Efficient Symmetric Hessian Propagation for Direct Optimal Control[☆]

Rien Quirynen[a,c,*], Boris Houska[b], Moritz Diehl[c]

[a]*Department ESAT-STADIUS, KU Leuven University, Kasteelpark Arenberg 10, 3001 Leuven, Belgium.*
[b]*School of Information Science and Technology, ShanghaiTech University, 319 Yueyang Road, Shanghai 200031, China.*
[c]*Department IMTEK, University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany.*

Abstract

Direct optimal control algorithms first discretize the continuous-time optimal control problem and then solve the resulting finite dimensional optimization problem. If Newton type optimization algorithms are used for solving the discretized problem, accurate first as well as second order sensitivity information needs to be computed. This article develops a novel approach for computing Hessian matrices which is tailored for optimal control. Algorithmic differentiation based schemes are proposed for both discrete- and continuous-time sensitivity propagation, including explicit as well as implicit systems of equations. The presented method exploits the symmetry of Hessian matrices, which typically results in a computational speedup of about factor 2 over standard differentiation techniques. These symmetric sensitivity equations additionally allow for a three-sweep propagation technique that can significantly reduce the memory requirements, by avoiding the need to store a trajectory of forward sensitivities. The performance of this symmetric

[☆]This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC, Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012-2017); EU: FP7-TEMPO (MC ITN-607957), H2020-ITN AWESCO (642682), ERC HIGHWIND (259 166), National Science Foundation China (NSFC), Nr. 61473185, and ShanghaiTech University, Grant-Nr. F-0203-14-012. R. Quirynen holds a PhD fellowship of the Research Foundation – Flanders (FWO).

[*]Corresponding author. Email address: rien.quirynen@esat.kuleuven.be

sensitivity propagation is demonstrated for the benchmark case study of the economic optimal control of a nonlinear biochemical reactor, based on the open-source software implementation in the ACADO Toolkit.

Keywords: optimal control, sensitivity analysis, algorithms and software, nonlinear predictive control

---

1. Introduction

Optimal control problems (OCP) arise in various applications, including the optimization of an actuation profile, system state and parameter estimation or optimal experiment design problems [1]. Shooting based direct optimal control methods rely on an accurate discretization of the original continuous-time OCP, resulting in a finite dimensional nonlinear program (NLP) [2, 3]. In the context of nonlinear model predictive control (NMPC) or moving horizon estimation (MHE) [4], for example, these optimal control problems have to be solved under strict timing constraints [5]. In many practical control applications, the simulation of the nonlinear dynamics as well as the propagation of first and second order derivative information is the main bottleneck in terms of computation time. In particular, Newton type optimization algorithms [6] such as Interior Point (IP) methods [7] or Sequential Quadratic Programming (SQP) [8] for optimal control require the reliable and efficient evaluation of such derivatives [9]. This is especially crucial in the context of economic objectives, where Gauss-Newton or other Hessian approximation strategies may fail to perform well [10].

Existing direct optimal control algorithms based on shooting methods either employ a *discretize-then-differentiate* [2] or a *differentiate-then-discretize* type of approach. The first technique carries out the differentiation after discretization, e.g., following the technique of Internal Numerical Differentiation (IND) [2] in combination with Algorithmic Differentiation (AD) [9] to evaluate these derivatives. Other discrete-time sensitivity propagation schemes are tailored for implicit integration methods [11, 12]. The *differentiate-then-discretize* approach involves an extension of the dynamic equations with their corresponding sensitivity equations, as implemented for example for first order sensitivities in SUNDIALS [13]. This can be performed for both forward [14, 15] and adjoint sensitivity analysis [16, 17]. Note that direct transcription methods [18] do not require such a propagation of sensitivities as in shooting based approaches, but they can still benefit from the proposed

symmetric evaluation of the Hessian contributions.

Unlike classical forward-over-adjoint (FOA) techniques [9, 19] to compute second order derivative information, the symmetric property when evaluating a Hessian matrix can be exploited as discussed in [9, 20] for explicit function evaluations. Following this research, a symmetric variant of AD was presented in [10] for explicit integration schemes in the context of exact Hessian based direct optimal control. Later in [21], this symmetric scheme has been extended to the case of an implicit integration method using the implicit function theorem in a discrete-time propagation technique. Similar to [10] and [21], the present article is concerned with the computation of Hessian matrices as required by Newton type optimization. The work by [22] instead proposes a *differentiate-then-discretize* type approach to compute Hessian vector products directly in the context of truncated Newton (TN) methods. In case of path-constrained optimal control problems, one could additionally use composite adjoints as discussed in [23].

1.1. Motivation and contributions

This paper proposes an efficient first and second order algorithmic differentiation scheme for both discrete- and continuous-time optimal control problems. Unlike the initial results from [10] and [21], this article presents and establishes the correctness of the symmetric Hessian propagation technique for any explicit or implicit integration method. The resulting second order sensitivity analysis typically allows for a computational speedup of about factor 2, by exploiting the symmetry of the Hessian. In addition, we present an extension of these results to continuous-time sensitivity propagation for an implicit system of Differential Algebraic Equations (DAE) of index 1. This discussion in a continuous-time framework allows for a generic sensitivity analysis, before applying a numerical discretization scheme.

Based on the symmetric sensitivity equations, a resulting three-sweep Hessian propagation (TSP) scheme has been proposed [10]. This technique is studied here both in discrete- and continuous-time, and shown to considerably reduce the memory requirements, in addition to the reduced computational burden, over classical forward-over-adjoint (FOA) approaches. The proposed TSP scheme avoids the need to store a trajectory of forward sensitivities based on the symmetric sensitivity equations, which generally results in a much smaller memory footprint to compute the Hessian. An implementation of these symmetric Hessian propagation techniques in the open-source

ACADO Toolkit software is presented and its performance is illustrated on the case study of a nonlinear biochemical reactor.

1.2. Notation and preliminaries

This paper denotes first order total and partial derivatives, respectively using the compact notation $\mathrm{D}_a F(a,b) = \frac{\mathrm{d}F(a,b)}{\mathrm{d}a}$ and $\partial_a F(a,b) = \frac{\partial F(a,b)}{\partial a}$. In addition, let us write the second order directional derivatives:

$$\langle c, \mathrm{D}^2_{a,b} F(a,b) \rangle = \sum_{k=1}^{n} c_k \frac{\mathrm{d}^2 F_k(a,b)}{\mathrm{d}a\,\mathrm{d}b}, \tag{1}$$

where $c \in \mathbb{R}^n$ is a constant vector and $F : \mathbb{R}^l \times \mathbb{R}^m \to \mathbb{R}^n$ a twice differentiable function. Notice that the map $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^{n \times l \times m} \to \mathbb{R}^{l \times m}$ does not denote a standard scalar product, since the first argument is a vector while the second argument is a tensor. We occasionally use the shorthand notation $\langle c, \mathrm{D}^2_{a,b} F \rangle$, if it is clear from the context that F depends on a and b. We write $\langle c, \mathrm{D}^2_a F(a) \rangle$ rather than $\langle c, \mathrm{D}^2_{a,a} F(a) \rangle$, in case F has only one argument. If the second order derivatives of F are continuous, the matrix $\langle c, \mathrm{D}^2_a F(a) \rangle$ is symmetric. When using partial instead of total derivatives, a similar compact notation for the directional second order derivatives is adopted:

$$\langle c, \partial^2_{a,b} F(a,b) \rangle = \sum_{k=1}^{n} c_k \frac{\partial^2 F_k(a,b)}{\partial a\,\partial b}. \tag{2}$$

The article is organized as follows. Section 2 introduces direct optimal control in a simplified setting in order to illustrate the need for efficient sensitivity analysis. Section 3 discusses discrete-time propagation techniques for a generic implicit integration method. We first present the classical first and second order techniques, then we propose and motivate our alternative symmetric propagation scheme. Section 4 then presents the continuous-time extension of these sensitivity equations, considering an implicit DAE system of index 1. A three-sweep Hessian propagation technique is introduced and discussed in Section 5, including implementation details. The open-source ACADO code generation software using these novel sensitivity propagation techniques is briefly discussed in Section 6. Numerical results for an illustrative case study are finally presented and discussed further in Section 7.

2. Problem Statement

Let us briefly introduce the problem formulation in which we are interested, including the system of Differential Algebraic Equations (DAE). We then introduce the framework of direct optimal control and Newton type algorithms to solve the resulting optimization problem, using first and second order sensitivity analysis.

2.1. Differential algebraic equations

Let us consider the following semi-explicit DAE system:

$$\begin{aligned} \dot{x}(t) &= f(x(t), z(t)), \quad x(0) = x_0(p), \\ 0 &= g(x(t), z(t)), \end{aligned} \tag{3}$$

in which $x(t) \in \mathbb{R}^{n_x}$ denotes the differential states, $z(t) \in \mathbb{R}^{n_z}$ the algebraic variables and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$, $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$. The parameters $p \in \mathbb{R}^{n_p}$ are additional variables which define the initial value function $x_0 : \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$. The latter DAE system is of index 1 [17] if the Jacobian $\partial_z g(\cdot)$ is non-singular. In case there are no algebraic variables, the system instead denotes a set of Ordinary Differential Equations (ODE):

$$\dot{x}(t) = f_{\text{ODE}}(x(t)), \quad x(0) = x_0(p). \tag{4}$$

We introduce the following two important assumptions.

Assumption 1. *The functions $f(x(t), z(t))$, $g(x(t), z(t))$ and $x_0(p)$ are twice continuously differentiable in all arguments.*

Assumption 2. *The semi-explicit DAE system in Eq. (3) has differential index 0 or index 1, which means that either $n_z = 0$, or the Jacobian matrix $\partial_z g(\cdot)$ must be invertible.*

Finally, let us refer to the following definition for consistent initial conditions such that the initial value problem in Eq. (3) has a unique solution $x(t, p)$ and $z(t, p)$ $\forall t \in [0, T], p$ given the previous two assumptions [14, 24].

Definition 3. *The initial values $(x(0, p), z(0, p))$ are called consistent when the following conditions hold:*

$$\begin{aligned} x(0, p) &= x_0(p) \\ 0 &= g(x(0, p), z(0, p)). \end{aligned} \tag{5}$$

This is a well defined nonlinear system in $(x(0, p), z(0, p))$ given the parameter values p and an index 1 DAE.

Remark 4. *The sensitivity propagation techniques in this article can be readily extended to the following implicit DAE formulation*

$$0 = f_{\mathrm{DAE}}(\dot{x}(t), x(t), z(t)), \quad x(0) = x_0(p), \tag{6}$$

which is of index 1 *when the Jacobian matrix* $\partial_{(\dot{x},z)} f_{\mathrm{DAE}}(\cdot)$ *is invertible [17]. To keep the notation relatively simple, let us however focus on the semi-explicit formulation from Eq.* (3) *instead.*

2.2. Optimal control problem formulation

This paper is concerned with derivatives for continuous-time optimal control problems (OCP). These could in the simplest case be of the form

$$\min_{x(\cdot),z(\cdot),p} \quad \phi\left(x(T)\right) \tag{7a}$$

$$\text{s.t.} \quad 0 = x(0) - x_0(p), \tag{7b}$$

$$\dot{x}(t) = f(x(t), z(t)), \quad \forall t \in [0, T], \tag{7c}$$

$$0 = g(x(t), z(t)), \quad \forall t \in [0, T], \tag{7d}$$

where the objective in Eq. (7a) consists of a terminal cost defined by the twice continuously differentiable function $\phi(\cdot)$, depending only on the differential states for notational convenience. For the ease of exposition, we do not introduce more general OCPs, which could comprise additional time-varying control inputs, inequality constraints on states and controls, or more general objective functionals.

2.3. Shooting parameterization

As previously defined, the unique solution of the initial value problem in Eq. (3) can be referred to as $x(t, p)$ and $z(t, p)$ $\forall t \in [0, T]$ and for a specific parameter value p. A single shooting [25] discretization of OCP (7) then results in the following unconstrained NLP:

$$\min_{p} \quad \Phi(p), \tag{8}$$

where $\Phi(p) = \phi\left(x(T, p)\right)$. A minimizer for this finite dimensional problem exists if and only if the continuous-time OCP (7) has an optimal, bounded solution. To preserve a more compact notation, single shooting will be used throughout this paper even though all presented techniques can also be employed within the direct multiple shooting method [2]. In practice, the function $x(T, p)$ is obtained approximately by numerical simulation of the system of differential algebraic equations [24].

2.4. Newton type optimization

Exact Newton methods of the form

$$p^{[k+1]} = p^{[k]} - \mathrm{D}_p^2\Phi(p^{[k]})^{-1}\,\mathrm{D}_p\Phi(p^{[k]})^\top, \quad \text{for } k = 0, 1, \ldots \tag{9}$$

converge locally quadratically to stationary points of problem (8) under mild conditions; see [6]. In general, one needs a well designed globalization strategy to guarantee convergence, but this topic is outside the scope of this paper. The first and second order derivatives in Eq. (9) are given by:

$$\begin{aligned} \mathrm{D}_p\Phi(p)^\top &= \mathrm{D}_p x(T,p)^\top \bar{\lambda}(p) \\ \mathrm{D}_p^2\Phi(p) &= \langle \bar{\lambda}(p), \mathrm{D}_p^2 x(T,p)\rangle + \mathrm{D}_p x(T,p)^\top \partial_x^2\phi\left(x(T,p)\right)\,\mathrm{D}_p x(T,p), \end{aligned} \tag{10}$$

where the notation $\bar{\lambda}(p)^\top = \partial_x\phi\left(x(T,p)\right)$ is used. Here, $\bar{\lambda}(p)$ is called the "backward seed" for the sensitivity propagation techniques. The evaluation of partial derivatives for the function $\phi(\cdot)$ is assumed to be relatively cheap using techniques of Algorithmic Differentiation [9]. The main computational effort is therefore typically the numerical simulation to evaluate $x(T,p)$ and the propagation of its first and second order sensitivities. Note that the directional second order derivatives $\langle \bar{\lambda}(p), \mathrm{D}_p^2 x(T,p)\rangle = \sum_{k=1}^{n_x} \bar{\lambda}_k(p)\mathrm{D}_p^2 x_k(T,\,p)$ are defined as in Eq. (1).

3. Discrete-Time Sensitivity Propagation

For a discussion on discrete-time propagation techniques for the sensitivities in (10), let us denote an integration scheme to simulate the index-1 DAE system in Eq. (3) by the following semi-explicit set of equations:

$$\begin{aligned} x_{n+1} &= F(x_n, z_n) \\ 0 &= G(x_n, z_n), \end{aligned} \tag{11}$$

for $n = 0, \ldots, N-1$ where $x_0 = x_0(p)$, the Jacobian matrix $\partial_z G(\cdot)$ is invertible and the functions $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$, $G : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ are twice continuously differentiable when Assumption 1 holds. In what follows, the dependency of the variables x_n, z_n on the parameter value p for $n = 0, \ldots, N-1$ will be omitted to arrive at a compact notation.

Note that the formulation in Eq. (11) includes both explicit and implicit integration methods [24]. The additional variables $z_n \in \mathbb{R}^{n_z}$ for $n = 0, \ldots, N-1$ denote all internal variables which are necessary to formulate

the integration scheme. In case of a collocation method, these variables for example denote the values of the differential state derivatives and algebraic variables at the collocation nodes [24]. The number of integration steps N is chosen to be fixed for notational convenience. The function $G(\cdot)$ would be empty in case of an explicit integration scheme.

3.1. First order sensitivity analysis

In what follows, we will refer to the function evaluations in Eq. (11) using the compact notation $F^n = F(x_n, z_n)$ and $G^n = G(x_n, z_n)$. Similarly, the first order derivatives read:

$$F_x^n = \partial_x F^n, \quad F_z^n = \partial_z F^n \quad \text{and} \quad G_x^n = \partial_x G^n, \quad G_z^n = \partial_z G^n, \tag{12}$$

of which the Jacobian matrix G_z^n is invertible.

Forward propagation. Let us define the sensitivities $S_n^x := \mathrm{D}_p x_n \in \mathbb{R}^{n_x \times n_p}$ and $S_n^z := \mathrm{D}_p z_n \in \mathbb{R}^{n_z \times n_p}$, which can be propagated forward using the following semi-explicit system of equations:

$$\begin{aligned} S_{n+1}^x &= F_x^n \, S_n^x + F_z^n \, S_n^z \\ 0 &= G_x^n \, S_n^x + G_z^n \, S_n^z, \end{aligned} \tag{13}$$

for $n = 0, \ldots, N-1$ where the initial value $S_0^x = \mathrm{D}_p x_0$ is given. These sensitivity equations (13) can be obtained directly by differentiating Eq. (11) with respect to p. Note that the matrix G_z^n is invertible such that $S_n^z = -G_z^{n^{-1}} G_x^n \, S_n^x$ could be computed explicitly using a matrix factorization. The end value S_N^x can be used to obtain the result $\mathrm{D}_p \Phi(p)^\top = \mathrm{D}_p(x_N)^\top \, \bar{\lambda} = S_N^{x^\top} \bar{\lambda}$ in the Newton type optimization scheme from Section 2.4.

Adjoint propagation. The sensitivity result $\mathrm{D}_p \Phi(p)$ can alternatively be computed directly by use of an adjoint propagation scheme where $\bar{\lambda}$ denotes the backward seed. For this purpose, let us define the adjoint variables $\lambda_n^x := \mathrm{D}_{x_n}(x_N)^\top \, \bar{\lambda} \in \mathbb{R}^{n_x}$ and $\lambda_n^z \in \mathbb{R}^{n_z}$ which can be propagated backward using the following semi-explicit system of equations:

$$\begin{aligned} \lambda_n^x &= F_x^{n^\top} \lambda_{n+1}^x + G_x^{n^\top} \lambda_{n+1}^z \\ 0 &= F_z^{n^\top} \lambda_{n+1}^x + G_z^{n^\top} \lambda_{n+1}^z, \end{aligned} \tag{14}$$

for $n = N-1, \ldots, 0$ where the initial value is given by the seed $\lambda_N^x = \bar{\lambda}$. This backward propagation scheme results in the sensitivity $\lambda_0^x = \mathrm{D}_{x_0}(x_N)^\top \, \bar{\lambda}$ such

that $\mathrm{D}_p\Phi(p)^\top = \mathrm{D}_p(x_0)^\top \lambda_0^x$ holds. These adjoint sensitivity equations (14) can be obtained directly by differentiating (11) with respect to x_n and multiplying the first equation with $\lambda_{n+1}^{x^\top}$:

$$\begin{aligned}\mathrm{D}_{x_n}(x_{n+1})^\top \lambda_{n+1}^x &= F_x^{n^\top} \lambda_{n+1}^x + \mathrm{D}_{x_n}(z_n)^\top F_z^{n^\top} \lambda_{n+1}^x \\ 0 &= G_x^{n^\top} + \mathrm{D}_{x_n}(z_n)^\top G_z^{n^\top},\end{aligned}$$

which defines $\mathrm{D}_{x_n}(z_n)^\top = -G_x^{n^\top} G_z^{n^{-\top}}$. By introducing $\lambda_{n+1}^z = -G_z^{n^{-\top}} F_z^{n^\top} \lambda_{n+1}^x$, one obtains the expressions in Eq. (14).

3.2. Second order sensitivity propagation

Next, we are interested in computing second order directional derivatives of the form $\langle \bar{\lambda}, \mathrm{D}_p^2 x_N \rangle = \langle \bar{\lambda}, \mathrm{D}_p^2 x(T,p) \rangle$ as required in Eq. (10) and following our notation in Eq. (1). Such directional second order derivatives can be computed by combining forward and backward techniques for first order sensitivity analysis. Combining the two techniques for first order derivatives results in four possible propagation schemes [9, 19]. However, in the *forward-over-forward* approach, computational effort would be spent in computing sensitivity directions that are not necessarily needed to form the Hessian result. Similarly, it is not efficient to perform more than one reverse sweep as discussed in [9]. In the following, among the two remaining approaches, preference will be given to the *forward-over-adjoint* (FOA) approach.

Let us introduce the following compact notation for the second order derivatives of the functions $F^n = F(x_n, z_n)$ and $G^n = G(x_n, z_n)$:

$$F_{a\,b}^n = \partial_{a,b}^2 F^n \quad \text{and} \quad G_{a\,b}^n = \partial_{a,b}^2 G^n. \tag{15}$$

Forward-over-adjoint (FOA) propagation. Let us apply forward differentiation directly to the adjoint propagation scheme in Eq. (14), where we also regard the dependency of the variables λ_n on the parameter p. This results in the additional variables $H_n^x \in \mathbb{R}^{n_x \times n_p}$ and $H_n^z \in \mathbb{R}^{n_z \times n_p}$, for which the corresponding FOA type equations read as:

$$\begin{aligned} H_n^x &= F_x^{n^\top} H_{n+1}^x + G_x^{n^\top} H_{n+1}^z \\ &\quad + \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix} \\ 0 &= F_z^{n^\top} H_{n+1}^x + G_z^{n^\top} H_{n+1}^z \\ &\quad + \begin{bmatrix} \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \end{aligned} \tag{16}$$

for $n = N-1, \dots, 0$ where the initial value is given by $H_N^x = \mathbb{0}$. The latter backward propagation results in the sensitivity $H_0^x = \langle \bar{\lambda}, \mathrm{D}^2_{x_0,p} x_N \rangle$ such that the Hessian matrix $\langle \bar{\lambda}, \mathrm{D}^2_p x_N \rangle = H_0^{x\top} \mathrm{D}_p x_0 + \langle \lambda_0^x, \mathrm{D}^2_p x_0(p) \rangle$ can be obtained. It is important to note that the FOA variables $H_n^x \in \mathbb{R}^{n_x \times n_p}$ and $H_n^z \in \mathbb{R}^{n_z \times n_p}$ are not symmetric or even square, and the same holds for the equations in the FOA type propagation scheme (16). The desired Hessian result $\langle \bar{\lambda}, \mathrm{D}^2_p x_N \rangle$ is however symmetric by definition.

3.3. Symmetric second order sensitivity propagation

We are here interested in an approach which can efficiently propagate a symmetric Hessian variable $H_n^{\mathrm{S}} \in \mathbb{R}^{n_p \times n_p}$ directly. In what follows, we show that such a symmetric propagation scheme provides multiple benefits over the classical FOA approach, while both provide the same second order sensitivities. The following theorem summarizes this result.

Theorem 5. *Let (x_{n+1}, z_n), (S_{n+1}^x, S_n^z) and $(\lambda_n^x, \lambda_{n+1}^z)$ be defined for $n = 0, \dots, N-1$ respectively by Eqs. (11), (13) and (14) and the corresponding initial and terminal values. The following propagation scheme then generates a symmetric variable H_n^{S}:*

$$H_{n+1}^{\mathrm{S}} = H_n^{\mathrm{S}} + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \tag{17}$$

for $n = 0, \dots, N-1$ starting from the initial value $H_0^{\mathrm{S}} = \langle \lambda_0^x, \mathrm{D}^2_p x_0(p) \rangle$. This symmetric variable satisfies $H_n^{\mathrm{S}} = \langle \lambda_n^x, \mathrm{D}^2_p x_n \rangle \in \mathbb{R}^{n_p \times n_p}$ and yields the desired Hessian result $H_N^{\mathrm{S}} = \langle \bar{\lambda}, \mathrm{D}^2_p x_N \rangle$.

Proof. The proof uses induction over n. For the case $n = 0$, the statement $H_0^{\mathrm{S}} = \langle \lambda_0^x, \mathrm{D}^2_p x_0 \rangle$ for the symmetric Hessian result holds by initialization. Let us assume that $H_n^{\mathrm{S}} = \langle \lambda_n^x, \mathrm{D}^2_p x_n \rangle$ holds for n. From the symmetric sequence in Eq. (17), the following then holds for the case $n+1$:

$$\begin{aligned} H_{n+1}^{\mathrm{S}} = {} & \langle \lambda_n^x, \mathrm{D}^2_p x_n \rangle \\ & + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}. \end{aligned} \tag{18}$$

From this expression, we can prove the desired result $H_{n+1}^{\mathrm{S}} = \langle \lambda_{n+1}^x, \mathrm{D}^2_p x_{n+1} \rangle$ based on the second order chain rule and the implicit function theorem. Using

equation $x_{n+1} = F(x_n, z_n)$ from (11), this second order chain rule reads:

$$\begin{aligned}\langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1} \rangle = \langle \tilde{\lambda}_n^x, \mathrm{D}_p^2 x_n \rangle + \langle \tilde{\lambda}_n^z, \mathrm{D}_p^2 z_n \rangle \\ + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix},\end{aligned} \tag{19}$$

where auxiliary variables $\tilde{\lambda}_n^{x^\top} := \lambda_{n+1}^{x^\top} F_x^n$ and $\tilde{\lambda}_n^{z^\top} := \lambda_{n+1}^{x^\top} F_z^n$ are defined. Let us recall the adjoint propagation from Eq. (14), where $0 = {F_z^n}^\top \lambda_{n+1}^x + {G_z^n}^\top \lambda_{n+1}^z$ and therefore $\lambda_{n+1}^{z^\top} = -\lambda_{n+1}^{x^\top} F_z^n {G_z^n}^{-1} = -\tilde{\lambda}_n^{z^\top} {G_z^n}^{-1}$ holds. The implicit function theorem for the equation $0 = G(x_n, z_n)$ then allows us to write the directional second order derivatives as:

$$\langle \tilde{\lambda}_n^z, \mathrm{D}_p^2 z_n \rangle = \langle \tilde{\mu}_n^x, \mathrm{D}_p^2 x_n \rangle + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \tag{20}$$

where additionally $\tilde{\mu}_n^{x^\top} := \lambda_{n+1}^{z^\top} G_x^n$ is defined and $\lambda_{n+1}^{z^\top} G_z^n = -\tilde{\lambda}_n^{z^\top}$ has been used. After combining the latter expression for $\langle \tilde{\lambda}_n^z, \mathrm{D}_p^2 z_n \rangle$ in (20) into the result from Eq. (19), one obtains:

$$\begin{aligned}\langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1} \rangle &= \langle \tilde{\lambda}_n^x, \mathrm{D}_p^2 x_n \rangle + \langle \tilde{\mu}_n^x, \mathrm{D}_p^2 x_n \rangle \\ &+ \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix} \\ &= \langle \lambda_n^x, \mathrm{D}_p^2 x_n \rangle \\ &+ \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix},\end{aligned} \tag{21}$$

where we used that $\lambda_n^x = {F_x^n}^\top \lambda_{n+1}^x + {G_x^n}^\top \lambda_{n+1}^z = \tilde{\lambda}_n^x + \tilde{\mu}_n^x$ from Eq. (14). This concludes the induction proof, because (21) shows that $H_{n+1}^{\mathrm{S}} = \langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1} \rangle$ holds, based on the original expression in Eq. (18). □

Let us briefly compare the classical FOA sensitivity propagation in Eq. (16) with the symmetric scheme in Eq. (17). One can observe that the novel equations propagate much less variables $H_n^{\mathrm{S}} \in \mathbb{R}^{n_p \times n_p}$ in case $n_p \ll (n_x + n_z)$, while the FOA variables are $H_n^x \in \mathbb{R}^{n_x \times n_p}$ and $H_n^z \in \mathbb{R}^{n_z \times n_p}$. In addition, the sensitivity equation (17) is symmetric, such that one can propagate only the lower triangular part of the variable $H_n^{\mathrm{S}} \in \mathbb{R}^{n_p \times n_p}$. Since this symmetric equation does not directly depend on H_n^{S} itself, the Hessian variable can be propagated in any direction. In Section 5, a three-sweep propagation scheme will be proposed in which these properties are exploited.

4. Continuous-Time Sensitivity Propagation

This section presents continuous-time sensitivity equations to propagate first and second order directional derivatives of the states with respect to the parameters p. We introduce the shorthand $f(t) = f(x(t), z(t))$ and $g(t) = g(x(t), z(t))$ for the DAE system in Eq. (3) whenever it is clear from the context at which point f and g are evaluated. Additionally, the partial derivatives of these functions are denoted by

$$f_x(t) = \partial_x f(t),\ f_z(t) = \partial_z f(t) \quad \text{and} \quad g_x(t) = \partial_x g(t),\ g_z(t) = \partial_z g(t).$$

The dependence of the simulation result $x(T, p)$ and the function $x_0(p)$ on the parameter value p is further omitted to allow a more compact notation.

Remark 6. *The continuous-time sensitivity equations could be derived from the discrete-time results in Section 3, by applying the limit for the discretization step size going to zero. For completeness, we however present these continuous-time results and instead provide a self-contained proof of correctness for the proposed symmetric Hessian propagation scheme.*

4.1. First order sensitivity equations

Forward propagation. Let us define the sensitivities $S^x(t) := \mathrm{D}_p x(t) \in \mathbb{R}^{n_x \times n_p}$ and $S^z(t) := \mathrm{D}_p z(t) \in \mathbb{R}^{n_z \times n_p}$. The forward system of sensitivity equations [14, 15] corresponding to the DAE in (3) can be written as:

$$\begin{aligned} \dot{S}^x(t) &= f_x(t) S^x(t) + f_z(t) S^z(t), \quad \text{with} \quad S^x(0) = \mathrm{D}_p x_0 \\ 0 &= g_x(t) S^x(t) + g_z(t) S^z(t), \end{aligned} \tag{22}$$

which is also of index 0 or 1 under Assumption 2. The end value $S^x(T)$ can be used to obtain the result $\mathrm{D}_p \Phi(p)^\top = \mathrm{D}_p(x_T)^\top \bar{\lambda} = S^x(T)^\top \bar{\lambda}$ in the Newton type optimization scheme.

Adjoint propagation. The sensitivity result $\mathrm{D}_p \Phi(p)$ can alternatively be computed directly by use of an adjoint propagation scheme. For this purpose, let us define the adjoint variables $\lambda^x(t) := \mathrm{D}_{x(t)} x(T)^\top \bar{\lambda} \in \mathbb{R}^{n_x}$ and $\lambda^z(t) \in \mathbb{R}^{n_z}$. As derived in detail by [16, 17], the adjoint system then reads as:

$$\begin{aligned} -\dot{\lambda}^x(t) &= f_x(t)^\top \lambda^x(t) + g_x(t)^\top \lambda^z(t) \\ 0 &= f_z(t)^\top \lambda^x(t) + g_z(t)^\top \lambda^z(t), \end{aligned} \tag{23}$$

which is again of index 0 or 1 and the initial value $\lambda^x(T) = \bar{\lambda}$ is the backward seed. The adjoint result $\mathrm{D}_p(x_T)^\top \bar{\lambda} = \mathrm{D}_p(x_0)^\top \lambda^x(0)$ is then obtained directly by this backward sensitivity propagation.

4.2. Second order sensitivity equations

Next, we are interested in computing second order directional derivatives of the form $\langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle$ as required in Eq. (10) and following our notation in Eq. (1). Let us start by presenting the classical *forward-over-adjoint* (FOA) type sensitivity equations in continuous-time.

Forward-over-adjoint (FOA). Applying forward differentiation directly to the adjoint system (23) yields $H^x(t) := \mathrm{D}_p \lambda^x(t) \in \mathbb{R}^{n_x \times n_p}$ and $H^z(t) := \mathrm{D}_p \lambda^z(t) \in \mathbb{R}^{n_z \times n_p}$. These variables are described by the *FOA system* [19]:

$$\begin{aligned} -\dot{H}^x(t) &= f_x(t)^\top H^x(t) + g_x(t)^\top H^z(t) \\ &\quad + \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle + \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle + \langle \lambda^z, g_{x\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix} \\ 0 &= f_z(t)^\top H^x(t) + g_z(t)^\top H^z(t) \\ &\quad + \begin{bmatrix} \langle \lambda^x, f_{z\,x} \rangle + \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle + \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}, \end{aligned} \tag{24}$$

where the first order sensitivities are defined earlier and the second order derivatives read $f_{a\,b} = \partial^2_{a,b} f(t)$ and $g_{a\,b} = \partial^2_{a,b} g(t)$. The initial value needs to be chosen $H^x(T) = \mathbb{0}$ and consistent values for $H^z(T)$ can be obtained following Definition 3. The Hessian result can then be obtained as $\langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle = \mathrm{D}_p(x_0)^\top H^x(0) + \langle \lambda^x(0), \mathrm{D}_p^2 x_0(p) \rangle$. It is interesting to note that the sensitivity equation (24) is not symmetric, unlike the resulting Hessian.

4.3. Symmetric second order sensitivity equations

Instead of the FOA system in Eq. (24), we would like to directly propagate a symmetric variable $H^{\mathrm{S}}(t) \in \mathbb{R}^{n_p \times n_p}$. We show that both approaches can provide the same second order sensitivities $\langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle$.

Theorem 7. *Let $(x(t), z(t))$, $(S^x(t), S^z(t))$ and $(\lambda^x(t), \lambda^z(t))$ be defined for $t \in [0, T]$ respectively by Eqs. (3), (22) and (23) and the corresponding consistent initial values. Let us introduce the following symmetric sensitivity equations to propagate $H^{\mathrm{S}}(t)$:*

$$\dot{H}^{\mathrm{S}}(t) = \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle + \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle + \langle \lambda^z, g_{x\,z} \rangle \\ \langle \lambda^x, f_{z\,x} \rangle + \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle + \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}, \tag{25}$$

for $t \in [0, T]$, starting from the initial value $H^{\mathrm{S}}(0) = \langle \lambda^x(0), \mathrm{D}_p^2 x_0(p) \rangle$. The symmetric variable satisfies $H^{\mathrm{S}}(t) = \langle \lambda^x(t), \mathrm{D}_p^2 x(t) \rangle$ and provides the desired Hessian result $H^{\mathrm{S}}(T) = \langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle$.

Proof. The expression $H^{\mathrm{S}}(t) = \langle \lambda^x(t), \mathrm{D}_p^2 x(t) \rangle$ is satisfied for $t = 0$ because of the initialization $H^{\mathrm{S}}(0) = \langle \lambda^x(0), \mathrm{D}_p^2 x_0(p) \rangle$. It is sufficient to show that the time derivative of this expression satisfies the differential equation system (25), in order to obtain the desired Hessian result. We start by differentiating $H^{\mathrm{S}}(t) = \langle \lambda^x(t), \mathrm{D}_p^2 x(t) \rangle$ with respect to time:

$$\dot{H}^{\mathrm{S}}(t) = \langle \dot{\lambda}^x(t), \mathrm{D}_p^2 x(t) \rangle + \langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle. \tag{26}$$

The second order chain rule for the derivative $\langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle$ reads as:

$$\langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle = -\langle \dot{\lambda}^x(t), \mathrm{D}_p^2 x(t) \rangle + S^x(t)^\top \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle S^x(t), \tag{27}$$

where $\lambda^x(t)^\top \mathrm{D}_{x(t)} \dot{x}(t) = -\dot{\lambda}^x(t)^\top$ has been used and the directional derivative is defined as $\langle \lambda^x, \mathrm{D}_p^2 \dot{x} \rangle = \sum_{k=1}^{n_x} \lambda_k^x \ \mathrm{D}_p^2 \dot{x}_k$ following our notation in Eq. (1). The latter expression (27) can be used to simplify Eq. (26):

$$\begin{aligned} \dot{H}^{\mathrm{S}}(t) &= \langle \dot{\lambda}^x(t), \mathrm{D}_p^2 x(t) \rangle + \langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle \\ &= S^x(t)^\top \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle S^x(t). \end{aligned} \tag{28}$$

By differentiating the differential equation $\dot{x}(t) = f(x(t), z(t))$ twice with respect to x, we can write the directional second order derivative:

$$\begin{aligned} \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle = &\begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle \\ \langle \lambda^x, f_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix} \\ &+ \langle \tilde{\lambda}^z(t), \mathrm{D}_x^2 z(t) \rangle, \end{aligned} \tag{29}$$

where $\tilde{\lambda}^z(t)^\top := \lambda^x(t)^\top f_z(t)$ is defined. Let us compute the remaining directional derivative $\langle \tilde{\lambda}^z(t), \mathrm{D}_x^2 z(t) \rangle$ by also differentiating the algebraic equation $0 = g(x(t), z(t))$ twice with respect to x:

$$0 = \begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^z, g_{x\,z} \rangle \\ \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix} - \langle \tilde{\lambda}^z(t), \mathrm{D}_x^2 z(t) \rangle, \tag{30}$$

where we used that $\tilde{\lambda}^z(t)^\top = \lambda^x(t)^\top f_z(t) = -\lambda^z(t)^\top g_z(t)$, based on the second expression in the adjoint system (23). Combining these expressions from Eqs. (30) and (29) into Eq. (28), yields the differential equation:

$$\begin{aligned} \dot{H}^{\mathrm{S}}(t) &= S^x(t)^\top \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle S^x(t) \\ &= \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle + \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle + \langle \lambda^z, g_{x\,z} \rangle \\ \langle \lambda^x, f_{z\,x} \rangle + \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle + \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}, \end{aligned} \tag{31}$$

where we used $S^z(t) = \mathrm{D}_{x(t)} z(t)\, S^x(t)$ and which corresponds to the desired symmetric sensitivity equation (25), concluding our proof. □

The symmetry of System (25) can be exploited by propagating its lower triangular part only. Notice also that unlike the classical FOA approach, the symmetric scheme results in an ODE system describing the propagation of the Hessian result $H^{\mathrm{S}}(t)$, although the original system is in general an implicit DAE system of index 1. In addition, the time direction in which these equations are simulated is arbitrary and can therefore be reversed. A very similar theorem and proof could be constructed for a backward in time propagation of the symmetric system.

4.4. Continuous versus discrete-time sensitivity propagation

Even though continuous-time sensitivity equations are generally very useful because they can be simulated using any integration scheme to obtain a numerical approximation of the continuous-time sensitivity result, one often prefers discrete-time numerical differentiation in the case of direct optimal control methods [2]. The nonlinear program in Eq. (8) is defined using the numerical simulation of the DAE system (3), such that the optimization algorithm requires the derivatives of this numerical approximation instead of the continuous-time sensitivities. In general, the discrete- and continuous-time sensitivities are equal up to terms of order $O(h^s)$, where h denotes the discretization step size and s is the order of the numerical integration scheme.

When applying Explicit Runge-Kutta (ERK) methods to ODE systems, the connection between the discrete-time and continuous-time sensitivity propagation is discussed in detail by [26, 27]. Forward mode of AD applied to any linear multistep integration method, including single-step schemes like Runge-Kutta (RK), provides derivatives which are equivalent to those obtained by applying the same integration method directly to the system of forward sensitivity equations [26, 28, 29]. In case of adjoint sensitivity analysis, a discrepancy between the state and costate discretizations leads to additional conditions that the coefficients of the integration scheme must satisfy to achieve the same order of accuracy [30].

5. Forward-Backward versus Three-Sweep Hessian Propagation

A stored trajectory for the state variables $x(t), z(t)$ from (3) for $t \in [0, T]$ is needed to simulate the adjoint equations and similar requirements hold for

the second order sensitivity equations. This section presents an alternative approach for second order sensitivity propagation, based on the symmetric right hand of Eq. (25). Even though the algorithm description will be based on the continuous-time sensitivity equations from Section 4, the same techniques can be applied to the discrete-time propagation schemes from Section 3 based on the symmetric matrix equations in (17).

The classical approach for second order sensitivity propagation performs a forward sweep, followed by a backward sweep. We will therefore refer to this as the forward-backward (FB) propagation technique. As illustrated in Algorithm 1, the trajectories of $x(t), z(t)$ and $S^x(t), S^z(t)$ from the forward simulation over $t \in [0, T]$ need to be stored. In the following backward sweep, the adjoint derivatives $\lambda^x(t), \lambda^z(t)$ as well as the second order derivatives are propagated. The latter can be based either on the FOA equations from (24) using $H^x(t), H^z(t)$ or based on the proposed symmetric scheme in (25) which propagates directly the symmetric variable $H^{\mathrm{S}}(t)$.

Algorithm 1: Forward-backward Hessian propagation

Input: The initial value $x_0(p)$.

$\longrightarrow$ Propagate and store $x(t), z(t)$ in (3) and $S^x(t), S^z(t)$ in (22).
Evaluate backward seed $\bar{\lambda}^\top = \partial_x \phi\left(x(T,p)\right)$ in Eq. (10).

$\longleftarrow$ Propagate backward $\lambda^x(t), \lambda^z(t)$ in Eq. (23) and $H^x(t), H^z(t)$ in Eq. (24) or $H^{\mathrm{S}}(t)$ in Eq. (25).

Output: The results for $x(T,p)$, $\mathrm{D}_p x(T,p)^\top \bar{\lambda}$ and $\langle \bar{\lambda}, \mathrm{D}_p^2 x(T,p) \rangle$.

As mentioned earlier, the symmetric equations represent a plain summation independent of the current value of $H^{\mathrm{S}}(\cdot)$ such that the time direction in which they are simulated can be reversed. This yields an alternative to the forward-backward propagation, which consists of three consecutive sweeps. Algorithm 2 illustrates this three-sweep propagation (TSP) technique [10]. Note that it can be computationally better to perform less sweeps in order to allow more reuse of expressions in the derivative evaluations [9], even though the TSP will show other advantages over the FB propagation. The main advantage is that storage of the full forward trajectory of $S^x(t), S^z(t)$ can be avoided, since these first order derivatives are propagated together with the symmetric Hessian results in the third sweep.

Algorithm 2: Three-sweep Hessian propagation (TSP) for symmetric scheme

Input: The initial value $x_0(p)$.

$\longrightarrow$ Propagate forward and store $x(t), z(t)$ in Eq. (3).
Evaluate backward seed $\bar{\lambda}^\top = \partial_x \phi\left(x(T,p)\right)$ in Eq. (10).
$\longleftarrow$ Propagate backward and store $\lambda^x(t), \lambda^z(t)$ in Eq. (23).
$\longrightarrow$ Propagate forward $S^x(t), S^z(t)$ in Eq. (22) and $H^{\mathrm{S}}(t)$ in Eq. (25).

Output: The results for $x(T,p)$, $\mathrm{D}_p x(T,p)^\top \bar{\lambda}$ and $\langle \bar{\lambda}, \mathrm{D}_p^2 x(T,p) \rangle$.

5.1. Implementation details: TSP versus FB scheme

Table 1 shows a comparison between the FOA and the symmetric sensitivity equations. Note that the table presents the dimensions of the propagated matrix variables in both schemes, which does not directly correspond to the computational complexity. The latter depends on the efficient evaluation of the derivatives, on which more information can be found in [9]. In summary, the symmetric Hessian propagation from Eq. (25) in continuous time or Eq. (17) in discrete time provides the following advantages over the classical FOA scheme respectively from Eqs. (24) or (16):

- The matrix valued right hand in Eq. (25) or (17) is symmetric, i.e., only the lower or upper triangular part needs to be propagated. In the case where sensitivities are needed with respect to all states, which is common in direct optimal control, Table 1 shows that $n_x(n_x+1)/2$ equations need to be propagated instead of n_x^2. This could result in a speedup of about factor 2, depending on the sparsity of the problem.

- In case of implicit systems, the symmetric scheme is explicit and needs no additional variables. This is in contrast to the FOA scheme.

- The derivatives in (25) or (17) can be evaluated using a symmetric AD technique for factorable functions as discussed in [10, 20].

- Since the time direction of the symmetric equations can be reversed, one can avoid the storage of a trajectory of forward sensitivities based on the TSP scheme. Unlike the technique of checkpointing [9], this typically causes a negligible amount of extra computational effort in the case of explicit differential equations [10].

Table 1: Theoretical cost comparison of second order sensitivity propagation.

Discrete-time	D-FOA eqs. (16)	D-SYM eqs. (17)
dimension (#variables)	$S_n^x, S_n^z, H_n^x, H_n^z$: $2\,n_x n_p + 2\,n_z n_p$	$S_n^x, S_n^z, H_n^{\mathrm{S}}$: $n_x n_p + n_z n_p + n_p(n_p+1)/2$
Continuous-time	C-FOA eqs. (24)	C-SYM eqs. (25)
dimension (#variables)	$S^x(t), S^z(t)$, $H^x(t), H^z(t)$: $2\,n_x n_p + 2\,n_z n_p$	$S^x(t), S^z(t), H^{\mathrm{S}}(t)$: $n_x n_p + n_z n_p + n_p(n_p+1)/2$

Table 2 illustrates the storage costs for the FB and the TSP scheme respectively in Algorithm 1 and 2. The latter table shows that the TSP scheme can considerably reduce the memory requirements by propagating the first order forward sensitivities together with the symmetric Hessian propagation in a separate third sweep. A detailed comparison of these differentiation schemes would need to include additional factors such as the used integration method, the evaluation of derivatives in the sensitivity equations and the sparsity of the system dynamics as discussed in [9].

The presented sensitivity propagation techniques rely on the ability to simulate a certain system of differential equations both forward and backward in time. Otherwise, one could employ a time reversing transformation as discussed in [16]. An efficient implementation of first order techniques can be found, e.g., in [13], which employs checkpointing to reduce storage requirements for adjoint sensitivity analysis. This approach could be applied in a similar way to the second order propagation schemes in this paper.

5.2. The TSP scheme for implicit equations

Based on the discussion in [21], it should be noted that the advantages of the TSP scheme regarding its storage requirements can become less apparent or even disappear in case of implicit integration methods. To illustrate this, let us look at the integration scheme in Eq. (11) and its first order sensitivity propagation in Eq. (13) and (14). A Newton type method to solve the implicit equation $0 = G(x_n, z_n)$ in (11) requires a factorization of the Jacobian G_z^n, which is also needed for the sensitivity equations (13) and (14). One needs

Table 2: The storage cost for the second order sensitivity propagation techniques.

Discrete-time	Forward-backward (D-FB)	Three-sweep (D-TSP)
trajectory	x_n, z_n, S^x_n, S^z_n:	$x_n, z_n, \lambda^x_n, \lambda^z_n$:
storage	$(n_x + n_z)(1 + n_p)$	$2\,(n_x + n_z)$
Continuous-time	Forward-backward (C-FB)	Three-sweep (C-TSP)
trajectory	$x(t), z(t), S^x(t), S^z(t)$:	$x(t), z(t), \lambda^x(t), \lambda^z(t)$:
storage	$(n_x + n_z)(1 + n_p)$	$2\,(n_x + n_z)$

to either store these Jacobian factorizations in the first sweep of Algorithm 1 and 2, or one needs to recompute them. It is possible that the advantage of using the TSP scheme over the classical forward-backward propagation, regarding its storage costs, becomes relatively small in this case. It therefore depends on the specific implementation, in the case of sensitivity propagation for implicit differential equations or implicit integration methods, whether the FB or the TSP scheme should be used. Nonetheless, the proposed symmetric sensitivity equations can still be used within the forward-backward propagation for its computational advantages as listed before.

6. ACADO Toolkit Software and Implementation Details

An efficient implementation of the presented first and second order sensitivity propagation techniques for general DAE systems as well as of the symmetric AD method [10], can be found in the open-source ACADO Toolkit software [31]. It supports many algorithmic features for nonlinear optimal control, including the Real-Time Iteration (RTI) scheme for Nonlinear MPC (NMPC) [32]. The RTI algorithm is based on Sequential Quadratic Programming (SQP) to solve the nonlinear optimization problem within direct multiple shooting [2]. The software is free of charge and can be downloaded from `www.acadotoolkit.org`. The ACADO code generation tool can be used to obtain real-time feasible code for dynamic optimization on embedded control hardware. It specifically pursues the export of efficient C-code based on the RTI scheme for NMPC [33]. A user friendly MATLAB interface is available to use the ACADO Toolkit and its code generation tool, without direct interaction with C/C++ programming [12, 31].

Table 3: The proposed second order sensitivity propagation techniques.

	Discrete-time	Continuous-time
FB	D-FB-FOA (16) D-FB-SYM (17)	C-FB-FOA (24) C-FB-SYM (25)
TSP	D-TSP-SYM (17)	C-TSP-SYM (25)

Unlike the discrete-time discussion in [10] and [21] for specific integration schemes, this article presented a generalization of the symmetric sensitivity equations to implicit integration methods and to continuous-time implicit DAE systems of index up to 1. Table 3 illustrates the presented options for Hessian propagation, combining the classical or symmetric sensitivity equations, with a FB or a TSP scheme within a discrete- or a continuous-time framework. Following the approach of Internal Numerical Differentiation (IND) [2], the open-source implementation in the ACADO Toolkit is based on a discrete-time sensitivity propagation, i.e., corresponding to the schemes in the first column of Table 3. Note that the presented implementation targets mostly small to medium-scale systems of about 10-100 states [12, 31], even though this does not limit the applicability of the proposed algorithmic techniques to such problems.

As an example, let us briefly look at the ODE model equations for a chain of n_m masses as described in [34]. Table 4 presents the timings results for a numerical simulation over 0.5 s using 10 steps of the explicit Runge-Kutta (RK) method of order 4, based on the D-TSP-SYM or the D-FB-FOA scheme. The table shows the computation times for a second order sensitivity analysis with respect to both the states and control variables ($n_p = n_x + n_u$) or only the controls ($n_p = n_u = 3$), using different numbers of masses n_m with the corresponding state dimension $n_x = 6\,(n_\mathrm{m} - 1)$. It can be observed from this table that the overall computational speedup is about factor 2 based on the symmetry of the proposed sensitivity equations in (17). The following section finally illustrates these results on the illustrative case study of the economically optimal control of a nonlinear biochemical reactor.

Table 4: Computation times for a numerical simulation of the chain mass example [34] using explicit RK4: three-sweep versus forward-backward sensitivity propagation.

		D-TSP-SYM		D-FB-FOA	
n_{m}	n_x	$n_p = n_x + n_u$	$n_p = n_u$	$n_p = n_x + n_u$	$n_p = n_u$
3	12	133 µs	29 µs	326 µs	57 µs
4	18	376 µs	66 µs	979 µs	125 µs
5	24	826 µs	103 µs	1945 µs	188 µs
6	30	1523 µs	139 µs	3219 µs	245 µs
7	36	2495 µs	176 µs	4970 µs	300 µs
8	42	3710 µs	211 µs	6902 µs	386 µs
9	48	5293 µs	255 µs	9650 µs	461 µs

7. Case Study: Economic Optimal Control of a Bioreactor

The numerical results in this section have been obtained using the open-source ACADO code generation tool on a standard computer, equipped with Intel i7-3720QM processor, and using a 64-bit version of Ubuntu 14.04 and the g++ compiler version 4.8.4. The code to reconstruct the presented numerical results can be found on the following public repository: `https://github.com/rienq/symmetricHessians`.

7.1. Modeling the system

Let us consider the following explicit ODE model of a continuous bioreactor for culture fermentation [35, 36, 37]:

$$\dot{x} = f_{\mathrm{reactor}}(x,u) = \begin{pmatrix} -DX_{\mathrm{b}} + \mu(x)X_{\mathrm{b}} \\ D(U_{\mathrm{f}} - X_{\mathrm{s}}) - \frac{\mu(x)X_{\mathrm{b}}}{Y_{\mathrm{b}}} \\ -DX_{\mathrm{p}} + (\alpha\mu(x) + \beta)X_{\mathrm{b}} \\ X_{\mathrm{b}}/T \\ U_{\mathrm{f}}/T \\ DX_{\mathrm{p}}/T \end{pmatrix}. \tag{32}$$

Here, the state vector $x = (X_{\mathrm{b}}, X_{\mathrm{s}}, X_{\mathrm{p}}, q_{\mathrm{b}}, q_{\mathrm{f}}, q_{\mathrm{p}})$ consists of three physical states: the biomass X_{b}, the substrate X_{s} and the product concentration X_{p} as well as three auxiliary states q_{b}, q_{f} and q_{p}. The latter auxiliary variables are also known as quadrature states [38], e.g., $\dot{q}_{\mathrm{b}}(t) = X_{\mathrm{b}}(t)/T$ such that $q_{\mathrm{b}}(T) = \frac{1}{T}\int_0^T X_{\mathrm{b}}(\tau)\mathrm{d}\tau$ denotes the average biomass concentration. These quadrature

Table 5: Parameter values and bounds for the bioreactor.

Name	Symbol	Value
dilution rate	D	$0.15\,\mathrm{h}^{-1}$
substrate inhibition constant	K_i	$22\,\mathrm{g/L}$
substrate saturation constant	K_m	$1.2\,\mathrm{g/L}$
product saturation constant	P_m	$50\,\mathrm{g/L}$
yield of the biomass	Y_b	0.4
first product yield constant	α	2.2
second product yield constant	β	$0.2\,\mathrm{h}^{-1}$
specific growth rate scale	μ_m	$0.48\,\mathrm{h}^{-1}$
maximum feed substrate	$\overline{U}$	$40.0\,\mathrm{g/L}$
minimum feed substrate	$\underline{U}$	$28.7\,\mathrm{g/L}$
maximum average feed substrate	$\overline{U}^\mathrm{A}$	$32.9\,\mathrm{g/L}$
maximum average biomass concentration	$\overline{X}_\mathrm{b}^\mathrm{A}$	$5.8\,\mathrm{g/L}$

states will be used further to formulate the objective and constraint functions in the OCP. The control input $u = U_\mathrm{f}$ of the system denotes the feed substrate concentration which is bounded $\underline{U} \leq U_\mathrm{f} \leq \overline{U}$. The specific growth rate $\mu(x)$ is given by the expression

$$\mu(x) = \mu_\mathrm{m} \frac{(1 - \frac{X_\mathrm{p}}{P_\mathrm{m}} X_\mathrm{s})}{K_\mathrm{m} + X_\mathrm{s} + \frac{{X_\mathrm{s}}^2}{K_\mathrm{i}}} \ .$$

The remaining parameter values and operating bounds are summarized in Table 5. The bioreactor can be modeled by an explicit ODE as in (32), but we later also refer to the following semi-explicit DAE formulation:

$$\begin{aligned} \dot{x} &= f_\mathrm{reactor}(x, \mu, u) \\ 0 &= \mu - \mu_\mathrm{m} \frac{(1 - \frac{X_\mathrm{p}}{P_\mathrm{m}} X_\mathrm{s})}{K_\mathrm{m} + X_\mathrm{s} + \frac{{X_\mathrm{s}}^2}{K_\mathrm{i}}}, \end{aligned} \tag{33}$$

where the specific growth rate has been introduced as an algebraic variable. Even though there is no clear advantage of using the latter DAE formulation in this case, it can be used to illustrate the proposed sensitivity propagation schemes for implicit systems of equations.

7.2. Optimal control problem formulation

Similar to the OCP formulation in [10], our aim here is to maximize the average productivity which corresponds to the Mayer term $q_{\mathrm{p}}(T)$ as defined in Eq. (32). The end time T of a single cycle is assumed to be fixed and equal to 48 hours. The resulting continuous-time OCP reads as:

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & -q_{\mathrm{p}}(T) && && (34a)\\
\text{s.t.} \quad 0 &= x(0) - \bar{x}_0, && && (34b)\\
\dot{x}(t) &= f_{\mathrm{reactor}}(x,u), && \forall t \in [0,T], && (34c)\\
\underline{U} &\leq U_{\mathrm{f}} \leq \overline{U}, && && (34d)\\
q_{\mathrm{f}}(T) &\leq \overline{U}^{\mathrm{A}}, && && (34e)\\
q_{\mathrm{b}}(T) &\leq \overline{X}_{\mathrm{b}}^{\mathrm{A}}, && && (34f)\\
X_{\mathrm{b}}(T) &= X_{\mathrm{b}}(0), && && (34g)\\
X_{\mathrm{s}}(T) &= X_{\mathrm{s}}(0), && && (34h)\\
X_{\mathrm{p}}(T) &= X_{\mathrm{p}}(0), && && (34i)
\end{aligned}$$

which, in addition to the initial value condition (34b) and the dynamics (34c), also includes the control bounds (34d) and an upper bound on the average concentration for feed substrate (34e) and biomass (34f). Following the problem formulation in [10, 37], periodicity constraints on the three physical states X_{b}, X_{s} and X_{p} are included in Eqs. (34g)-(34i). Note that the initial value $\bar{x}_0$ for the states is considered constant, unlike the OCP formulation from Eq. (7). Instead, the sensitivity information with respect to the control inputs will be needed in a Newton type optimization method for the above OCP after performing a shooting discretization [2]. For the sake of simplicity, we consider a piecewise constant control parametrization over the horizon of $N = 20$ equidistant intervals. The solution trajectories for the states and controls are shown in Figure 1.

7.3. Numerical results

As discussed in Section 6, the open-source ACADO Toolkit can be used to efficiently solve the OCP in Eq. (34), based on direct multiple shooting and an SQP type algorithm. The presented sensitivity propagation techniques will be illustrated in discrete-time, using 5 integration steps of the form in Eq. (11) within each of the $N = 20$ shooting intervals over the control horizon

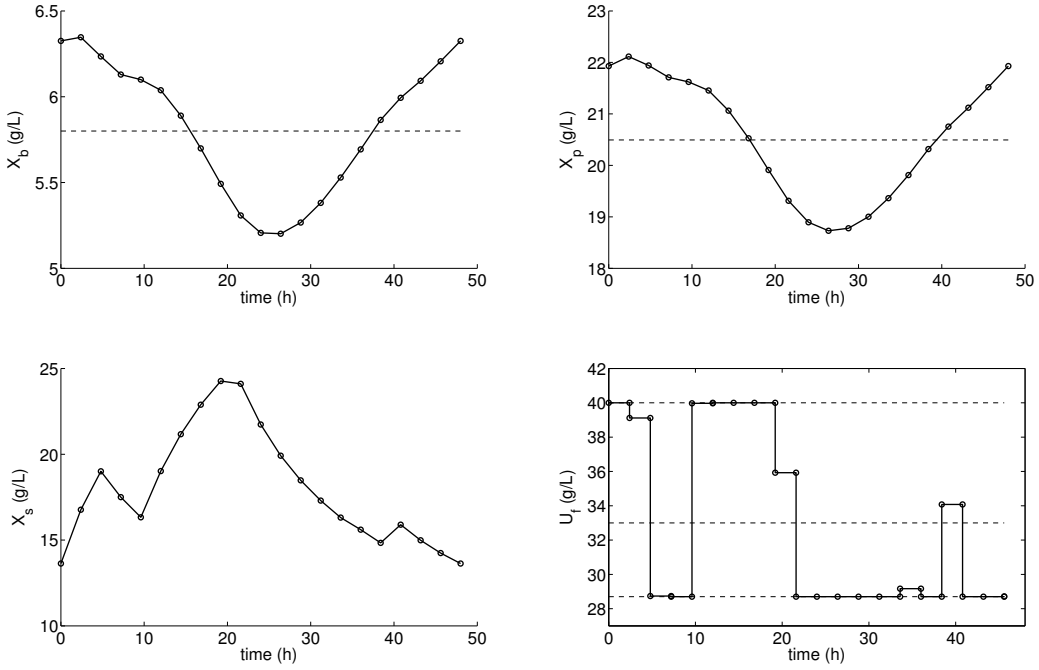

Figure 1: This figure presents the optimal state and control trajectories, corresponding to the periodic OCP for the nonlinear bioreactor in Eq. (34).

Table 6: Detailed computation times for exact Hessian based SQP using the explicit RK method: three-sweep versus forward-backward sensitivity propagation.

	D-TSP-SYM		D-FB-FOA	
Forward sweep 1	15 µs	6 %	95 µs	24 %
Backward sweep 2	18 µs	7 %	172 µs	44 %
Forward sweep 3	99 µs	39 %	-	-
Total simulation	135 µs	53 %	272 µs	69 %
Condensing	17 µs	7 %	19 µs	5 %
Regularization	47 µs	18 %	47 µs	12 %
QP solution	56 µs	22 %	56 µs	14 %
Total SQP step	256 µs		394 µs	

of $T = 48$ hours. Let us use the explicit Runge-Kutta (RK) method of order 4 for the ODE model in Eq. (32). Respectively, we use an implicit RK method of order 4 to simulate the semi-explicit DAE formulation from Eq. (33). Note that the latter DAE system is equivalent to the original ODE model and has been introduced mainly to illustrate the sensitivity propagation techniques in the presence of implicit algebraic equations.

Table 6 details the average computation times for the different components during one iteration of the SQP algorithm, using the explicit RK method for the ODE model. The table includes a comparison of the proposed symmetric three-sweep Hessian propagation (TSP-SYM) versus the classical forward-over-adjoint based FB scheme (FB-FOA). The symmetric propagation algorithm indeed yields a speedup of about factor 2 as discussed in Section 5.1, when comparing the total simulation time. Moreover, recall from Table 2 that the TSP scheme propagates all forward and second order sensitivities in a separate third sweep and is therefore also more memory efficient than the classical FB approach. Table 7 shows the average computation times using the implicit RK method for the DAE system, based on either the symmetric or the FOA equations for the forward-backward Hessian propagation. As mentioned earlier, the linear algebra routines for computing the Jacobian factorization often dominate the computational effort within an implicit scheme. This can also be observed in this table, since the total simulation times are relatively similar for both techniques in this case.

Table 7: Average computation times for exact Hessian based SQP using the implicit RK method: symmetric versus FOA equations within forward-backward propagation.

	D-FB-SYM		D-FB-FOA	
Total simulation	414 µs	74 %	451 µs	76 %
Total SQP step	557 µs		594 µs	

8. Conclusion

In this paper, we presented a novel sensitivity propagation technique to compute Hessian contributions as needed for Newton type optimization in direct optimal control. By maintaining and exploiting the symmetry, we proposed a scheme which allows a computational speedup of about factor 2 over the classical forward-over-adjoint approach for both discrete- and continuous-time sensitivity analysis. The techniques have been presented in a general framework, including implicit integration methods and DAE systems of index up to 1. A novel three-sweep propagation technique has been proposed using this set of symmetric sensitivity equations, which results in a reduced memory footprint by avoiding the trajectory of forward sensitivities to be stored. Based on an open-source implementation in the ACADO Toolkit [33], the performance has been illustrated for the economic optimal control of a nonlinear biochemical reactor.

References

[1] H. Bock, T. Lohmann, J. Schlöder, Numerical methods for parameter estimation and optimal experimental design in chemical reaction systems, Industrial and Engineering Chemistry Research 31 (1992) 54–57.

[2] H. G. Bock, Recent advances in parameter identification techniques for ODE, in: Numerical Treatment of Inverse Problems in Differential and Integral Equations, Birkhäuser, 1983, pp. 95–121.

[3] L. Biegler, Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation, Computers and Chemical Engineering 8 (1984) 243–248.

[4] J. Rawlings, D. Mayne, Model Predictive Control: Theory and Design, Nob Hill, 2009.

[5] M. Diehl, H. J. Ferreau, N. Haverbeke, Nonlinear model predictive control, Vol. 384 of Lecture Notes in Control and Information Sciences, Springer, 2009, Ch. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pp. 391–417.

[6] J. Nocedal, S. Wright, Numerical Optimization, Springer-Verlag, 2000.

[7] L. T. Biegler, Nonlinear Programming, MOS-SIAM Series on Optimization, SIAM, 2010.

[8] P. T. Boggs, J. W. Tolle, Sequential quadratic programming, Acta Numerica (1995) 1–51.

[9] A. Griewank, Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation, no. 19 in Frontiers in Appl. Math., SIAM, Philadelphia, 2000.

[10] R. Quirynen, B. Houska, M. Vallerio, D. Telen, F. Logist, J. V. Impe, M. Diehl, Symmetric algorithmic differentiation based exact Hessian SQP method and software for economic MPC, in: Proceedings of the IEEE Conference on Decision and Control (CDC), 2014, pp. 2752–2757.

[11] R. Quirynen, M. Vukov, M. Diehl, Auto generation of implicit integrators for embedded NMPC with microsecond sampling times, in: M. Lazar, F. Allgöwer (Eds.), Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, 2012, pp. 175–180.

[12] R. Quirynen, M. Vukov, M. Zanon, M. Diehl, Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators, Optimal Control Applications and Methods 36 (2014) 685–704.

[13] A. Hindmarsh, P. Brown, K. Grant, S. Lee, R. Serban, D. Shumaker, C. Woodward, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Transactions on Mathematical Software 31 (2005) 363–396.

[14] M. Caracotsios, W. Stewart, Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations, Computers and Chemical Engineering. 9 (1985) 359–365.

[15] W. F. Feehery, J. E. Tolsma, P. I. Barton, Efficient sensitivity analysis of large-scale differential-algebraic systems, Applied Numerical Mathematics 25 (1997) 41–54.

[16] Y. Cao, S. Li, L. Petzold, Adjoint sensitivity analysis for differential-algebraic equations: algorithms and software, Journal of Computational and Applied Mathematics 149 (2002) 171–191.

[17] Y. Cao, S. Li, L. Petzold, R. Serban, Adjoint Sensitivity Analysis for Differential-Algebraic Equations: The Adjoint DAE System and its Numerical Solution, SIAM Journal on Scientific Computing 24 (3) (2003) 1076–1089.

[18] J. Betts, Practical Methods for Optimal Control Using Nonlinear Programming, SIAM, Philadelphia, 2001.

[19] D. B. Özyurt, P. I. Barton, Cheap Second Order Directional Derivatives of Stiff ODE Embedded Functionals, SIAM Journal on Scientific Computing 26 (2005) 1725–1743.

[20] R. M. Gower, M. P. Mello, A new framework for the computation of hessians, Optimization Methods and Software 27 (2) (2012) 251–273.

[21] R. Quirynen, B. Houska, M. Diehl, Symmetric hessian propagation for lifted collocation integrators in direct optimal control., in: Proceedings of the American Control Conference (ACC), 2016.

[22] E. B. Canto, J. R. Banga, A. A. Alonso, V. S. Vassiliadis, Restricted second order information for the solution of optimal control problems using control vector parameterization, Journal of Process Control 12 (2) (2002) 243 – 255.

[23] R. Hannemann, W. Marquardt, Continuous and discrete composite adjoints for the hessian of the lagrangian in shooting algorithms for dynamic optimization, SIAM journal on scientific computing 31 (6) (2010) 4675–4695.

[24] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems, 2nd Edition, Springer, Berlin Heidelberg, 1991.

[25] D. Kraft, On converting optimal control problems into nonlinear programming problems, in: K. Schittkowski (Ed.), Computational Mathematical Programming, Vol. F15 of NATO ASI, Springer, 1985, pp. 261–280.

[26] A. Walther, Automatic differentiation of explicit Runge-Kutta methods for optimal control, Computational Optimization and Applications 36 (1) (2006) 83–108.

[27] A. Sandu, Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part IV, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, Ch. On the Properties of Runge-Kutta Discrete Adjoints, pp. 550–557.

[28] J. Albersmeyer, Adjoint-based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems, Ph.D. thesis, Ruprecht-Karls-Universitität Heidelberg (2010).

[29] M. Alexe, A. Sandu, Forward and adjoint sensitivity analysis with continuous explicit Runge-Kutta schemes, Applied Mathematics and Computation 208 (2) (2009) 328–346.

[30] W. W. Hager, Rates of convergence for discrete approximations to unconstrained control problems, SIAM Journal on Numerical Analysis 13 (4) (1976) 449–472.

[31] B. Houska, H. J. Ferreau, M. Diehl, ACADO toolkit – an open source framework for automatic control and dynamic optimization, Optimal Control Applications and Methods 32 (3) (2011) 298–312.

[32] M. Diehl, H. G. Bock, J. Schlöder, R. Findeisen, Z. Nagy, F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations, Journal of Process Control 12 (4) (2002) 577–585.

[33] B. Houska, H. J. Ferreau, M. Diehl, An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range, Automatica 47 (10) (2011) 2279–2285.

[34] L. Wirsching, H. G. Bock, M. Diehl, Fast NMPC of a chain of masses connected by springs, in: Proceedings of the IEEE International Conference on Control Applications, Munich, 2006, pp. 591–596.

[35] G. P. Kumar, I. V. K. S. Sastry, M. Cidambaram, Periodic operation of a bioreactor with input multiplicities, The Canadian Journal of Chemical Engineering 71 (1993) 766–770.

[36] S. J. Parulekar, Analysis of forced periodic operations of continuous bioprocesses - single input variations, Chemical Engineering Science 53(14) (1998) 2481–2502.

[37] L. Ruan, X. Chen, Comparison of Several Periodic Operations of a Continous Fermentation Process, Biotechnol. Prog. 12 (1996) 286–288.

[38] R. Serban, A. Hindmarsh, CVODES: the sensitivity-enabled ODE solver in SUNDIALS, in: Proceedings of IDETC/CIE 2005, 2005.