, Oakleigh: Cambridge University Press, 1998.
27. Logsdon JS, Biegler LT. Accurate determination of optimal reflux policies for the maximum distillate problem in batch distillation. *Ind Eng Chem Res*. 1993;32(4):692–700.
28. Hong W, Wang S, Li P, Wozny G, Biegler LT. A quasi-sequential approach to large-scale dynamic optimization problems. *AIChE J*. 2006;52(1):255–268.
29. Tamimi J, Li P. A combined approach to nonlinear model predictive control of fast systems. *J Process Control*. 2010;20(9): 1092–1102.
30. Garg D, Patterson MA, Francolin C, Darby CL, Huntington GT, Hager WW, Rao AV. Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a Radau pseudospectral method. *Comput Optimization Appl*. 2011;49:335–358.
31. Kameswaran S, Biegler LT. Convergence rates for direct transcription of optimal control problems using collocation at Radau points. *Comput Optimization Appl*. 2008;41(1):81–126.
32. Bartl M, Li P, Biegler LT. Improvement of state profile accuracy in nonlinear dynamic optimization with the quasi sequential approach. *AIChE J*. 2011;57(8):2185–2197.
33. Hairer E, Wanner G. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd ed. Vol. 14. Berlin, Heidelberg, New York: Springer, 1996.
34. Barz T, Kuntsche S, Wozny G, Arellano-Garcia H. An efficient sparse approach to sensitivity generation for large-scale dynamic optimization. *Comput Chem Eng*. 2011;35(10):2053–2065.
35. Marlow WH. *Mathematics for Operations Research*. New York: Dover Publications, 1993.
36. Griewank A, Walther A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 2008.
37. Nowak U, Weimann L. A Family of Newton Codes for Systems of Highly Nonlinear Equations. Tech. Rep. TR-910, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany, 1991 http://www.zib.de/Publications/Reports/TR-91-10.pdf. Last accessed on: January 9, 2012.
38. Davis TA. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans Math Software*. 2004;30(2): 196–199.
39. Walther A, Griewank A. ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++. Tech. Rep. Version 2.1.3, Dresden, Germany: Institute of Scientific Computing, Technische Universität Dresden. https://projects.coin-or.org/ADOL-C. 2010. Last accessed on: September 20, 2011.
40. Cruz Bournazou MN, Arellano-Garcia H,Wozny G, Lyberatos G, Kravaris C. ASM3 extended for two-step nitrification-denitrification: a model reduction for Sequencing Batch Reactors. *J Chem Technol Biotechnol*. 2011; URL http://dx.doi.org/10.1002/jctb.3694; DOI 10.1002/jctb.3694; Article first published online: 6 JAN 2012.
41. Wendt M, Strack M, Arellano Garcia H, Barz T. Optimierung dynamischer Destillationsprozesse: Einsatz von modellbasierten Methodiken führen zu optimalen Führungsstrategien in Mehrproduktanlagen. *CIT Plus*. 2009;7–8:27–29.
42. Gu T. *Mathematical Modeling and Scale-up of Liquid Chromatography*. Berlin, New york: Springer, 1995.
43. Walther A. Automatic differentiation of explicit Runge–Kutta methods for optimal control. *Comput Optimization Appl*. 2007;36(1): 83–108.

Appendix A: Generation of Second-Order Sensitivities Based on the Decomposition Approach

From Eq. 13, we get for the second-order differential calculus w.r.t. u_0 and u

$$\frac{d^2}{du^2}[x] = \frac{\partial^2}{\partial u_0^2}[x] + 2 \cdot \frac{\partial^2}{\partial u_0 u}[x] + \frac{\partial^2}{\partial u^2}[x] \qquad \text{(A1)}$$

Considering the dependencies in Eq. 11, we get for the different terms in Eq. A1

$$\frac{\partial^2}{\partial u_0^2}[x] = \frac{\partial}{\partial u_0}\left[\frac{\partial x}{\partial x_0}\frac{\partial x_0}{\partial u_0}\right] = \left[\frac{\partial x}{\partial x_0}\otimes I_{Nu}\right]\cdot\frac{\partial^2 x_0}{\partial u_0^2} + \left[I_{Nx}\otimes\left(\frac{\partial x_0}{\partial u_0}\right)^T\right]\cdot\frac{\partial^2 x}{\partial x_0^2}\frac{\partial x_0}{\partial u_0} \quad \text{(A2a)}$$

$$\frac{\partial^2}{\partial u_0 \partial u}[x] = \frac{\partial}{\partial u_0}\left[\frac{\partial x}{\partial u}\right] = \frac{\partial^2 x}{\partial u \partial x_0}\frac{\partial x_0}{\partial u_0} \quad \text{(A2b)}$$

$$\frac{\partial^2}{\partial u \partial u_0}[x] = \frac{\partial}{\partial u}\left[\frac{\partial x}{\partial x_0}\frac{\partial x_0}{\partial u_0}\right] = \left[\frac{\partial x}{\partial x_0}\otimes I_{Nu}\right]\cdot\underbrace{\frac{\partial^2 x_0}{\partial u_0 \partial u}}_{=0} + \left[I_{Nx}\otimes\left(\frac{\partial x_0}{\partial u_0}\right)^T\right]\cdot\frac{\partial^2 x}{\partial x_0 \partial u} \quad \text{(A2c)}$$

$$\frac{\partial^2}{\partial u^2}[x] = \frac{\partial^2 x}{\partial u^2} \quad \text{(A2d)}$$

where Eqs. A2b and A2c are equivalent.

Differentiation of the term S' in Eq. 14 w.r.t. u_0 and u and adding both contributions gives

$$S'' = \frac{\partial(S')}{\partial u_0} + \frac{\partial(S')}{u} = \left[\frac{\partial x}{\partial x_0}\otimes I_{Nu}\right]\cdot\frac{\partial^2 x_0}{u_0^2} + \left[I_{Nx}\otimes\left(\frac{\partial x_0}{u_0}\right)^T\right]\cdot\left[\frac{\partial^2 x}{\partial x_0^2}\frac{\partial x_0}{\partial u_0}\right] + \frac{\partial^2 x}{\partial u \partial x_0}\frac{\partial x_0}{\partial u_0} + \left[\frac{\partial x}{\partial x_0}\otimes I_{Nu}\right]\cdot\underbrace{\frac{\partial^2 x_0}{\partial u_0 \partial u}}_{=0} + \left[I_{Nx}\otimes\left(\frac{\partial x_0}{\partial u_0}\right)^T\right]\cdot\frac{\partial^2 x}{\partial x_0 \partial u} + \frac{\partial^2 x}{\partial u^2} \quad \text{(A3)}$$

which results in the same outcome as for Eq. A1, and thus, verifies Eq. 16.

Appendix B: Generation of Second-Order Directional Derivatives

Second-order local directional derivatives in Eq. 24 are defined as

$$dl''(\delta_u^1, \delta_u^2) = [I_{Nx}\otimes(\delta_u^1)^T]\cdot SL''\cdot\delta_u^2 \quad \text{(B1)}$$

In the following, these definitions for the matrix products and their transposed are used[35]

$$(A\otimes B)\cdot(C\otimes D) = (A\cdot C)\otimes(B\cdot D); \quad A^T\cdot B^T = (B\cdot A)^T \quad \text{(B2)}$$

Premultiplication and postmultiplication of the left-hand side in Eq. 19 and using the rules in Eq. B2 yields the definition for $dl''(\delta_u^1, \delta_u^2)$ (see Eq. B1), which is then used in Eq. 24

$$I_{Nx}\otimes(\delta_u^1)^T]\cdot[J_x\otimes I_{Nu}]\cdot SL''\cdot\delta_u^2 = J_x\otimes(\delta_u^1)^T\cdot SL''\cdot\delta_u^2 = J_x\cdot[I_{Nx}\otimes(\delta_u^1)^T]\cdot SL''\cdot\delta_u^2 \quad \text{(B3)}$$

Premultiplication of the right-hand side in Eq. 19 by $[I_{Nx}\otimes(\delta_u^1)^T]$ and using the rules in Eq. B2 yields the multiplier of the first term on the right-hand side in Eq. 24

$$\left[I_{Nx}\otimes(\delta_u^1)^T\right]\cdot\left[I_{Nx}\otimes(SL')^T\right] = \left[I_{Nx}\otimes(\delta_u^1)^T\cdot(SL')^T\right] = \left[I_{Nx}\otimes\left(SL'\cdot\delta_u^1\right)^T\right] = \left[I_{Nx}\otimes\left(dl'(\delta_u^1)\right)^T\right] \quad \text{(B4)}$$

In the same way as in Eqs. B3 and B4, all other terms in Eq. 19 are reformulated to obtain Eq. 24.

Manuscript received July 19, 2011, revision received Oct. 4, 2011, and final revision received Dec. 20, 2011.