Comput Optim Applic (2007) 36:83–108
DOI 10.1007/s10589-006-0397-3

Automatic differentiation of explicit Runge-Kutta methods for optimal control

Andrea Walther

Received: 5 August 2004 / Revised: 7 July 2005 / Published online: 21 November 2006

Abstract This paper considers the numerical solution of optimal control problems based on ODEs. We assume that an explicit Runge-Kutta method is applied to integrate the state equation in the context of a recursive discretization approach. To compute the gradient of the cost function, one may employ Automatic Differentiation (AD). This paper presents the integration schemes that are automatically generated when differentiating the discretization of the state equation using AD. We show that they can be seen as discretization methods for the sensitivity and adjoint differential equation of the underlying control problem. Furthermore, we prove that the convergence rate of the scheme automatically derived for the sensitivity equation coincides with the convergence rate of the integration scheme for the state equation. Under mild additional assumptions on the coefficients of the integration scheme for the state equation, we show a similar result for the scheme automatically derived for the adjoint equation. Numerical results illustrate the presented theoretical results.

Keywords Optimal control · Automatic differentiation · Sensitivity equation · Adjoint equation

1 Introduction

For numerous real-world applications, it is possible to influence the state or process under consideration by varying the value of several parameters. Quite often, this relation is modeled by a system of ordinary differential equations that involves control functions. The control functions can be used to minimize a cost function to obtain a desired state or process. There are many books and research papers on optimal control problems based on ODEs and many theoretical as well as numerical aspects are well studied.

In order to compute an optimal control for a given problem, we may employ derivative information in several representations. In general we distinguish two classes. The first one comprises approaches that are based on the Pontryagin maximum (or minimum) principle and

A. Walther (✉)
Institute of Scientific Computing, Technische Universität Dresden, Germany
e-mail: Andrea.Walther@tu-dresden.de

yields a multi-point boundary value problem for the state and the additional adjoint equation. Hence, these so-called *indirect methods* correspond to the solution of the continuous KKT system. An introduction to this technique can be found in Pesch [24]. Implementation issues as well as examples of applications are contained e.g. in Bulirsch et al. [4] and Hiltmann [22]. A software package based on an indirect method is described in Oberle [23].

The second class is termed *direct methods*. Here, in a first step one transforms the infinite-dimensional optimal control problem into a finite-dimensional optimization task. For that purpose, one may use collocation methods, see e.g. von Stryk [27] and Betts [1]. Subsequently, one solves the complete resulting discrete KKT system. Consequently, both the discretized state as well as the discretized control are treated as optimization variables by the optimization algorithm. These methods, which are known as *full discretization* concepts, lead to a huge number of equality constraints that have an almost block-diagonal Jacobian matrix. As alternative, only the discretized controls can be treated as optimization variables. Hence, the objective value can only be obtained by one integration of the state equation and the optimization is based on the non-trivial gradient of the cost function with respect to the discretized controls. These approaches are usually called *recursive discretization*. The name is inspired by the fact that all classical ODE integration schemes (e.g. Runge-Kutta schemes) define the solution to the state ODE recursively, time step by time step. We point the reader to Büskens and Maurer [6] as well as Bock and Plitt [2] for more information on these methods. One major problem of this approach is to find a correct method to compute the gradient information. For that purpose, one may derive a suitable integration scheme for the adjoint differential equation. Usually, these adjoint discretization schemes do not coincide with the integration schemes used for the state equation (see e.g. [5, 19]). Therefore, their computation is by no means obvious.

Over the last decades, several research groups have developed the technique of Automatic or Algorithmic Differentiation (AD), which allows the generation of exact derivative information for a given code segment. A comprehensive introduction to this method can be found in Griewank [16]. Furthermore, there are numerous examples for the application of AD in the optimal control context, see e.g. [8, 9, 12]. Theoretical aspects were for example studied in [13, 14]. Applying recursive discretizations, we can use AD to compute the consistent objective gradient which depends on the forward integration scheme as presented, for example, in [14]. That is, AD yields the exact discrete gradient information for the chosen discretization of the state equation. Therefore, the usage of an AD tool eliminates the hand-coding of derivative calculations, a rather involved and error-prone process. However, one has to keep in mind that AD computes the exact derivative of an approximation of the objective and may not yield an approximation to the exact derivatives of the objective. The purpose of this paper is to analyze this discrepancy and its impacts for the recursive discretization in more detail. That means, we analyse for a fixed control function the convergence rate of the discrete adjoint associated with the corresponding discretized control problem to the corresponding continuous adjoint. For that purpose, we present the discretization schemes automatically derived by AD to integrate the sensitivity equation and the adjoint equation of the underlying optimal control problem. Subsequently, we prove that the discretization method obtained for the sensitivity equation inherits the convergence properties of the discretization scheme used for the state equation. A similar result can be proven for the adjoint equation provided that some additional assumptions on the coefficients of the original discretization method are fulfilled. Moreover, we sketch the computation of the required gradient information using the approximations of the sensitivity equation and the adjoint equation, respectively. However, we also show that the gradient information provided by AD is the exact discrete derivative information, which need not coincide with the one based on the approximation of the sensitivity and adjoint equation, respectively. This aspect is subject of ongoing research and was studied for example in [14, 15].

Hence, we choose for a given fixed control the discretize-then-optimize approach and compare the approximation that we achieve with the continuous solutions of the optimize-then-discretize method. The analysis in this paper can be considered as a continuation of [15], where the effect of this difference is examined by means of two numerical examples. Here, we analyse the discrete adjoint information arising for one subproblem of a gradient-based optimization procedure. Therefore, it is related to [11, 21], where the convergence rate of the solution of the discretized control problem to the solution of the continuous problem is studied for the full discretization approach. That is, these papers study the difference between the optimal solutions of the continuous optimization problem and the discrete version.

The present paper has the following structure. In Section 2 we introduce the continuous optimal control problem to be considered. Furthermore, we discuss two possibilities of computing the objective gradient for the continuous formulation. Subsequently, we present the corresponding finite-dimensional optimization problem in Section 3. Section 4 introduces discretization schemes for computing an approximate solution of the sensitivity equation. We prove some convergence properties of the schemes generated by the forward mode of AD. Convergence results for the recursive discretization applying the reverse more of AD are proved in Section 5. Section 6 gives a numerical illustration of the obtained results. Finally, we draw some conclusions in Section 7.

2 The continuous optimal control problem and its gradient

We consider the following optimal control problem:

$$\begin{aligned} \text{Minimize} \quad J(y) &= \varphi(y(t_f)) && && (1) \\ \text{s.t.} \quad \frac{dy}{dt}(t) &= f(y(t), u(t)) && t \in [0, t_f] && (2) \\ y(0) &= y^0 && && (3) \end{aligned}$$

where $y(t) \in \mathbb{R}^n$ denotes the state and $u(t) \in \mathbb{R}^m$ denotes the control. The dynamics are given by a right-hand side function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $y^0 \in \mathbb{R}^n$ describes the initial conditions. To compute the value of the objective function, $\varphi : \mathbb{R}^n \to \mathbb{R}$ evaluates the state at the prescribed final time t_f. For simplicity, we assume that all functions are sufficiently smooth such that the existence of a solution is ensured and all necessary differentiations can be performed. Furthermore, no constraints on the control are considered yet. Box constraints and more complicated constraints on the control will be subject of further studies.

The state y is fully determined by the control u. Therefore, we may introduce an objective in terms of the control variable only. To this end, we define for a given control u and a corresponding solution $y = \psi(u)$ of the state Eqs. (2)–(3) the objective function

$$\tilde{J}(u) = J(\psi(u))$$

depending only on u. Since optima are locally characterized as roots of the gradient, we have a closer look at its representation. For that purpose, let the symbol D denote Fréchet derivatives in infinite-dimensional spaces. Applying the chain rule, one obtains

$$D\tilde{J}(u) = DJ(\psi(u))\, D\psi(u).$$

Now, one can employ two distinct ways for calculating $D\tilde{J}(u)$. The first one determines for a given control u the sensitivity $s(t) \in \mathbb{R}^n$ in the direction of $d(t) \in \mathbb{R}^m$ using the sensitivity equation:

$$\frac{ds}{dt}(t) = f_y(\psi(u)(t), u(t))\, s(t) + f_u(\psi(u)(t), u(t))\, d(t), \quad s(0) = 0. \tag{4}$$

One obtains immediately the *forward* or *sensitivity* representation of the gradient:

$$D\tilde{J}(u)d = \nabla\varphi(\psi(u)(t_f))\, s(t_f)\,. \tag{5}$$

Here, the attribute *forward* refers to the fact that the sensitivity equation has to be integrated in forward direction from 0 to t_f to find the Gâteaux variation $D\tilde{J}(u)d$. It should be noted that in (5), $\nabla\varphi(\psi(u)(t_f))$ denotes the gradient of φ with respect to y evaluated at $\psi(u)(t)$, and not the total derivative of the composite function $\varphi \circ \psi$.

The following alternative computes the complete $D\tilde{J}(u)$ by employing the adjoint differential equation

$$-\frac{d\lambda}{dt}(t) = f_y(\psi(u)(t), u(t))^T\, \lambda(t), \quad \lambda(t_f) = \nabla\varphi(\psi(u)(t_f))^T. \tag{6}$$

The solution $\lambda(t) \in \mathbb{R}^n$ of this linear ODE is called the adjoint variable and yields the gradient's *backward* or *adjoint* representation [3, Section 2.4]

$$(D\tilde{J}(u)(t))^T = f_u(\psi(u)(t), u(t))^T \lambda(t) \in \mathbb{R}^{m\times 1}. \tag{7}$$

It follows that one has to integrate the adjoint equation in *backward* direction from t_f to t for calculating the gradient at time t. The connection between the interpretation of the objective gradient as a function depending on time and its function space interpretation is illustrated by the following formula for the Gâteaux variation in the direction of $\bar{u}$:

$$D\tilde{J}(u)^T \bar{u} = \int_0^{t_f} D\tilde{J}(u)(t)^T\, \bar{u}(t)\, dt. \tag{8}$$

All AD-tools that the author is aware of provide the forward mode which can be seen as a discrete version of the sensitivity equation. On the other hand, not all AD-tools allow the application of the reverse mode that forms a discrete analog of the adjoint method. Therefore, we will consider sensitivity-based methods as well as adjoint-based methods to compute gradient information in this paper.

3 The discrete optimal control problem

To solve the problem (1)–(3) numerically, we have to perform some discretizations. Therefore, assume that the time interval $[0, t_f]$ is divided into $N - 1$ sub-intervals of equal length. Then the time grid consists of points

$$t_i = (i-1)\cdot h \qquad \text{for } i = 1, \dots, N \tag{9}$$

where $h = t_f/(N-1)$ is the time step length. For simplicity, we restrict here the presentation to uniform grids.

All control and state components will be approximated at the grid points only, where we use the notation

$$\begin{aligned} &y_i \text{ to approximate } y(t_i), \quad && \mathbf{y} = (y_1, \dots, y_N)^T \in \mathbb{R}^{Nn}, \\ &u_i \text{ to approximate } u(t_i), \quad && \mathbf{u} = (u_1, \dots, u_N)^T \in \mathbb{R}^{N}. \end{aligned}$$

Here, we restrict the analysis to one distributed control for notational simplicity. The argumentation for several distributed controls only complicates the formulation and can be easily integrated. We obtain the finite-dimensional NLP problem

$$\text{Minimize} \quad \tilde{\mathbf{J}}(\mathbf{u}) = \varphi(\psi_N(\mathbf{u}))$$

where $\tilde{\mathbf{J}} : \mathbb{R}^{Nm} \to \mathbb{R}$ and $\mathbf{y} = \psi(\mathbf{u}) = (\psi_1(\mathbf{u}), \dots, \psi_N(\mathbf{u}))^T$ approximately satisfies the state ODE (2) of the problem. Throughout the paper, we took ψ to represent an explicit Runge-Kutta scheme with constant step size h.

In dependence on the specific Runge-Kutta scheme applied, the control function u has to be evaluated at arguments $t_i + c_j h$ that may not lie on the time grid (9). Then, one can consider these intermediate values as independent optimization variables or as auxiliary values that are computed by an interpolation scheme using the values of $\mathbf{u}$ on the grid points. The first approach is applied for example in [21] for the full discretization. The latter is common for recursive discretization methods. Therefore, we assume throughout that the intermediate values $\tilde{u}_j$ of the control function at time $t_i + c_j h$, $0 < c_j < 1$, are computed by an interpolation polynomial p.

For constant and linear interpolation at time $\tilde{t}$ with $t_i < \tilde{t} < t_{i+1}$, only the values of u_i and u_{i+1} are required. Hence, the number of unknowns equals Nm. Quadratic interpolation is rarely used due to the resulting oscillations in the approximation of the control. Instead, one applies more frequently cubic interpolation based on a cubic spline, see e.g. [25]. This approach requires in addition to the Nm unknowns for linear interpolation only the values of u'_1 and u'_N. The corresponding coefficients of the polynomials are defined by

$$\alpha_i = u_i, \quad \beta_i = \frac{u_{i+1} - u_i}{h} - h\frac{2\gamma_i + \gamma_{i+1}}{6}, \quad \delta_i = \frac{\gamma_{i+1} - \gamma_i}{h}$$

where $\gamma = (\gamma_1, \dots, \gamma_N)$ is determined by the solution of a tridiagonal system

$$D\gamma = r \tag{10}$$

with a system matrix D independent of u. The right-hand side r is defined as

$$\begin{aligned} r_1 &= \frac{6}{h}\left(\frac{u_2 - u_1}{h} - u'_1\right), \quad r_N = \frac{6}{h}\left(u'_N - \frac{u_N - u_{N-1}}{h}\right), \\ r_i &= \frac{3(u_{i+1} - 2u_i + u_{i-1})}{h^2}, \quad i = 2, \dots, N-1. \end{aligned}$$

Table 1 Interpolation polynomials $p(v_i, t)$ of order ν

Type	Constant	Linear	Cubic
κ	1	2	4
$p(v_i, t)$	u_i	$(u_{i+1} - u_i t)/h + u_i$	$\alpha_i + \beta_i t + \gamma_i t^2/2 + \delta_i t^3/6$

Hence, the cubic interpolation is available at low cost. For the sake of notational simplicity, we will assume that the values of u'_1 and u'_N are fixed. However, the presented analysis can be easily extended to allow also varying derivatives. The discussed interpolation methods up to fourth order are shown in Table 1 and seem to cover numerous applications of the recursive discretization. To abbreviate the notation, we combine the first two arguments of the interpolation polynomial that belong to the control to one argument $v_i \equiv (u_i, u_{i+1}) \in \mathbb{R}^2$.

Then, we obtain the following algorithm for computing the discrete state vector $\mathbf{y}$ and the value of the objective function:

Algorithm 1: Integration of state equation

```
$y_1 = y^0$
For $i = 1, N-1, 1$
    For $j = 1, s, 1$
        $\tilde{y}_j = y_i + h \sum_{l=1}^{j-1} a_{jl} k_l, \quad \tilde{u}_j = p(v_i, c_j h)$
        $k_j = f(\tilde{y}_j, \tilde{u}_j)$
    $y_{i+1} = y_i + h \sum_{j=1}^{s} b_j k_j$
$\tilde{\mathbf{J}} = \varphi(y_N)$
```

with $c_j = \sum_{l=1}^{j-1} a_{jl}$ and $a_{jl} = 0$ for $l \geq j$. In the following sections we will extend Algorithm 1 to compute in addition to the state and the cost function also derivative information.

4 Forward mode of AD in recursive discretization

For a given vector function F evaluated at a certain point x and a direction $\dot{x}$, the scalar forward mode of AD yields the product $F'(x)\dot{x}$, i.e. the Jacobian $F'(x)$ multiplied by the vector $\dot{x}$ from the right. The evaluation cost for such a product can be bounded above by the product of a small constant and the cost to evaluate the function itself. In the literature the value of the constant varies between three and five depending on the considered computational complexity estimates [16].

In our context of an optimal control problem, the discrete evaluation of the cost function $\tilde{\mathbf{J}}$ given by Algorithm 1 defines the vector function F that has to be differentiated. Using the direction $\dot{\mathbf{u}} = (\dot{u}_1, \ldots, \dot{u}_N)^T \in \mathbb{R}^N$, the forward mode of AD applied to F yields the following integration procedure [16]:

Algorithm 2: Applying AD, forward differentiation of state equation

```
$y_1 = y^0, \quad \dot{y}_1 = \mathbf{0}_n$
For $i = 1, N-1, 1$
    For $j = 1, s, 1$
        $\tilde{y}_j = y_i + h \sum_{l=1}^{j-1} a_{jl} k_l, \quad \tilde{u}_j = p(v_i, c_j h),$
        $\dot{\tilde{y}}_j = \dot{y}_i + h \sum_{l=1}^{j-1} a_{jl} \dot{k}_l \quad \dot{\tilde{u}}_j = p_u(v_i, c_j h)\dot{\mathbf{u}}$
        $k_j = f(\tilde{y}_j, \tilde{u}_j), \quad \dot{k}_j = f_y(\tilde{y}_j, \tilde{u}_j)\dot{\tilde{y}}_j + f_u(\tilde{y}_j, \tilde{u}_j)\dot{\tilde{u}}_j$
    $y_{i+1} = y_i + h \sum_{j=1}^{s} b_j k_j, \quad \dot{y}_{i+1} = \dot{y}_i + h \sum_{j=1}^{s} b_j \dot{k}_j$
$\tilde{\mathbf{J}} = \varphi(y_N), \quad \dot{\tilde{\mathbf{J}}} = \nabla\varphi(y_N)\dot{y}_N$
```

Here, the customary notation in AD literature is used. Note that for cubic interpolation, the derivative $p_u(v_i, c_j h)$ takes also the differentiation of the coefficients $\alpha_i, \beta_i, \gamma_i, \delta_i$ with respect to $\mathbf{u}$ into account.

Employing the recursive discretization for solving the optimal control problem (1)–(3), one is interested in computing the complete gradient $\nabla\tilde{\mathbf{J}}(\mathbf{u})$ of the cost function with respect to the control variables $\mathbf{u} \in \mathbb{R}^N$. One obtains the exact discrete derivative information $\nabla\tilde{\mathbf{J}}(\mathbf{u})$ using the forward mode by choosing the N unit vectors as directions $\dot{\mathbf{u}}$ because of the following argument. Applying the chain rule, it follows that the derivative of $\tilde{\mathbf{J}}$ with respect to the ith component u_i has the representation

$$\frac{\partial\tilde{\mathbf{J}}(\mathbf{u})}{\partial u_i} = \sum_{l=2}^{N} \nabla\varphi(\psi_N(\mathbf{u})) \frac{\partial\psi_N(\mathbf{u})}{\partial\psi_{N-1}(\mathbf{u})} \cdots \frac{\partial\psi_{l+1}(\mathbf{u})}{\partial\psi_l(\mathbf{u})} \frac{\partial\psi_l(\mathbf{u})}{\partial u_i}. \tag{11}$$

Let $e_i \in \mathbb{R}^N$ be the ith unit vector. Then, one can conclude from the statements for $\dot{\tilde{u}}_j$, $\dot{k}_j$ and $\dot{y}$ that in the step $i = 1$ the derivative of ψ_2 with respect to u_i is added to $\dot{y}_2$. Furthermore from the statements for $\dot{\tilde{y}}_j$ and $\dot{k}_j$, one can see that in step $i > 1$ the vector $\dot{y}_i$ is first used to compute the derivative of $\psi_{i+1}(\mathbf{u})$ with respect to $\psi_i(\mathbf{u})$ as well as the derivative of $\psi_{i+1}(\mathbf{u})$ with respect to u_i. Subsequently, these values together with $\dot{y}_i$ form $\dot{y}_{i+1}$, i.e. one has

$$\dot{y}_{i+1} = \sum_{l=2}^{i+1} \frac{\partial\psi_{i+1}(\mathbf{u})}{\partial\psi_i(\mathbf{u})} \cdots \frac{\partial\psi_{l+1}(\mathbf{u})}{\partial\psi_l(\mathbf{u})} \frac{\partial\psi_l(\mathbf{u})}{\partial u_i}.$$

Finally, the vector $\dot{y}_N$ is multiplied by $\nabla\varphi(\psi_N(\mathbf{u}))$ to obtain the component of the gradient belonging to u_i. Hence, for evaluating $\nabla\tilde{\mathbf{J}}(\mathbf{u})$ one has to propagate N unit vectors through Algorithm 2. Obviously, this way to calculate $\nabla\tilde{\mathbf{J}}$ is very costly for high dimensions N. Some savings can be obtained by using the vector forward mode of AD, where a bundle of directions is propagated. Nevertheless, the reverse mode of AD should be used for gradient calculation for moderate and high dimensions whenever possible as illustrated in Section 5.

However, our aim was to analyze the approximate solution of the sensitivity Eq. (4) generated by AD. For that purpose, Algorithm 3 states one possibility to integrate the sensitivity equation applying the same Runge-Kutta scheme used for integrating the state equation:

Algorithm 3: Integration of state and sensitivity equation

```
$y_1 = y^0, \quad s_1 = \mathbf{0}_n$
For $i = 1, N-1, 1$
    For $j = 1, s, 1$
        $\tilde{y}_j = y_i + h \sum_{l=1}^{j-1} a_{jl} k_l, \qquad \tilde{u}_j = p(v_i, c_j h)$
        $\tilde{s}_j = s_i + h \sum_{l=1}^{j-1} a_{jl} \tilde{k}_l \qquad \tilde{d}_j = q(w_i, c_j h)$
        $k_j = f(\tilde{y}_j, \tilde{u}_j), \quad \tilde{k}_j = f_y(\tilde{y}_j, \tilde{u}_j)\tilde{s}_j + f_u(\tilde{y}_j, \tilde{u}_j)\tilde{d}_j$
    $y_{i+1} = y_i + h \sum_{j=1}^{s} b_j k_j, \quad s_{i+1} = s_i + h \sum_{j=1}^{s} b_j \tilde{k}_j$
$\tilde{\mathbf{J}} = \varphi(y_N)$
```

Here, $q(w_i, c_j h)$ with $w_i = (d_i, d_{i+1})$ interpolates the chosen direction $\mathbf{d} \in \mathbb{R}^N$ at the intermediate times $t_i + c_j h$ with the same interpolation order κ as p. Let us assume that the Runge-Kutta scheme applied in Algorithm 1 for the state integration is consistent of order ν and an appropriate interpolation order $\kappa \geq \nu$ is used. Then the resulting approximations $y_i, i = 1, \ldots, N$, are consistent of order ν. In addition, it follows that also the approximate solution s of the sensitivity equation obtained by Algorithm 3 is consistent of order ν. Hence, one obtains that the convergence order of the integration method proposed in Algorithm 3 for the sensitivity equation equals ν since the right-hand side f is assumed to be sufficiently smooth.

To obtain a related convergence result for Algorithm 2, we examine now the relation between Algorithms 2 and 3. As can be seen, there are similar recursions for $\dot{\tilde{y}}_j$ and $\tilde{s}_j$ as well as for $\dot{k}_j$ and $\tilde{k}_j$. The only difference lies in the direction $\dot{\tilde{u}}_j$ and $\tilde{d}_j$. Therefore, we have to examine the computation of $\dot{\tilde{u}}_j$ in more detail. For constant and linear interpolation it is very easy to check that $\dot{\tilde{u}}_j = p_u(v_i, c_j h)\dot{\mathbf{u}}$ also yields a constant and linear interpolation of $\dot{\mathbf{u}}$. For cubic interpolation, the system matrix D of (10) is constant with respect to $\mathbf{u}$. Let r_u denote the differentiation of r with respect to all components of $\mathbf{u}$. Then, the product $\tilde{r} = r_u \dot{\mathbf{u}}$ agrees with r if one substitutes $\dot{u}_i$ with u_i since r is only linear in $\mathbf{u}$. Furthermore, one has for the derivative γ_u of γ with respect to $\mathbf{u}$ the identity

$$\gamma_u \dot{\mathbf{u}} = D^{-1} r_u \dot{\mathbf{u}} = D^{-1} \tilde{r} = \tilde{\gamma}. \tag{12}$$

Analyzing also the remaining coefficient vectors of the cubic interpolation polynomial, e.g. $\tilde{\alpha} \equiv \alpha_u \dot{\mathbf{u}} = \dot{\mathbf{u}}$, one finds easily that

$$\dot{\tilde{u}}_j = p_u(v_i, c_j h)\dot{\mathbf{u}} = \tilde{\alpha}_i + \tilde{\beta}_i t + \tilde{\gamma}_i t^2/2 + \tilde{\delta}_i t^3/6 \tag{13}$$

is a cubic interpolation scheme for the given direction $\dot{\mathbf{u}}$ and the slopes $\dot{\mathbf{u}}'_1 = \dot{\mathbf{u}}'_N = 0$. It follows that Algorithms 2 and 3 yield the same approximate solution $\dot{y}_i$ and s_i, respectively, when $\dot{\mathbf{u}} = \mathbf{d}$ and $d'_1 = d'_N = 0$ holds. Hence, we have proven the following convergence result:

Theorem 4.1 (Convergence of forward mode discretization). *Assume that f and u are ν times continuously differentiable with $\nu \leq 4$. Suppose that the integration method applied* *in Algorithm* 1 *is convergent of order* ν*. Furthermore, let the order of the interpolation polynomial taken from Table* 1 *be not less than* ν*. Then one has for a given direction d with* $d_1' = d_N' = 0$ *that the approximate solution* $\dot{y}_i$*,* $i = 1, \ldots, N$*, generated by the forward mode of AD (Algorithm* 2*) converges with order* ν *to the continuous solution of the sensitivity Eq.* (4) *if* $\dot{\mathbf{u}} = \mathbf{d}$.

This theorem ensures that the discretization scheme generated by the forward mode of AD applied to the state equation inherits the convergence rate of the discretization scheme used for the state equation without any additional assumptions. Note that nonzero values for the derivative of $\mathbf{d}$ at the times t_0 and t_N can be easily taken into account. Using the gradient representation (5), we are now able to compute an approximation of the continuous gradient of order $\nu \leq 4$ based on the approximate solution $\dot{y}_i, i = 1, \ldots, N$, of the sensitivity equation by choosing the Nm unit vectors as directions $\dot{\mathbf{u}}$ in Algorithm 2. Obviously, this approach is usually very costly for reasonable discretizations of the original problem. Therefore, we will now turn to the reverse mode of AD that provides a cheap computation of gradient information.

5 Reverse mode of AD in recursive discretization

For a given vector function F evaluated at a certain point x with $y = F(x)$ and a weight vector $\bar{y}$, the scalar reverse mode of AD yields the product $F'(x)^T \bar{y}$, i.e. the transposed Jacobian $F'(x)^T$ multiplied by the vector $\bar{y}$ from the right. Once more the evaluation cost for such a product can be bounded above by the product of a small constant and the cost to evaluate the function itself. Depending on the considered complexity estimates, the value of the constant varies again between three and five [16]. Hence, the exact gradient of a scalar valued function given by a computer program can be computed via AD with an effort that is independent of the number of optimization variables. It seems that this impressive result is yet not well enough appreciated compared to the inexactness of finite differences and the effort to evaluate them.

In the context of an optimal control problem, the weight vector $\bar{\mathbf{J}} \in \mathbb{R}$ is set to $\bar{\mathbf{J}} = 1$ and we are interested in the gradient $\nabla\tilde{\mathbf{J}}(\mathbf{u})$ with $\mathbf{u} \in \mathbb{R}^N$. Then, the scalar reverse mode of AD applied to the state integration given by Algorithm 1 corresponds to the following integration procedure [16]:

Algorithm 4: Applying AD, reverse differentiation of state equation

```
$y_1 = y^0$
For $i = 1, N-1, 1$
    For $j = 1, s, 1$
        $\tilde{y}_j = y_i + h \sum_{l=1}^{j-1} a_{jl} k_l$
        $\tilde{u}_j = p(v_i, c_j h)$
        $k_j = f(\tilde{y}_j, \tilde{u}_j)$
    $y_{i+1} = y_i + h \sum_{j=1}^{s} b_j k_j$
```

```
$\tilde{\mathbf{J}} = \varphi(y_N)$
$\bar{y}_N = \nabla\varphi(y_N)^T, \bar{u}_i = \mathbf{0}_m \quad 1 \le i \le N$
For $i = N-1, 1, -1$
    $\bar{y}_i = \bar{y}_{i+1} \qquad \bar{k}_j = h\, b_j \bar{y}_{i+1}, \quad j = 1, \ldots, s$
    For $j = s, 1, -1$
        $\bar{\tilde{y}}_j = f_y(\tilde{y}_j, \tilde{u}_j)^T \bar{k}_j, \qquad \bar{\tilde{u}}_j = f_u(\tilde{y}_j, \tilde{u}_j)^T \bar{k}_j$
        $\bar{u} \mathrel{+}= p_u(v_i, c_j h)^T \bar{\tilde{u}}_j,$
        $\bar{y}_i \mathrel{+}= \bar{\tilde{y}}_j, \qquad \bar{k}_l \mathrel{+}= h\, a_{jl}\, \bar{\tilde{y}}_j, \quad l = 1, \ldots, j-1$
$\bar{y}_1 = 0.$
```

Once more, the common notation in AD literature is used, where $a \mathrel{+}= b$ denotes $a = a + b$. Applying this basic form of the scalar reverse mode, first the forward integration is performed to calculate the values $\tilde{y}_j$ and $\tilde{u}_j$. Then, the second for-loop computes the desired gradient $\nabla\tilde{\mathbf{J}}(\mathbf{u})$ assuming that the intermediate values $\tilde{y}_j$ and $\tilde{u}_j$ of state i are still available. These intermediate values can be stored sequentially onto a data structure and retrieved in reverse order when they are needed. This approach is easy to implement but results in a memory requirement that is proportional to the number N of time steps. To reduce this potential enormous memory requirement, optimal checkpointing procedures are available that can easily be incorporated into Algorithm 4 [20].

After these general remarks about the reverse mode of AD we return to the analysis of the approximate solutions of the state Eq. (2) and the adjoint Eq. (6) generated by the reverse mode of AD. First, one has to note that the integration method for the state is not changed. Therefore, the convergence order of the integration method applied for the state integration does not change. This is also true for other approaches to compute the gradient information $\nabla\tilde{\mathbf{J}}(\mathbf{u})$ using the recursive discretization. Therefore, we face the first important difference between the full discretization and the recursive discretization. Using the full discretization, it may happen that the discrete solution of the state does not converge to the continuous solution if a second order Runge-Kutta method is used, see [21, Section 6]. In the remaining part of this section, we will prove that for a fixed control function the approximate solution of the continuous adjoint equation converges to the exact solution of the continuous adjoint equation with order ν if the state approximation converges already with order ν to the exact solution of the state equation. Here, we face the second important difference between the full discretization and the recursive discretization: The usual order conditions for Runge-Kutta schemes are shown in Table 2, where the conditions for order ν comprise those listed in Table 2 for that specific order as well as those for all lower orders. Using the full discretization, it was proven by Hager that additional

Table 2 Order ν of a Runge-Kutta discretization for recursive discretization

ν	Conditions with $c_i = \sum_{j=1}^{s} a_{ij}$, $d_i = \sum_{j=1}^{s} b_j a_{ji}$
1	$\sum b_i = 1$
2	$\sum b_i c_i = \frac{1}{2}$
3	$\sum c_i d_i = \frac{1}{6}$, $\sum b_i c_i^2 = \frac{1}{3}$
4	$\sum b_i c_i^3 = \frac{1}{4}$, $\sum b_i c_i a_{ij} c_j = \frac{1}{8}$, $\sum b_i a_{ij} c_j^2 = \frac{1}{12}$, $\sum b_i a_{ij} a_{jk} c_k = \frac{1}{24}$

conditions have to be fulfilled for $\nu \geq 3$ to ensure the order ν also in the optimal control context [21].

With respect to the approximate solution of the adjoint equation given by the reverse mode of AD, one obtains from Algorithm 4 the recursion

$$\bar{y}_i = \bar{y}_{i+1} + \sum_{j=1}^{s} \bar{\tilde{y}}_j, \qquad \bar{y}_N = \nabla\varphi(y_N)^T, \tag{14}$$

$$\bar{\tilde{y}}_j = f_y(\tilde{y}_j, \tilde{u}_j)^T \bar{k}_j = f_y(\tilde{y}_j, \tilde{u}_j)^T \left(h\, b_j\, \bar{y}_{i+1} + h \sum_{l=j+1}^{s} a_{lj}\, \bar{\tilde{y}}_l \right).$$

Now, we transform this recursion into a formulation that is better suited for the analysis. This modification is similar to the one used by Hager [21]. We introduce a new variable $\hat{y}_j$ by setting

$$\hat{y}_j = \bar{y}_{i+1} + \sum_{l=j+1}^{s} \frac{a_{lj}}{b_j} \bar{\tilde{y}}_l, \quad j = 1, \dots, s, \tag{15}$$

where we assume $b_j > 0$, $j = 1, \dots, s$. It follows that

$$\bar{\tilde{y}}_j = h\, b_j\, f_y(\tilde{y}_j, \tilde{u}_j)^T \hat{y}_j. \tag{16}$$

Exploiting the identities (15) and (16) yields

$$\hat{y}_j - \bar{y}_{i+1} = \sum_{l=j+1}^{s} \frac{a_{lj}}{b_j} \bar{\tilde{y}}_l = \sum_{l=j+1}^{s} \frac{a_{lj}}{b_j} \big(h\, b_l\, f_y(\tilde{y}_l, \tilde{u}_l)^T \hat{y}_l \big).$$

Using (14) and (16), we obtain the recursion

$$\bar{y}_i = \bar{y}_{i+1} + h \sum_{j=1}^{s} b_j\, f_y(\tilde{y}_j, \tilde{u}_j)^T \hat{y}_j, \quad \bar{y}_N = \nabla\varphi(y_N)^T,$$

$$\hat{y}_j = \bar{y}_{i+1} + \sum_{l=j+1}^{s} \frac{a_{lj}}{b_j} (h\, b_l\, f_y(\tilde{y}_l, \tilde{u}_l)^T \hat{y}_l)$$

which marches still backwards. Therefore, we now change the direction of the integration. This yields

$$\bar{y}_{i+1} = \bar{y}_i - h \sum_{j=1}^{s} b_j\, f_y(\tilde{y}_j, \tilde{u}_j)^T \hat{y}_j, \quad \bar{y}_N = \nabla\varphi(y_N)^T$$

$$\begin{aligned} \hat{y}_j &= \bar{y}_i - h \sum_{l=1}^{s} b_l\, f_y(\tilde{y}_l, \tilde{u}_l)^T \hat{y}_l + \sum_{l=j+1}^{s} \frac{a_{lj}}{b_j} \big(h\, b_l\, f_y(\tilde{y}_l, \tilde{u}_l)^T \hat{y}_l \big) \\ &= \bar{y}_i - h \sum_{l=1}^{s} \bar{a}_{jl} f_y(\tilde{y}_l, \tilde{u}_l)^T \hat{y}_l \quad \text{with} \quad \bar{a}_{jl} = \frac{b_j b_l - b_l a_{lj}}{b_j} \end{aligned}$$

assuming $a_{lj} = 0$ if $l \leq j$. Defining $g(y, \bar{y}, u) = -f_y(y, u)^T \bar{y}$, we obtain

$$\bar{y}_{i+1} = \bar{y}_i + h \sum_{j=1}^{s} b_j \, g(\tilde{y}_j, \hat{y}_j, \tilde{u}_j), \quad \bar{y}_N = \nabla\varphi(y_N)^T \tag{17}$$

$$\hat{y}_j = \bar{y}_i + h \sum_{l=1}^{s} \bar{a}_{jl} g(\tilde{y}_l, \hat{y}_l, \tilde{u}_l), \quad \tilde{y}_j = y_i + h \sum_{l=1}^{j-1} a_{jl} f(\tilde{y}_l, \tilde{u}_l), \tag{18}$$

which requires the knowledge of the intermediate values $y_1, \ldots, y_N$. The derived integration scheme (17)–(18) for the adjoint Eq. (6) is very similar to the one obtained by Hager for the full discretization. However, the important difference is given by the fact that for the recursive discretization the function f depends on the state and the control as well as the function g on the state, the control, and the costate whereas for the full discretization the functions f, i.e. the right-hand side of the state equation, and ϕ, i.e. the right-hand side of the costate equation, both depend on the state and the costate. In addition, for the recursive discretization we interpolate the control at the intermediate points $t_i + c_j h$ and do not consider the intermediate values as additional control variables. Hence, we can not simply refer to the consistency proof by Hager [21], although there will be a significant similarity of the proofs.

Since the summations for $\hat{y}_j$ and $\tilde{y}_j$ involve different coefficients $\bar{a}_{jl}$ and a_{jl}, it is also not possible to apply the usual consistency theory for Runge-Kutta schemes. Therefore, we use a similar approach to that of Butcher [7] but take the specific situation with the different coefficients $\bar{a}_{jl}$ and a_{jl} as well as the interpolation of the control into account. Despite the fact that the analysis works also for higher order, we restrict the proof to schemes of order less than 5 since is seems quite unusual to have an appropriate interpolation that maintains a higher order of the applied Runge-Kutta method.

Defining the function $\tilde{g} = (f, g, u')^T$ acting on $(y, \bar{y}, u)$, the solution λ of the adjoint Eq. (6) has the Taylor expansion

$$\begin{aligned}\lambda(t+h) = \lambda(t) + gh + \frac{1}{2} g'\tilde{g}\, h^2 + \frac{1}{6}[g''\tilde{g}^2 + g'\tilde{g}']h^3 \\ + \frac{1}{24}[g'''\tilde{g}^3 + 3g''\tilde{g}\tilde{g}' + g'\tilde{g}'']h^4 + \mathcal{O}(h^5).\end{aligned}$$

Here, the functions g and $\tilde{g}$ as well as their derivatives are evaluated at the point $(y(t), \lambda(t), u(t))$. The derivatives should be viewed in an operator context introduced for example in [7] or [26]. Hence, the derivative g' operates on a vector and yields a scalar. The second derivative g'' acts on a pair of vectors to give a vector. Finally, one has $\tilde{g}' = (f_y f + f_u u', g_y f + g_{\bar{y}} g + g_u u', u'')^T$ and so on.

For the Taylor expansion around the approximations $\bar{y}_i$, we define

$$\zeta_1(h) = \begin{pmatrix} y_i \\ \bar{y}_i \\ u_i \end{pmatrix} \quad \text{and} \quad \zeta_j(h) = \begin{pmatrix} \tilde{y}_j \\ \hat{y}_j \\ \tilde{u}_j \end{pmatrix} \quad 1 < j \leq s$$

as well as the functions

$$G(\zeta) = \sum_{j=1}^{s} b_j g(\zeta_j) \quad \text{and} \quad F_j(\zeta) \;=\; \begin{pmatrix} \sum_{l=1}^{j-1} a_{jl} f(\tilde{y}_l, \tilde{u}_l) \\ \sum_{l=1}^{s} \bar{a}_{jl} g(\tilde{y}_l, \hat{y}_l, \tilde{u}_l) \\ (p(v_i, c_j h) - u_i)/h \end{pmatrix}.$$

Then, one has

$$\zeta(h) = \zeta(0) + hF(\zeta(h)) \quad \text{and} \quad \bar{y}_{i+1} \;=\; \bar{y}_i + hG(\zeta(h)).$$

Expanding $G(\zeta(h))$ in a Taylor series around $h = 0$, we obtain

$$\begin{aligned} \bar{y}_{i+1} = \bar{y}_i + hG + \frac{1}{2}[2G'F]h^2 + \frac{1}{6}[3G''F^2 + 6G'F'F]h^3 \\ + \frac{1}{24}[4G'''F^3 + 24G''FF'F + 24G'(F')^2F + 12G'F''F^2]\,h^4 + \mathcal{O}(h^5), \end{aligned}$$

where G, F, and their derivatives are evaluated at $\zeta(0)$. Now, we can prove the main result of this section:

Theorem 5.1 (Convergence of reverse mode discretization I). *Assume that f and u are ν times continuously differentiable with $\nu \leq 4$. Let the integration method applied in Algorithm* 1 *be of order ν with coefficients $b_j > 0$, $1 \leq j \leq s$, i.e., it fulfills the conditions of Table* 2 *up to order ν. Furthermore, let the order of the interpolation polynomial taken from Table* 1 *be not less than ν. Then, the approximate solution $\bar{y}_i$, $i = 1, \ldots, N$, generated by the reverse mode of AD (Algorithm* 4*) converges with order ν to the continuous solution of the adjoint Eq.* (6).

Proof: We have to check for a given ν that the Taylor expansions of λ and $\bar{y}$ agree through all terms of order h^κ, $\kappa = 1, \ldots, \nu$. For that purpose, the arguments of the functions f and g are skipped since they are always evaluated at the point $(y_i, \bar{y}_i, u_i)$.

For $\nu = 1$, the equation

$$g = \sum b_j g$$

has to be fulfilled, which yields with $\sum b_j = 1$ the condition for order 1 as stated in Table 2. Here, we employ the summation convention that if an index range does not appear on a summation sign then the summation is over each index, taking values from 1 to s.

For $\nu = 2$, we introduce abbreviations $c_j = \sum_{l=1}^{j-1} a_{jl}$, $\bar{c}_j = \sum_{l=1}^{s} \bar{a}_{jl}$, and $\hat{u}_j = (p(v_i, c_j h) - u_i)/h = (\tilde{u}_j - u_i)/h$. One has to check whether

$$g'\tilde{g} = g_y f + g_{\bar{y}} g + g_u u_i' \tag{19}$$

agrees with

$$2G'F = 2\sum_{j=1}^{s} b_j(c_j g_y f + \bar{c}_j g_{\bar{y}} g + g_u \hat{u}_j) \tag{20}$$

up to terms of order 1. One obtains immediately the conditions

$$\sum b_j c_j = \frac{1}{2} = \sum b_j \bar{c}_j.$$

The first equality holds because of the usual order conditions for Runge-Kutta schemes. Hence, it is fulfilled because the scheme for the forward integration (Algorithm 1) is of order 2. The second equality is obtained by

$$\sum b_j \bar{c}_j = \sum b_j \frac{b_j b_l - b_l a_{lj}}{b_j} = \sum b_j - \sum b_j c_j = \frac{1}{2}.$$

Additionally we have to take the interpolation of the control into account. For the interpolation polynomials of order $\kappa \geq 2$ that are given in Table 2, we have

$$\hat{u}_j = c_j u_i' + \mathcal{O}(h). \tag{21}$$

Hence, the Taylor expansions of λ and $\bar{y}$ agree through all terms of order κ, $\kappa = 1, 2$ if the conditions for order $\kappa \leq 2$ as stated in Table 2 are fulfilled.

For $\nu = 3$, one obtains that $g_{\bar{y}\bar{y}} = 0$ since g depends only linearly on $\bar{y}$. Hence, we have to consider

$$g''\tilde{g}^2 = g_{yy} ff + 2g_{y\bar{y}} fg + (2g_{yu} f + 2g_{\bar{y}u} g + g_{uu} u_i')u_i' \tag{22}$$

and

$$3G''F^2 = 3\sum b_j\big(c_j^2 g_{yy} ff + 2c_j\bar{c}_j g_{y\bar{y}} fg + (2c_j g_{yu} f + 2\bar{c}_j g_{\bar{y}u} g + g_{uu}\hat{u}_j)\hat{u}_j\big). \tag{23}$$

For the terms $g_{yy} ff$ and $g_{y\bar{y}} fg$ the identities

$$\frac{1}{3} = \sum b_j c_j^2 = \sum b_j c_j \bar{c}_j \tag{24}$$

must hold. The first one is again an usual order condition for Runge-Kutta schemes. Therefore it is valid since the scheme applied in Algorithm 1 is supposed to be of order 3. For the next equality, one has

$$\sum b_j c_j \bar{c}_j = \sum b_j c_j \frac{b_j b_l - b_l a_{lj}}{b_j} = \sum b_j c_j \left(1 - \frac{1}{b_j}\sum_{l=1}^{s} b_l a_{lj}\right)$$
$$= \frac{1}{2} - \sum b_l a_{lj} c_j = \frac{1}{6}.$$

With respect to the control u, condition (21) holds for the interpolation polynomial of order $\kappa \geq 3$. This fact together with the equality (24) yields that (22) and (23) agree except for terms of higher order. Second, we examine

$$g'\tilde{g}' = g_y(f_y f + f_u u_i') + g_{\bar{y}}(g_y f + g_{\bar{y}} g + g_u u_i') + g_u u_i'' \tag{25}$$

and

$$G'F'F = \sum b_j(a_{jl}\, g_y(c_l f_y f + f_u \hat{u}_l) + \bar{a}_{jl}\, g_{\bar{y}}(c_l g_y f + \bar{c}_l g_{\bar{y}} g + g_u \hat{u}_l)), \tag{26}$$

where no term $b_j g_u$ occurs in (26) since the derivative of the third entry of F_j, i.e., of $(p(v_i, 0) - u_i)/h$, with respect to u_i vanishes for the interpolation polynomial of order κ with $\kappa \geq 3$ at $t = 0$. It follows from the Taylor expansions of λ and $\bar{y}$ that the equations

$$\frac{1}{6} = \sum b_j a_{jl} c_l \;=\; \sum b_j \bar{a}_{jl} c_l \;=\; \sum b_j \bar{a}_{jl} \bar{c}_l \tag{27}$$

must be valid. These equalities cover three terms of (25). The first one is an usual condition for Runge-Kutta methods and holds because of the order of the forward integration. For the second and third equation, we obtain

$$\sum b_j \bar{a}_{jl} c_l = \sum b_j \frac{b_j b_l - b_l a_{lj}}{b_j} c_l = \sum b_j b_l c_l - \sum b_l a_{lj} c_l = \frac{1}{6},$$

$$\sum b_j \bar{a}_{jl} \bar{c}_l = \sum\; b_j \frac{b_j b_l - b_l a_{lj}}{b_j} \left(1 - \frac{1}{b_l} \sum_{k=1}^{s} b_k a_{kl}\right) = \frac{1}{6}.$$

Applying again the property (21) for the interpolation polynomials of order $\kappa \geq 3$ yields for the terms $g_y f_u u_i'$ and $g_{\bar{y}} g_u u_i'$ of (25) two equalities of (27) both of which are valid. The remaining term $g_u u_i''$ of (26) together with the term $g_u u_i'$ of (19) forms the sum

$$g_u \left(\frac{h^2}{2} u_i' + \frac{h^3}{6} u_i''\right)$$

in the Taylor expansion of λ. Due to the interpolation order $\nu \geq 3$, we obtain

$$p(v_i, c_j h) = u_i + c_j h u_i' + \frac{1}{2} c_j^2 h^2 u_i'' + \frac{1}{6} c_j^3 h^3 u_i''' + \mathcal{O}(h^4) \tag{28}$$

(see e.g. [25, Property 8.3]). It follows that we have for the term $g_u \hat{u}_j$ of (20) not only $g_u \hat{u}_j = g_u c_j u_i' + \mathcal{O}(h)$, but for the corresponding term in the Taylor expansion of $\bar{y}$ that

$$\begin{aligned} h^2 \sum b_j g_u \hat{u}_j &= h^2 g_u \sum b_j \big(c_j u_i' + c_j^2 h u_i''/2 + \mathcal{O}(h^2)\big) \\ &= g_u \left(\frac{h^2}{2} u_i' + \frac{h^3}{6} u_i'' + \mathcal{O}(h^4)\right). \end{aligned} \tag{29}$$

Hence, for an interpolation order $\kappa \geq 3$ the Eqs. (25) and (26) agree except for terms of higher order.

For $\nu = 4$, first one has

$$\begin{aligned} g'''\tilde{g}^3 = &\ g_{yyy}f^3 + 3g_{yy\bar{y}}f^2g + \left(3g_{yyu}f^2 + 6g_{y\bar{y}u}fg\right)u_i' \\ &+ (3g_{yuu}f + 3g_{\bar{y}uu}g)(u_i')^2 + g_{uuu}g(u_i')^3 \end{aligned} \tag{30}$$

and

$$\begin{aligned} 4G'''F^3 = 4\sum b_j\big(&c_j^3g_{yyy}f^3 + 3c_j^2\bar{c}_jg_{yy\bar{y}}f^2g + \left(3c_j^2g_{yyu}f^2 + 6c_j\bar{c}_jg_{y\bar{y}u}fg\right)u_i' \\ &+ (3c_jg_{yuu}f + 3\bar{c}_jg_{\bar{y}uu}g)(u_i')^2 + g_{uuu}g(u_i')^3\big) \end{aligned} \tag{31}$$

when taking $g_{\bar{y}\bar{y}} = 0$ into account. Hence, the equalities

$$\frac{1}{4} = \sum b_jc_j^3 = \sum b_jc_j^2\bar{c}_j \tag{32}$$

must hold for the first four terms of (30). Once more, the first one is an usual order condition for Runge-Kutta schemes and holds because of the order of the forward integration. For the next equality, one obtains

$$\sum b_jc_j^2\bar{c}_j = \sum b_jc_j^2\Big(1 - \frac{1}{b_j}\sum b_la_{lj}\Big) = \frac{1}{2} - \sum b_la_{lj}c_j^2 = \frac{1}{4}.$$

Employing condition (21) for the interpolation polynomial of order $\nu = 4$ together with the equality (32) yield that (30) and (31) agree except for terms of higher order. Second, we consider the term

$$\begin{aligned} 3g''\tilde{g}\tilde{g}' = &\ 3((g_{yy}f + g_{y\bar{y}}g + g_{yu}u_i')(f_yf + f_uu_i') + (g_{y\bar{y}}f + g_{\bar{y}u}u_i')(g_yf + g_{\bar{y}}g + g_uu_i') \\ &+ u_i''(g_{yu}f + g_{\bar{y}u}g + g_{uu}u_i')) \end{aligned} \tag{33}$$

as well as

$$\begin{aligned} 24G''FF'F = &\ 24\sum\sum(b_ja_{jl}(c_jg_{yy}f + \bar{c}_jg_{y\bar{y}}g + g_{yu}\hat{u}_j)(c_lf_yf + f_u\hat{u}_l)) \\ &+ 24\sum(b_j\bar{a}_{jl}(c_jg_{y\bar{y}}f + g_{\bar{y}u}\hat{u}_j)(c_lg_yf + \bar{c}_lg_{\bar{y}}g + g_u\hat{u}_l)). \end{aligned} \tag{34}$$

It follows that the equalities

$$\frac{1}{8} = \sum b_jc_ja_{jl}c_l = \sum b_j\bar{c}_ja_{jl}c_l = \sum b_jc_j\bar{a}_{jl}c_l = \sum b_jc_j\bar{a}_{jl}\bar{c}_l \tag{35}$$

must be fulfilled. The first one is again an usual order condition for Runge-Kutta schemes that holds since the order of the forward integration is assumed to be 4, whereas the next three equalities can be proved by

$$\sum b_j\bar{c}_j a_{jl}c_l = \sum b_j\left(1-\frac{1}{b_j}\sum_{k=1}^{s} b_k a_{kj}\right)a_{jl}c_l = \sum b_j a_{jl}c_l - \sum b_k a_{kj}a_{jl}c_l = \frac{1}{8},$$

$$\sum b_j c_j \bar{a}_{jl}c_l = \sum b_j c_j \frac{b_j b_l - b_l a_{lj}}{b_j} c_l = \sum b_j c_j b_l c_l - \sum b_l c_l a_{lj} c_j = \frac{1}{8},$$

$$\begin{aligned}\sum b_j c_j \bar{a}_{jl}\bar{c}_l &= \sum b_j c_j \frac{b_j b_l - b_l a_{lj}}{b_j}\left(1-\frac{1}{b_l}\sum_{k=1}^{s} b_k a_{kl}\right)\\ &= \sum b_j c_j b_l - \sum b_l a_{lj} c_j - \sum b_j c_j b_k a_{kl} + \sum b_k a_{kl} a_{lj} c_j = \frac{1}{8}.\end{aligned}$$

Employing condition (21) for the interpolation polynomial of order $\nu = 4$ together with the equality (35) yields that (33) and (34) agree up to terms of higher order except for the part involving u_i'', namely $u_i''(g_{yu}f + g_{\bar{y}u}g + g_{uu}u_i')$. We will examine this one later. Next, we analyze

$$\begin{aligned} g'\tilde{g}'' = \; & g_y\left[f_{yy}f^2 + 2f_{yu}fu_i' + f_y(f_y f + f_u u_i') + f_{uu}(u_i')^2\right]\\ & + g_{\bar{y}}\left[g_{yy}f^2 + 2g_{y\bar{y}}fg + 2g_{yu}fu_i' + 2g_{\bar{y}u}gu_i'\right.\\ & \left. + g_{uu}(u_i')^2 + g_y(f_y f + f_u u_i') + g_{\bar{y}}(g_y f + g_{\bar{y}}g + g_u u_i')\right]\\ & + u_i''(g_y f_u + g_{\bar{y}}g_u) + g_u u_i''' \end{aligned} \tag{36}$$

as well as

$$\begin{aligned} 12G'F''F^2 = \; & 12\sum b_j a_{jl}\, g_y\left[c_l^2 f_{yy}f^2 + 2c_l f_{yu}f\hat{u}_l + f_{uu}(\hat{u}_l)^2\right]\\ & + 12\sum b_j\bar{a}_{jl}\, g_{\bar{y}}\left[c_l^2 g_{yy}f^2 + 2c_l\bar{c}_l g_{y\bar{y}}fg + 2c_l g_{yu}f\hat{u}_l + 2\bar{c}_l g_{\bar{y}u}g\hat{u}_l + g_{uu}(\hat{u}_l)^2\right] \end{aligned} \tag{37}$$

and

$$\begin{aligned} G'(F')^2F = & \sum b_j a_{jl}a_{lk}g_y f_y(c_k f_y f + f_u\hat{u}_k) + \sum b_j\bar{a}_{jl}a_{lk}g_{\bar{y}}g_y(c_k f_y f + f_u\hat{u}_k)\\ & + \sum b_j\bar{a}_{jl}\bar{a}_{lk}g_{\bar{y}}^2(c_k g_y f + \bar{c}_k g_{\bar{y}}g + g_u\hat{u}_k). \end{aligned} \tag{38}$$

From (36) and (37), we can conclude that

$$\frac{1}{12} = \sum b_j a_{jl}c_l^2 = \sum b_j\bar{a}_{jl}c_l^2 = \sum b_j\bar{a}_{jl}c_l\bar{c}_l$$

must be valid, where the first one is an usual order condition for Runge-Kutta schemes. Hence it is fulfilled because of the order of the forward integration. The remaining two equalities hold since we have

$$\sum b_j \bar{a}_{jl} c_l^2 = \sum b_j \frac{b_j b_l - b_l a_{lj}}{b_j} c_l^2 = \sum b_j b_l c_l^2 - \sum b_l c_l^3 = \frac{1}{12},$$

$$\begin{aligned} \sum b_j \bar{a}_{jl} c_l \bar{c}_l &= \sum (b_j - a_{lj}) c_l \left(b_l - \sum_{k=1}^{s} b_k a_{kl} \right) \\ &= \sum b_l c_l - \sum b_k a_{kl} c_l b_j - \sum b_l c_l a_{lj} + \sum b_k a_{kl} c_l a_{lj} = \frac{1}{12}. \end{aligned}$$

From (36), (38), the Taylor expansions of λ and $\bar{y}$ as well as the property (21), it follows that

$$\frac{1}{24} = \sum b_j a_{jl} a_{lk} c_k = \sum b_j \bar{a}_{jl} a_{lk} c_k = \sum b_j \bar{a}_{jl} \bar{a}_{lk} c_k = \sum b_j \bar{a}_{jl} \bar{a}_{lk} \bar{c}_k$$

must hold. Once more, the first one is fulfilled because the scheme applied in Algorithm 1 is assumed to be of order 4. The next three identities are valid because

$$\begin{aligned} \sum b_j \bar{a}_{jl} a_{lk} c_k &= \sum (b_j b_l - b_l a_{lj}) a_{lk} c_k = \sum b_j b_l a_{lk} c_k - \sum b_l a_{lj} a_{lk} c_k = \frac{1}{24}, \\ \sum b_j \bar{a}_{jl} \bar{a}_{lk} c_k &= \sum (b_j - a_{lj})(b_l b_k - b_k a_{kl}) c_k \\ &= \sum b_k c_k - \sum b_k a_{kl} c_k - \sum b_l a_{lj} b_k c_k + \sum b_k c_k a_{kl} a_{lj} = \frac{1}{24}, \\ \sum b_j \bar{a}_{jl} \bar{a}_{lk} \bar{c}_k &= \sum (b_j - a_{lj})(b_l - a_{kl}) \left(b_k - \sum_{i=1}^{s} b_i a_{ik} \right) \\ &= 1 - \sum b_i c_i - \sum b_l c_l + \sum d_k c_k - \sum b_l c_l + \sum b_l c_l b_i c_i \\ &\quad + \sum d_l c_l - \sum b_i a_{ik} a_{kl} c_l = \frac{1}{24}. \end{aligned}$$

The remaining terms are $u_i''(3g_{yu} f + 3g_{\bar{y}u} g + 3g_{uu} u_i' + g_y f_u + g_{\bar{y}} g_u)$ and $g_u u_i'''$. The terms involving u_i' in (22) and (25) as well as the terms involving u_i'' in (33) and (36) contribute via the sum

$$\begin{aligned} &\frac{u_i'}{6}(2g_{yu} f + 2g_{\bar{y}u} g + g_{uu} u_i' + g_y f_u + g_{\bar{y}} g_u) h^3 \\ &\quad + \frac{u_i''}{24}(3g_{yu} f + 3g_{\bar{y}u} g + 3g_{uu} u_i' + g_y f_u + g_{\bar{y}} g_u) h^4 \end{aligned} \tag{39}$$

to the Taylor expansion of λ. On the other hand, we have from (23) and (26)

$$\frac{h^3}{6} \sum_{j=1}^{s} b_j \left((6c_j g_{yu} f + 6\bar{c}_j g_{\bar{y}u} g + 3g_{uu} \hat{u}_j) \hat{u}_j + \sum_{l=1}^{s} 6(a_{jl}\, g_y f_u + \bar{a}_{jl} g_{\bar{y}} g_u) \hat{u}_l \right) \tag{40}$$

as contribution to the Taylor expansion of $\bar{y}$. The Eq. (28) yields for the interpolation order $\nu \geq 4$

$$\hat{u}_j = c_j u_i' + \frac{c_j^2 h u_i''}{2} + \frac{c_j^3 h^2 u_i'''}{6} + \mathcal{O}(h^3). \tag{41}$$

Therefore, it is easy to check that (39) and (40) agree except for terms of higher order. Furthermore, in the Taylor expansion of λ, the terms $g_u u_i'$, $g_u u_i''$, and $g_u u_i'''$ occur in the sum

$$g_u\left(\frac{h^2}{2}u_i' + \frac{h^3}{6}u_i'' + \frac{h^4}{24}u_i'''\right).$$

Because of (41), we have for the term $g_u \hat{u}_j$ of (20) not only (29), but for the corresponding term in the Taylor expansion of $\bar{y}$ that

$$\begin{aligned} h^2 \sum b_j g_u \hat{u}_j &= h^2 g_u \sum b_j \left(c_j u_i' + \frac{c_j^2 h u_i''}{2} + \frac{c_j^3 h^2 u_i'''}{6} + \mathcal{O}(h^3)\right) \\ &= g_u\left(\frac{h^2}{2}u_i' + \frac{h^3}{6}u_i'' + \frac{h^4}{24}u_i''' + \mathcal{O}(h^5)\right). \end{aligned}$$

Therefore, also the remaining terms agree except for terms of higher order.

We proved so far that the approximate solution generated by the scalar reverse mode of AD has the consistency order ν. Since we assume also that the functions f and u are sufficiently smooth on the whole time interval, one has that the convergence order of the integration scheme obtained by applying the scalar reverse mode of AD equals ν. □

Hence, we have proved that for a fixed control the approximate solution of the adjoint equation generated by the reverse mode of AD converges at the same rate as the approximate solution of the state equation. That is, the discretization scheme generated automatically for the adjoint differential equation by applying the technique of Automatic Differentiation has the same convergence order as the discretization scheme that is used for the forward integration under fairly mild additional assumptions. This result differs from the convergence result for the full discretization where the convergence rate of the optimal solution of the discretized control problem to the optimal solution of the continuous problem is considered. Analyzing the proof, this difference is due to the fact that the terms involving second- or higher-order derivatives of g with respect to $\bar{y}$ vanish for the recursive discretization. For that reason, no additional conditions on the coefficients are needed. However, it may happen that due to constraints on the control the right-hand sides f and g are only Lipschitz continuous. This problem will not be discussed here but is subject to further investigations.

It is possible to eliminate the restriction $b_i > 0$ for Runge-Kutta schemes up to second order using the full discretization approach [11]. We are able to prove a related result for the recursive discretization. Instead of the recursion (17)–(18), we exploit another formulation of the discretization scheme. It can be shown by inspection for $s \leq 4$ and using a rather technical induction for $s > 4$ that (17)–(18) is equivalent to the recursion

$$\bar{y}_{i+1} = \bar{y}_i + h \sum b_j\, g(\tilde{y}_j, \check{y}_j, \tilde{u}_j), \quad \bar{y}_N = \nabla\varphi(y_N)^T \tag{42}$$

$$\check{y}_j = \bar{y}_i + h \sum_{l=1}^{s} \check{a}_{jl} g(\tilde{y}_l, \hat{y}_l, \tilde{u}_l), \quad \check{a}_{jl} = b_l - a_{jl} \tag{43}$$

$$\tilde{y}_j = y_i + h \sum_{l=1}^{j-1} a_{jl} f(\tilde{y}_l, \tilde{u}_l), \tag{44}$$

which does not require the restriction $b_j > 0$. Using this Runge-Kutta scheme, we obtain the following result:

Theorem 5.2 (Convergence of reverse mode discretization II). *Assume that f and u are ν times continuously differentiable with $\nu \le 4$. Let the integration method applied in Algorithm* 1 *be of order ν, i.e., it fulfills the conditions of Table* 2 *up to order ν. Furthermore, let the order of the interpolation polynomial taken from Table* 1 *be not less than ν. Then, the approximate solution $\bar{y}_i$, $i = 1, \ldots, N$, generated by the reverse mode of AD (Algorithm* 4*) converges at least with order $\kappa = \min\{2, \nu\}$ to the continuous solution of the adjoint Eq.* (6).

Proof: Once more, we assume throughout, that the functions f and u are sufficiently smooth on the whole domain such that convergence follows immediately from consistency. The difference between the two formulations is caused by the different values of $\bar{c}_j = \sum \bar{a}_{jl}$ and $\check{c}_j = \sum \check{a}_{jl} = 1 - \sum a_{jl} = 1 - c_j$ that need not to coincide.

For $\nu = 1$ only the condition $\sum b_j = 1$ has to be fulfilled as shown in the proof of Theorem 5.1. This equality holds since the integration scheme applied in Algorithm 1 is supposed to be of order 1 and we obtain $\kappa = 1$

For $\nu = 2$, it follows according to the proof of Theorem 5.1 that we have to check whether

$$\sum b_j c_j = \frac{1}{2} = \sum b_j \check{c}_j.$$

holds. The first equality holds because of the usual order conditions for Runge-Kutta schemes in the ODE context. The second equality is valid since we have

$$\sum b_j \check{c}_j = \sum b_j (1 - c_j) = \sum b_j - \sum b_j c_j = \frac{1}{2}.$$

Using the identity (21) for the interpolated control we find $\kappa = 2$

For $\nu = 3$, the equation $\sum b_j c_j \check{c}_j = 1/3$ must hold in analogy to (24). However, the recursion (42)–(44) yields

$$\sum b_j c_j \check{c}_j = \sum b_j c_j (1 - c_j) = \sum b_j c_j - \sum b_j c_j^2 = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}.$$

Hence, using the formulation (42)–(44) it is not possible to prove a convergence order that exceeds 2. □

Using the approximate solution $\bar{y}_i$ of the adjoint Eq. (6) of order ν, we are able to compute the desired gradient information with order ν using the adjoint representation (7) given in

Section 2. Furthermore, one has to note the following important fact: Comparing (7) with the formulas for $\bar{\mathbf{u}}$ of Algorithm 4, we obtain that the standard scalar reverse mode yields an approximation of the continuous gradient of order ν if the influence of the control on the interpolation procedure at intermediate times $t_i + c_j h$ with $0 < c_j < 1$ is completely neglected. For that purpose, it is required to passivate the interpolation procedure during the application of AD to ignore the dependence of the intermediate values on the control variables. The passivation of variables is possible for some AD-tools, e.g. for ADOL-C [18]. However, this approach to compute gradient information has not been exploited so far. Instead, usually the vector $\bar{\mathbf{u}} = (\bar{u}_1, \dots, \bar{u}_N)$ is used for the optimization since $\bar{u}$ is directly generated by the reverse mode of AD without passivation. Then the values $\bar{u}_j$ represent the exact discrete gradient information for the integration method given by Algorithm 1. To verify this fact, we compare the discrete gradient representation (11) with the computation of $\bar{\mathbf{u}}$. For the Runge-Kutta method of Algorithm 1, we have

$$\frac{\partial \psi_{i+1}(\mathbf{u})}{\partial \psi_i(\mathbf{u})} = I + h \sum_{j=1}^{s} f_y(\tilde{y}_j, \tilde{u}_j) \frac{\partial \tilde{y}_j}{\partial y_i}.$$

Hence, we obtain by inspecting the statements for $\bar{\tilde{y}}_j$ and $\bar{k}_j$ of Algorithm 4

$$\bar{y}_i = \left(I + h \sum_{j=1}^{s} f_y(\tilde{y}_j, \tilde{u}_j) \frac{\partial \tilde{y}_j}{\partial y_i} \right)^T \bar{y}_{i+1} = \left(\frac{\partial \psi_{i+1}(\mathbf{u})}{\partial \psi_i(\mathbf{u})} \right)^T \bar{y}_{i+1}.$$

Since $\bar{y}_N$ is initialized to $\nabla \varphi(y_N)^T$, one obtains the formula

$$\bar{y}_i = \left(\frac{\partial \psi_{i+1}(\mathbf{u})}{\partial \psi_i(\mathbf{u})} \right)^T \cdots \left(\frac{\partial \psi_N(\mathbf{u})}{\partial \psi_{N-1}(\mathbf{u})} \right)^T \nabla \varphi(y_N)^T.$$

Now, this derivative vector is used to compute the derivative of $\psi_i(\mathbf{u})$ with respect to $\mathbf{u}$. For that purpose, the derivative of $\psi_i(\mathbf{u})$ with respect to $\mathbf{u}$ is composed for each intermediate time $c_j h$ because of the definition of $\bar{k}_j$ and $\bar{\tilde{u}}_j$. Subsequently $\bar{\tilde{u}}_j$ is multiplied by the derivative of $\tilde{u}_j$ with respect to $\mathbf{u}$ and the result is added to the appropriate components of $\bar{\mathbf{u}}$. Therefore, Algorithm 4 yields the exact discrete derivatives (11) of the discrete cost function $\mathbf{J}$ evaluated by Algorithm 1 with respect to the control vector $\mathbf{u}$. However, since this gradient computation does not coincide with a discrete version of the adjoint representation (7) it is not possible to deduce an order of convergence for the discrete gradient information $\bar{\mathbf{u}}$ generated directly by the reverse mode of AD.

6 Numerical illustration

For the verification of the theoretical results presented in the last sections, consider the following simple test problem taken from [21]:

$$\min \frac{1}{2} \int_0^1 u(t)^2 + 2x(t)^2 dt \quad \text{s.t.} \quad \frac{dx}{dt}(t) = 0.5x(t) + u(t), \quad x(0) = 1.$$

This problem can easily be transformed into the form (1)–(3) described in Section 2 using $y_1 = x$, $y_2 = \frac{dx}{dt}$ and

$$\frac{dy_1}{dt}(t) = 0.5y_1(t) + u(t), \quad y_1(0) = 1, \tag{45}$$

$$\frac{dy_2}{dt}(t) = y_1(t)^2 + 0.5u(t)^2, \quad y_2(0) = 0. \tag{46}$$

Then the objective is given by $J(y) = \varphi(y(1)) = y_2(1)$. This optimization problem has the optimal solution [21]

$$y_1^*(t) = \frac{2e^{3t} + e^3}{e^{3t/2}(2 + e^3)}, \qquad u^*(t) = \frac{2(e^{3t} - e^3)}{e^{3t/2}(2 + e^3)},$$

$$y_2^*(t) = \frac{2e^{3t} - e^{6-3t} - 2 + e^6}{(2 + e^3)^2}.$$

To approximate y^*, we use u^* as control and the Runge-Kutta schemes

$$\text{(a)}\ A = \begin{bmatrix} 0 & 0 \\ \frac{1}{2} & 0 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \text{(b)}\ A = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix},\ b = \begin{bmatrix} \frac{1}{6} \\ \frac{2}{3} \\ \frac{1}{6} \end{bmatrix},$$

$$\text{(c)}\ A = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{3}{4} & 0 \end{bmatrix},\ b = \begin{bmatrix} \frac{2}{9} \\ \frac{1}{3} \\ \frac{4}{9} \end{bmatrix}, \quad \text{(d)}\ A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},\ b = \begin{bmatrix} \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} \end{bmatrix},$$

where scheme (a) is of order 2, schemes (b) and (c) are of order 3, and scheme (d) is of order 4. These schemes were already used for numerical studies in [21]. For scheme (a), we apply linear interpolation, for the remaining schemes cubic interpolation. The resulting L^∞ errors for the discrete state at the grid points using the step size $h = 1/N$ with $N = 10, 20, 40, 80, 160, 320$ are given in Fig. 1.

It follows that the computed errors correspond very well to the order of the applied Runge-Kutta scheme. Note, that in the context of full discretization methods, the scheme (b) may not even yield a convergent approximation for the state equation (c.f. [21]).

From the state Eqs. (45)–(46) follows that the sensitivity equations are given by

$$\frac{ds_1}{dt}(t) = 0.5s_1(t) + d(t), \quad s_1(0) = 0,$$

$$\frac{ds_2}{dt}(t) = 2x_1(t)s_1(t) + u(t)d(t), \quad s_2(0) = 0.$$

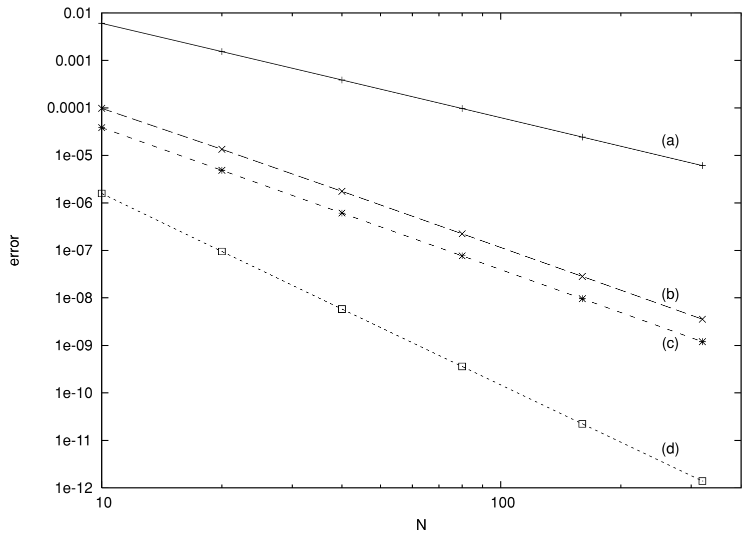

Fig. 1 Discrete state error in L^∞

For the numerical experiments, we set $d(t) = 1$, which yields the analytical solution

$$s_1(t) = 2e^{-t/2} - 2, \quad s_2(t) = \frac{4(e^{2t} - e^{3t/2} - e^{3-t} + e^{3-3t/2})}{2 + e^3}.$$

Applying the AD-tool ADOL-C, we implement Algorithm 2 to obtain the approximate solution $\dot{\mathbf{y}}$ of the sensitivity equation. The L^∞ errors for this discrete sensitivity at the grid

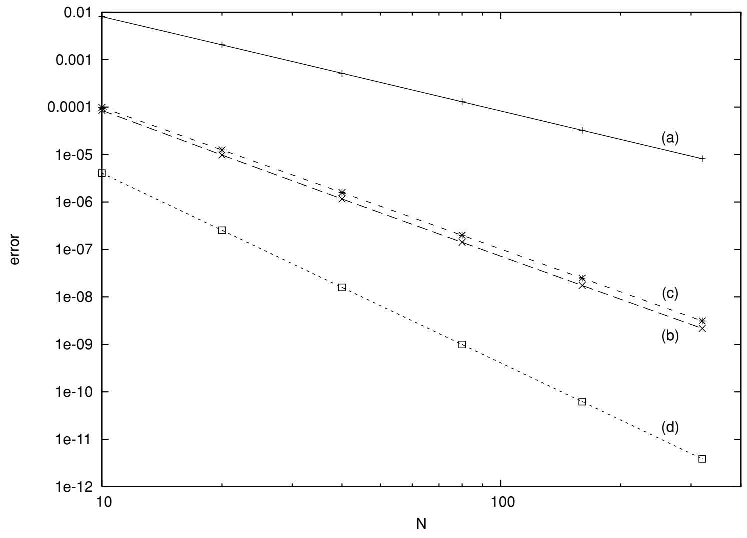

Fig. 2 Discrete sensitivity error in L^∞

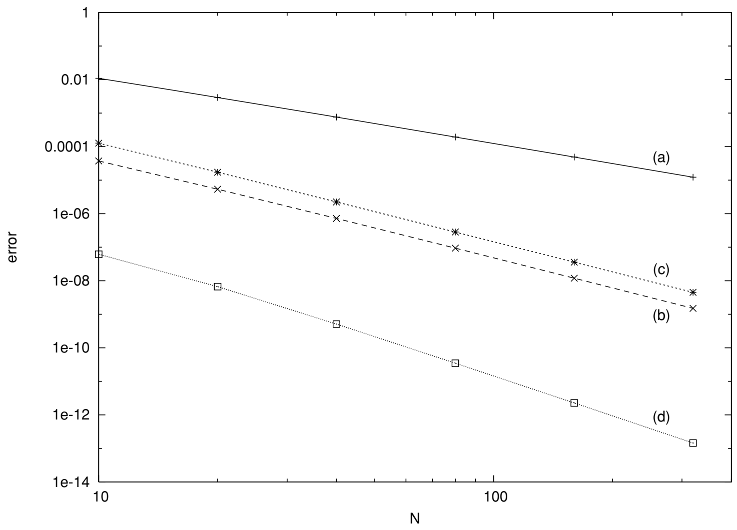

Fig. 3 Discrete adjoint error in L^∞

points are illustrated by Fig. 2. Once more, the resulting error plot illustrates perfectly the order of the applied Runge-Kutta scheme.

Finally for the state Eqs. (45)–(46), the corresponding adjoint equation is given by

$$\frac{\lambda_1}{dt}(t) = -0.5\lambda_1(t) - 2y_1(t)\lambda_2(t), \qquad \lambda(1) = 0,$$

$$\frac{\lambda_2}{dt}(t) = 0, \qquad \lambda_2(1) = 1,$$

with the analytical solution

$$\lambda_1(t) = \frac{e^{3-t} - 2e^{2t}}{e^{t/2}(2 + e^3)}, \qquad \lambda_2(t) = 1.$$

Employing the reverse mode of ADOL-C, we implement Algorithm 4 to obtain the approximate solution $\bar{\mathbf{y}}$ of the continuous adjoint equation. The L^∞ errors for this discrete adjoint at the grid points are shown in Fig. 3. This error plot illustrates the convergence rate of the adjoint Runge-Kutta methods corresponding to the schemes (a), (b), and (c) very well. Only for scheme (d), the corresponding adjoint scheme shows the expected convergence behaviour only for the finer discretizations. This may be due to the inexact information of the forward computation required by Algorithm 4. However, it is important to note that no additional conditions are required to obtain the convergence order 3 for scheme (c) which differs from the full discretization context.

7 Conclusion

This paper presents an analysis of the derivative information generated by the forward mode and reverse mode of AD for the recursive discretization in optimal control.

It has been shown that the forward mode of AD generates an approximate solution to the corresponding sensitivity equation of order ν if the Runge-Kutta method used for the integration of the state equation is already of order ν. Exploiting the approximate solution, we can compute an approximation of the continuous gradient of order ν. The standard initialization using unit vectors yields the exact discrete gradient information for the forward integration procedure as illustrated in this paper.

Furthermore, it is shown that the approximate solution of the adjoint equation obtained by the reverse mode of AD is of order ν if the integration method of the discrete version of the state equation is of order ν provided that one has $b_i > 0$. These results for the recursive discretization differ remarkably from the results obtains for the full discretization by Hager [21]. The less restrictive conditions for the coefficients of the Runge-Kutta scheme result from the fixed control for the recursive discretization whereas the control depends on the state and adjoint for the full discretization. Once more, the approximate solution presented here can be applied to compute an approximation of the continuous gradient of the same order ν. However, this gradient information is not generated directly by a black-box AD approach. The gradient information obtained by applying the standard reverse mode of AD equals again the exact discrete gradient as illustrated by this paper.

The question whether it is preferable to use the continuous gradient or the discrete gradient in an optimization procedure is still open and the answer depends certainly on the underlying problem to be solved. So far, all applications of AD known to the author provide the exact discrete gradient information. Based on the results of this paper, one may use now an approximation to the continuous gradient as alternative. Hence, it is possible to study the influences of the different derivative information on the optimization process. This subject will be investigated in the future. Furthermore, one has to examine the case when control constraints destroy the differentiability of the right-hand sides, which is ignored in this paper.

Acknowledgments The author is indebted to Roland Griesse because the joint work with him raise several questions examined in this paper. Furthermore, the author likes to thank Andreas Griewank and Christian Grossmann for many discussions on the subject.

References

1. J.T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*, SIAM, Philadelphia, 2001.
2. H.G. Bock and K.-J. Plitt, “A multiple shooting algorithm for direct solution of optimal control problems,” in *Proceedings of the 9th IFAC World Congress*, Budapest, Pergamon Press, 1984.
3. A.E. Bryson and Y. Ho, *Applied Optimal Control—Optimization, Estimation, and Control*. Hemisphere Publishing Corporation, New York, 1975.
4. R. Bulirsch, E. Nerz, H.J. Pesch, and O. von Stryk, “Combining direct and indirect methods in nonlinear optimal control: range maximization of a hang glider,” in R. Bulirsch, A. Miele, J. Stoer, K.H. Well, (eds.), in Optimal Control, Calculus of Variations, Optimal Control Theory and Numerical Methods, Birkhäuser, Basel, 1993, pp. 273–288.
5. C. Büskens, *Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustandsbeschränkungen*. Dissertation, Westfälische Wilhelms-Universität Münster, 1998.
6. C. Büskens and H. Maurer “SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. J Comput Appl Math, vol. 120, pp. 85–108, 2000.
7. J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley, New York, 1987.
8. J.-B. Caillau and J. Noailles, “Continuous optimal control sensitivity analysis with ad,” in [10], pp. 109–117.
9. D. Casanova, R.S. Sharp, M. Final, B. Christianson, and P. Symonds, “Application of automatic differentiation to race car performance optimisation,” in [10], pp. 117–124.

10. G.F. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann (eds.), *Automatic Differentiation: from Simulation to Optimization.* Springer Verlag, New York, 2001.
11. A.L. Dontchev, W. Hager, and V. Veliov, "Second-order runge-kutta approximations in control constrained optimal control," SIAM J. Numer. Anal. vol. 38, pp. 202–226, 2000.
12. M. Hinze and T. Slawig, "Adjoint gradients compared to gradients from algorithmic differentiation in instantaneous control of the navier–stokes equations. Optim. Methods Softw. vol. 18, no. 3, 299–315, 2003.
13. Yu.G. Evtushenko, "Automatic differentiation viewed from optimal control theory," in [17], pp. 25–30.
14. Yu.G. Evtushenko, "Computation of exact gradients in distributed dynamic systems," Optim. Methods Softw. vol. 9, nos. 1–3, pp. 45–75, 1998.
15. R. Griesse and A. Walther, "Evaluating gradients in optimal control—Continuous adjoints versus automatic differentiation," Journal of Optimization Theory and Applications (JOTA), vol. 122, no. 1, pp. 63–86, 2004.
16. A. Griewank, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation.* Frontiers in Appl. Math. 19, Phil., 2000.
17. A. Griewank and G. Corliss (eds.) *Automatic Differentiation of Algorithms: Theory, Implementation, and Applications.* SIAM, Philadelphia, Penn., 1991.
18. A. Griewank, D. Juedes, and J. Utke, "ADOL-C: A package for the automatic differentiation of algorithms written in C/C++," TOMS vol. 22, pp. 131–167, 1996.
19. A. Griewank and A. Walther, "Applying the checkpointing routine treeverse to discretizations of burgers' equation, Lect. Notes Comput. Sci.and Engin., 8," in *High Performance Scientific and Engineering Computing*, H.-J. Bungartz, F. Durst, and C. Zenger (eds.), Springer Berlin Heidelberg, 1999.
20. A. Griewank and A. Walther, "Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation," ACM Trans. Math. Software, vol. 26, pp. 19–45, 2000.
21. W. Hager, "Runge-Kutta methods in optimal control and the transformed adjoint system," Numer. Math. vol. 87, pp. 247–282, 2000.
22. P. Hiltmann, *Numerische Lösung von Mehrpunkt-Randwertproblemen und Aufgaben Der Optimalen Steuerung mit Steuerfunktionen über endlichdimensionalen Räumen*, Dissertation, TU München, Mathematisches Institut, Germany, 1989.
23. H. Oberle and W. Grimm, *BNDSCO—A program for the numerical solution of optimal control problems*, Report No. 515, Institute for Flight System Dynamics, Oberpfaffenhofen, German Aerospace Research Establishment DLR, 1989.
24. H.J. Pesch, "Offline and online computation of optimal trajectories in the aerospace field," in *Applied Mathematics in Aerospace Science and Engineering*, A. Miele. A. Salvetti (eds.), Plenum Press, New York, Mathematical Concepts and Methods in Science and Engineering, 1994 vol. 44, pp. 165–220.
25. A. Quarteroni, R. Sacco, and F. Saleri, *Numercial Mathematics.* Springer, New York, 2000.
26. K. Strehmel und R. Weiner, *Numerik Gewöhnlicher Differentialgleichungen*, Teubner Studienbücher: Mathematik. Teubner, Stuttgart, 1995.
27. O. von Stryk, *User's Guide for DIRCOL (Version 2.1): A Direct Collocation Method for the Numerical Solution of Optimal Control Problems.* Fachgebiet Simulation und Systemoptimierung (SIM), Technische Universität Darmstadt, 2000.