Solving parameter estimation problems with discrete adjoint exponential integrators

Ulrich Römer a,†* and Mahesh Narayananurthi b and Adrian Sandu b

aTechnische Universität Darmstadt, Darmstadt, Germany; bVirginia Tech, Blacksburg, VA, USA

(Received 20 March 2017; accepted 8 February 2018)

The solution of inverse problems in a variational setting finds best estimates of the model parameters by minimizing a cost function that penalizes the mismatch between model outputs and observations. The gradients required by the numerical optimization process are computed using adjoint models. Exponential integrators are a promising family of time discretization schemes for evolutionary partial differential equations. In order to allow the use of these discretization schemes in the context of inverse problems, adjoints of exponential integrators are required. This work derives the discrete adjoint formulae for W-type exponential propagation iterative methods of Runge–Kutta type (EPIRK-W). These methods allow arbitrary approximations of the Jacobian while maintaining the overall accuracy of the forward integration. The use of Jacobian approximation matrices that do not depend on the model state avoids the complex calculation of Hessians in the discrete adjoint formulae. The adjoint code itself is generated efficiently via algorithmic differentiation and used to solve inverse problems with the Lorenz-96 model and a model from computational magnetics. Numerical results are encouraging and indicate the suitability of exponential integrators for this class of problems.

Keywords: exponential integrators; discrete adjoints; algorithmic differentiation; 4D-Var data assimilation

AMS Subject Classification: 34H05; 34K29; 34K35

1. Introduction

Differential equations are widely used to model the dynamics of physical processes. Even if the form of the equations perfectly captures the physical effects under consideration, the predictive capability of the mathematical model depends on the availability of accurate parameter values and initial conditions. Data assimilation, roughly defined as the solution of inverse problems with models defined by differential equations, fuses information from model outputs and (noisy) physical measurements to produce better estimates of parameter values.

In this work, we consider inverse problems where the dynamics are modelled by ordinary differential equations or time-dependent partial differential equations, and where a maximum a posteriori (MAP) estimate of the parameter values is computed. Specifically, we consider four-dimensional variational (4D-Var) data assimilation problems where physical measurements are taken at different points in time, and the parameters are updated using all available measurements in the given time window and imposing the system dynamics as strong constraints.
The time dimension is discretized using a W-method [55] formulation of a class of exponential integrators that can achieve a high order of accuracy with low stage count and have been shown to be capable of dealing with many stiff problems [56,57]. To maximize the posterior distribution, gradient-based optimization techniques are employed. The gradients required in the optimization process are given by the discrete adjoint approach, that is, are obtained by algorithmic differentiation (AD) of the chosen exponential integration method.

Consider the following initial value problem in autonomous form:

\[
\frac{dy}{dt} = f(y(t), \theta), \quad y(t_0) = y_{ini}(\theta), \quad t_0 \leq t \leq t_F,
\]

where \( y(t) \in \mathbb{R}^K \) is the model state, \( \theta \in \mathbb{R}^P \) is the vector of model parameters and the right-hand side function \( f : \mathbb{R}^K \times \mathbb{R}^P \rightarrow \mathbb{R}^K \) is assumed to be sufficiently smooth.

A general inverse problem to estimate the uncertain parameters \( \theta \) is formulated as follows [20]:

\[
\theta_{\text{MAP}} = \arg \min_{\theta} \Phi(\theta) = u(y(t_0), \theta) + \int_{t_0}^{t_F} q(y(t), \theta) \, dt + w(y(t_F), \theta)
\]

subject to:

\[
\frac{dy}{dt} = f(y(t), \theta), \quad y(t_0) = y_{ini}(\theta), \quad t_0 \leq t \leq t_F,
\]

where the goal function \( \Phi : \mathbb{R}^P \rightarrow \mathbb{R} \) is expressed as a sum of an integral involving a non-linear function of the model state and parameters, \( q : \mathbb{R}^K \times \mathbb{R}^P \rightarrow \mathbb{R} \) that measures the discrepancy between model prediction and measurements, and non-linear functions of the initial and final states and the model parameters, \( u, w : \mathbb{R}^K \times \mathbb{R}^P \rightarrow \mathbb{R} \), which are chosen based on prior knowledge about the parameters. We assume that \( q, u \) and \( w \) possess the differentiability required by the optimality conditions and optimization methods used in the paper.

Two different approaches exist to solve problem (2). In the first optimize then discretize approach, the first-order optimality conditions for the continuous problem (2) are derived first, and then a discretization scheme is applied to numerically solve the resulting equations [7]. This approach affords the freedom to choose different discretization schemes for the forward and adjoint problem. In the first discretize and then optimize approach, the original problem (2) is first discretized using an appropriate time stepping method for the ODE constraint (2b), and then by approximating the integral in (2a) by a quadrature. The first-order optimality conditions for the discrete optimization problem are then derived [7,8,42]. In this paper, we study the latter approach, as it yields the exact gradient of the discrete system [45,46] and can be partially automated with the help of AD [22,38]. Other reasons for not preferring the former approach may include the difficulty in obtaining the necessary conditions for problems that are non-trivial and the need to re-derive these for each new problem [7,9], and the additional overhead involved in the derivation and implementation of the corresponding discrete system.

The paper is structured as follows. In Section 2, the exponential integrator and the discrete parameter estimation problem are introduced. Section 3 states the first-order optimality equations and the discrete adjoint of the exponential integrator. Numerical experiments are carried out in Section 4 for two different test problems. Conclusions are drawn in Section 5.

2. Formulation of the discrete parameter estimation problem

In this section, we describe the discrete parameter estimation problem based on a time discretization that uses a W-formulation [55] of a class of exponential methods.
2.1 EPIRK-W time discretization methods

Exponential Propagation Iterative Methods of Runge–Kutta type (EPIRK) [56,57] uses a very general ansatz among exponential time-stepping schemes that allow the construction of methods with low number of stages and high order. A general \(s\)-stage EPIRK method reads:

\[
Y_{n,i} = y_n + a_{i,1} \psi_{i,1}(g_{i,1} h J_n) h f(y_n) \\
+ \sum_{j=2}^{i} a_{ij} \psi_{ij}(g_{ij} h J_n) h \Delta^{(j-1)} r(y_n), \quad i = 1 \ldots s - 1, \tag{3a}
\]

\[
y_{n+1} = y_n + b_1 \psi_{x,1}(g_{x,1} h J_n) h f(y_n) + \sum_{j=2}^{s} b_j \psi_{sj}(g_{sj} h J_n) h \Delta^{(j-1)} r(y_n), \tag{3b}
\]

where \(y_n\) is the state at the current time \(t_n\), \(J_n = \partial f / \partial y|_{t_n}\) is the Jacobian matrix of the right-hand side function (1), \(Y_{n,1} \ldots Y_{n,s-1}\) are the intermediate stages and \(y_{n+1}\) is the next-step solution. The forward difference \(\Delta^{(j)} r(y_n)\) is defined recursively with the help of the remainder function \(r(y)\) as follows:

\[
r(y) := f(y) - f(y_n) - J_n (y - y_n), \\
\Delta^{(0)} r(y) := r(y), \quad Y_{0,0} := y_n, \\
\Delta^{(j)} r(Y_{n,j}) = \Delta^{(j-1)} r(Y_{n,j+1}) - \Delta^{(j-1)} r(Y_{n,j}), \quad j \geq 1, \\
\Delta^{(j)} r(y_n) = \Delta^{(j)} r(Y_{n,0}) = \Delta^{(j-1)} r(Y_{n,1}) - \Delta^{(j-1)} r(Y_{n,0}), \quad j \geq 1. \tag{4}
\]

Coefficients \(a,b,g\) and \(p\) (see Equation (5)) are computed to build methods of certain specified order by matching the Taylor series coefficient of the exact, \(y(t_n + h)\), and numerical solution, \(y_{n+1}\). Each \(\psi_{ij}(\cdot)\) matrix function is a linear combination of matrix functions \(\phi_k(\cdot)\):

\[
\psi_{ij}(Z) = \sum_{k=1}^{s} p_{i,j,k} \phi_k(Z), \tag{5}
\]

where the scalar analytic function \(\phi_k(z)\) is defined as

\[
\phi_k(z) = \int_0^1 e^{z(1-\theta)} \frac{\theta^{k-1}}{(k-1)!} d\theta = \sum_{i=0}^{\infty} \frac{z^i}{(i+k)!}, \quad k = 1, 2, \ldots, \tag{6}
\]

with the matrix function counterparts defined using a series expansion. The functions \(\phi_k(z)\) satisfy the following recurrence relation:

\[
\phi_0(z) = e^z; \quad \phi_k(z) = \frac{\phi_k(z) - 1/k!}{z}, \quad k = 1, 2, \ldots; \quad \phi_k(0) = \frac{1}{k!}. \tag{7}
\]

The major computational cost of EPIRK methods is the evaluation of matrix-function-times-vector products of the form \(\psi_{ij}(h y J_n) \cdot v\), which are linear combinations of products \(\phi_k(h y J_n) \cdot v\). For large systems, Krylov-subspace methods are the preferred choice for the evaluation of these products [26]. It has been shown that for certain kinds of problems, Krylov-based approximation of matrix exponential products converge faster than the corresponding Krylov-based linear system solves [26]. Here we use a Krylov-subspace based algorithm to compute the matrix exponential-like vector products as explained in Section 3.2.
Although classical EPIRK methods perform well on stiff problems [57], using them in the first discretize and then optimize approach will require the computation of the Hessian of the right-hand side function. In order to circumvent this, we resort to a W-method formulation of the EPIRK method [36]. W-methods first introduced in [55] for Rosenbrock-type methods admit the use of arbitrary approximations in place of the exact Jacobian, while maintaining the derived order of convergence. In [36], the authors have recently developed W-type EPIRK methods. An s-stage EPIRK-W method reads:

\[ Y_{n,i} = y_n + a_{i,1} \psi_{i,1}(g_{i,1}hT_n)hf(y_n) + \sum_{j=2}^{i} a_{i,j} \psi_{i,j}(g_{i,j}hT_n)h\Delta^{(j-1)}r(y_n), \quad i = 1 \ldots s - 1, \]  

\[ y_{n+1} = y_n + b_1 \psi_{s,1}(g_{s,1}hT_n)hf(y_n) + \sum_{j=2}^{s} b_{j} \psi_{s,j}(g_{s,j}hT_n)h\Delta^{(j-1)}r(y_n), \]  

where the exact Jacobian \( J_i \) in (3) has been replaced with an arbitrary approximation \( T_n \). The exact implication of this will become clear from the discussion that follows. To simplify notation, we define:

\[ A_{n,i,j} := a_{i,j} \psi_{i,j}(g_{i,j}hT_n), \quad B_{n,j} := b_{j} \psi_{s,j}(g_{s,j}hT_n). \]

With this notational change, one step of an EPIRK-W method reads:

\[ Y_{n,i} = y_n + A_{n,i,1}hf(y_n) + \sum_{j=2}^{i} A_{n,j,i} h\Delta^{(j-1)}r(y_n), \quad i = 1 \ldots s - 1, \]  

\[ y_{n+1} = y_n + B_{n,1}hf(y_n) + \sum_{j=2}^{s} B_{n,j} h\Delta^{(j-1)}r(y_n). \]

### 2.2 The discrete formulation of the parameter estimation problem

In the first discretize and then optimize approach to solve (2), we partition the time interval into \( N \) subintervals with nodes \( t_0, t_1, \ldots, t_N = t_f \). The constraint ODE (2b) is replaced by its EPIRK-W discretization (9) on each subinterval \( [t_n, t_{n+1}] \). The integral term of the goal function (2a) is also discretized by a quadrature rule evaluated on these time nodes. The discrete version of the parameter estimation problem (2) reads:

\[ \theta_{MAP} = \arg \min_{\theta} \tilde{\Phi}(\theta) = \sum_{k=0}^{N} q_k(y_k, \theta) \]  

subject to: \( Y_{n,i} = y_n + A_{n,i,1}hf(y_n, \theta) \)

\[ + \sum_{j=2}^{i} A_{n,j,i} h\Delta^{(j-1)}r(y_n, \theta), \quad i = 1 \ldots s - 1, \]  

\[ y_{n+1} = y_n + B_{n,1}hf(y_n, \theta) \]

\[ + \sum_{j=2}^{s} B_{n,j} h\Delta^{(j-1)}r(y_n, \theta), \quad 0 \leq n \leq N - 1, \]
The definition of the discrete cost function (10a) includes the initial term \( u(y_{\text{ini}}, \theta) \) in \( q_0(y_{\text{ini}}, \theta) \) and the final term \( w(y_N, \theta) \) in \( q_N(y_N, \theta) \).

We proceed by describing a special case of the optimization problem (10) that will also be used in the numerical examples of Section 4.

### 2.3 4D-VAR data assimilation

The four-dimensional variational data assimilation problem is a special case of (10). Adopting the notation of [54], let \( \theta_{\text{true}} \) denote the true but unknown parameter vector and \( \theta_B \) its prior, which is also referred to as background. We assume that \( \mathcal{E}_B = \theta_B - \theta_{\text{true}} \) is normally distributed with zero mean and covariance matrix \( B \). Let measurements \( z_i \) be given at time points \( t_{ji}, \quad i = 1, \ldots, N_{\text{obs}} \), which are assumed to be a subset of the nodes of the time interval, for simplicity.

Let \( \mathcal{H} \) denote an observation operator that maps a state \( y_{ji} \), at time \( t_{ji} \), to the space of observations. The measurement error is given as \( \mathcal{E}_{\text{obs}}^i = \mathcal{H}(y_{\text{true},ji}) - z_i \), where \( y_{\text{true},ji} \) represents the state associated to \( \theta_{\text{true}} \) at time \( t_{ji} \). We assume a normal distribution for \( \mathcal{E}_{\text{obs}}^i \), with mean zero and covariance matrix \( R^i \). Assuming no model errors, the 4D-Var cost function and gradient are defined as follows:

\[
\tilde{\Phi}(\theta) = \frac{1}{2} (\theta - \theta_B)^T B^{-1} (\theta - \theta_B) + \frac{1}{2} \sum_{i=1}^{N_{\text{obs}}} (\mathcal{H}(y_{ji}) - z_i)^T R^{-1}_i (\mathcal{H}(y_{ji}) - z_i),
\]

(11)

\[
\nabla_{\theta} \tilde{\Phi}(\theta) = B^{-1} (\theta - \theta_B) + \sum_{i=1}^{N_{\text{obs}}} \left( \frac{dy_{ji}}{d\theta} \right)^T \left( \frac{d\mathcal{H}}{dy_{ji}} \right)^T R^{-1}_i (\mathcal{H}(y_{ji}) - z_i).
\]

(12)

The value \( \theta_{\text{map}} \) minimizing (10) with the goal function given by (11) represents the maximum-likelihood estimate of \( \theta_{\text{true}} \). In a data assimilation context, \( \theta_{\text{map}} \) is also referred to as analysis.

### 3. First-order optimality conditions and the discrete EPIRK-W adjoint

This section is devoted to the first-order optimality system associated with the discrete inverse problem (10), which involves discrete adjoints for the EPIRK-W method. Various authors have addressed discrete adjoints in optimal control, constrained by ordinary differential equations. In [23], discrete adjoints for Runge–Kutta methods were derived together with control specific order conditions. A similar approach was taken in [29] for Runge–Kutta W-methods. In both works, the discrete adjoint method could be reformulated again as a Runge–Kutta method and this was exploited in the subsequent error analysis. A general discussion on the use of discrete time integration adjoints in the solution of inverse problems is presented in [47]. Discrete adjoints of Rosenbrock methods are formulated in [15,16]. Theoretical properties of general discrete Runge–Kutta adjoints are proved in [45], and theoretical properties of discrete adjoints of linear multistep methods in [46]. Discrete adjoints of variable-step integrators are studied in [2]. Efficient implementations of Runge–Kutta adjoints are provided by software developed by the authors [1,5,35,48,53,58]. Novel symplectic integrators are constructed and their relation to direct methods in optimal control theory is studied in [11,13,51].

We derive the discrete adjoint of EPIRK-W in Appendices A.2.1–A.2.4. One can notice that the discrete adjoint equations do not assume the structure of an exponential method. Discrete adjoints for another closely related class of exponential methods can be found in [44].
3.1 First-order optimality system

The first-order optimality system corresponding to (10) is obtained by following the derivation given in Appendices A.2.1–A.2.4, and setting the derivatives of the discrete Lagrangian to zero. It consists of:

(A) the original EPIRK-W method, i.e. (10b), (10c) and (10d),

(B) the discrete adjoint system for the external stages,

\[ \frac{\partial q_n(y_n, \theta)}{\partial y_n} + \lambda_{n+1} \cdot \left( I + B_{n,1} h J(y_n, \theta) + \sum_{j=2}^{s} B_{n,j} h \sum_{\ell=0}^{j-2} C_{\ell,j} [T_n - J(y_n, \theta)] \right) \]

\[ + \sum_{m=1}^{s-1} A_{n,m} \cdot \left( I + A_{n,m,1} h J(y_n, \theta) + \sum_{j=2}^{m} A_{n,m,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} [T_n - J(y_n, \theta)] \right), \]  

(13a)

and internal stages,

\[ A_{n,l}^T = \lambda_{n+1}^T \cdot \left\{ \sum_{j=2}^{s} B_{n,j} h C_{j-i-1,j} [J(Y_{n,i}, \theta) - T_n] \right\} \]

\[ + \sum_{m=1}^{s-1} A_{n,m} \cdot \left\{ \sum_{j=2}^{m} A_{n,m,j} h C_{j-i-1,j} [J(Y_{n,i}, \theta) - T_n] \right\}, \]  

(13b)

(C) and finally, the optimality condition,

\[ 0 = \frac{d\Phi}{d\theta}(\theta) \]

\[ = \lambda_0^T \cdot \frac{\partial y_{ini}(\theta)}{\partial \theta} + \sum_{k=1}^{N} \frac{\partial q_k(y_k, \theta)}{\partial \theta} \]

\[ + \sum_{k=0}^{N-1} \lambda_{k+1}^T \left( B_{k,1} h \frac{\partial f(y_k, \theta)}{\partial \theta} + \sum_{j=2}^{s} B_{k,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} \frac{\partial r(Y_{k,j-1}, \theta)}{\partial \theta} \right) \]

\[ + \sum_{m=1}^{N-1} \sum_{k=0}^{s-1} A_{k,m}^T \left( A_{k,m,1} h \frac{\partial f(y_k, \theta)}{\partial \theta} + \sum_{j=2}^{m} A_{k,m,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} \frac{\partial r(Y_{k,j-1}, \theta)}{\partial \theta} \right), \]  

(14)

where the coefficients \( C_{\ell,j} \) are defined as \( C_{\ell,j} = (-1)^{j-1} \binom{j-1}{\ell} \); \( \lambda_n \) and \( A_{n,m} \) are the Lagrange multipliers for the external and internal stages, respectively. It should be emphasized that (13) is solved from \( t_F \) to \( t_0 \) in reversed time direction.

Although the first-order optimality system can itself be used to solve the discrete parameter estimation problem (10), we, instead, use AD to derive the adjoint of the time integration scheme (8). In the experiments in Section 4, we use the adjoint time integrator in conjunction with its forward counterpart to solve the parameter estimation problem for two different examples. In a future work, we intend to use the optimality system derived in Appendices A.2.1–A.2.4, to analyse the software-generated adjoint code.
Remark 1 In parameter estimation problem (2), the goal function and ODE constraint depend on a vector of unknown parameters \( \theta \in \mathbb{R}^p \) that need to be inferred from measurements. In the general case, computation of gradient of the cost function involves complicated derivatives with respect to \( \theta \) as shown in (14). This can be circumvented by converting it into an initial state estimation problem. To this end, the ODE system can be extended with the addition of parameters \( \tau(t) = \theta \) as formal variables to the state vector as follows:

\[
\tilde{y}' = \begin{bmatrix} y' \\ \tau \end{bmatrix} = \begin{bmatrix} f(y, \tau) \\ 0 \end{bmatrix} = \tilde{f}(\tilde{y}), \quad \tilde{y}(t_0) = \begin{bmatrix} y(t_0) \\ \tau(t_0) \end{bmatrix} = \begin{bmatrix} y_{\text{ini}} \\ \theta \end{bmatrix} = \tilde{y}_{\text{ini}}. \tag{15}
\]

3.2 Algorithmic differentiation

In the previous section, we presented the first-order optimality conditions for the discretized optimization problem in (10). To derive the adjoint we use AD [38], a technique that generates code to compute the sensitivities of an output with respect to an input for a given program, to compute the gradient of the Lagrangian. The site www.autodiff.org lists numerous tools that perform AD for programs written in two dozen languages. Excellent resources for the theory and implementation techniques of AD include [22,37].

Two different modes of AD exist for first-order sensitivities – forward and reverse. The forward mode AD produces the tangent linear model of the input program, whereas the reverse mode produces the adjoint model. Mathematically, the forward mode computes Jacobian-vector products for the given program, and the adjoint mode Jacobian-transposed vector products, where the vector is some seed direction. Depending on the desire to compute a combination of the rows or columns of the Jacobian, one uses the forward or adjoint mode, respectively. For a detailed discussion of cost in terms of operations and memory for each of the two modes, we refer to [22]. It suffices to say that for the computation of the gradient of the Lagrangian, where the number of inputs far exceeds that of the outputs, the reverse mode is more economical in terms of operation count.

A drawback of the reverse mode is the necessity to store the entire forward program trajectory that will be required while ‘reversing’ the program to compute the sensitivities. If the given program has a loop structure, then all the variables that affect the output and change from one iteration to the next have to be stored in a stack like data structure to be used during reversal of the loop in the program. As a result, storage requirements will grow with long running loop-based programs. One technique to alleviate the situation is to use checkpointing [22], where periodically the stack is written to a file before being flushed for reuse. The checkpoint files are read in the reverse order while ‘reversing’ the computations to calculate sensitivities. Techniques are available to balance the costs of checkpointing and recomputation [21].

In this work, we implemented a single-step EPIRK-W integrator in Fortran making use of the following major operations:

- Linear combinations of vectors (axpy) are carried out using the built-in array operation syntax.
- Matrix vector products (gemv) are implemented as an explicit subroutine call instead of using the Fortran intrinsic (MATMUL).
- The right-hand side \( f(y) \) is an external callback function provided by the caller.
- Computation of \( \psi_{i,j} \) function products is rather involved and is explained in detail below.

The \( \psi_{i,j} \) products can be written as a linear combination of individual \( \varphi_k \) products, where we approximate each \( \varphi_k \) product in the Krylov-subspace [56] as \( \varphi_k(h \gamma T_n)b \approx \|b\| V \varphi_k(h \gamma H)e_1 \), where \( V \) is the matrix containing orthogonal basis vectors of the Krylov-subspace \( K_m = \text{span}\{b, T_1b, T_2b, \ldots, T_{m-1}b\} \) and \( H \) is the upper-Hessenberg matrix resulting from an Arnoldi
process. Following [52, Theorem1], \( \phi_k(h\gamma H)e_1 \) products are computed by constructing an augmented matrix and exponentiating it using Expokit [52].

The adjoint of a single-step EPIRK-W integrator is obtained with the help of Tapenade [25]. We differentiate through both \texttt{axpy} and \texttt{gemv} operations, but use black-box routines (http://www-sop.inria.fr/tropics/tapenade/faq.html#Libs1) for both \texttt{f(y)} and \( \psi_{ij}(h\gamma T_n)v \). The adjoint of \( f(y) \) is an external callback provided by the caller. The adjoint implementation of \( \psi_{ij}(h\gamma T_n)v \) is based on the following result.

**Lemma 3.1 (Adjoint of matrix-function-times-vector operation)** Let \( y_n \) be the independent and \( y_{n+1} \) the dependent of a single-step EPIRK-W method (8). From Equations (5) and (6), we have:

\[
\psi_{ij}(h\gamma T_n) = \sum_{k=1}^{s} p_{ij,k} \phi_k(h\gamma T_n), \quad \phi_k(z) = \sum_{i=0}^{\infty} \frac{z^i}{(i+k)!},
\]

where \( T_n \) is an arbitrary approximate Jacobian that does not depend on \( y_n \). The adjoint of the operation:

\[
\alpha \leftarrow \psi_{ij}(h\gamma T_n) \cdot v
\]

where \( v \) is a vector independent of \( y_n \), reads:

\[
\bar{v} \leftarrow \psi_{ij}(h\gamma T_n^T) \cdot \bar{\alpha},
\]

where the adjoint variables are \( \bar{v} = \bar{y}_{n+1}^{T} \frac{\partial y_{n+1}}{\partial y} \) and \( \bar{\alpha} = \bar{y}_{n+1}^{T} \frac{\partial y_{n+1}}{\partial \alpha} \).

**Proof** Given an arbitrary matrix \( X \) and a vector \( v \) that is a function of the independent \( y_n \), the adjoint of the product \( z \leftarrow Xv \) is the operation \( \bar{v} \leftarrow X^T \bar{z} \). Consequently, since \( T_n \) does not depend on \( y_n \), we have:

\[
\bar{v} \leftarrow \psi_{ij}(h\gamma T_n)^T \cdot \bar{\alpha}.
\]

It remains to show that

\[
\psi_{ij}(h\gamma T_n)^T = \psi_{ij}(h\gamma T_n^T).
\]

We are given

\[
\psi_{ij}(h\gamma T_n) = \sum_{k=1}^{s} p_{ij,k} \phi_k(h\gamma T_n).
\]

Taking the transpose then yields

\[
\psi_{ij}(h\gamma T_n)^T = \sum_{k=1}^{s} p_{ij,k} \phi_k(h\gamma T_n)^T.
\]

Since

\[
\phi_k(z) = \sum_{i=0}^{\infty} \frac{z^i}{(i+k)!},
\]

we have

\[
\phi_k(h\gamma T_n) = \sum_{i=0}^{\infty} \frac{(h\gamma T_n)^i}{(i+k)!} = \sum_{i=0}^{\infty} \frac{(h\gamma)^i(T_n)^i}{(i+k)!}.
\]
Taking the transpose then yields
\[
\phi_k(hyT_n)^T = \sum_{i=0}^{\infty} \frac{(hy)^i(T_n)^i}{(i+k)!} = \sum_{i=0}^{\infty} \frac{(hy)^i(T_n^T)^i}{(i+k)!} = \phi_k(hyT_n^T).
\] (19)

From Equations (17) and (19), it can be inferred that
\[
\psi_{ij}(hyT_n)^T = \psi_{ij}(hyT_n^T),
\] (20)
thus proving the result.

**Remark 2** If the matrix argument is the exact Jacobian $J_n$, or any matrix that depends on $y_n$, then the adjoint of the operation
\[
\alpha \leftarrow \psi_{ij}(hyJ_n) \cdot v
\]
involves the additional computation of the Hessian tensor $\partial J_n/\partial y_n$. This is the main motivation for using a W-method formulation of EPIRK in our work: the assumption that matrices $T_n$ are independent of $y_n$ allows the adjoint operations to be implemented in accordance with Lemma 3.1, while the W-property allows maintaining the overall accuracy of the forward simulation.

In the following section, we consider two test problems where a 4D-Var problem is solved for parameter/initial state estimation. The discrete adjoint EPIRK-W integrator is used to compute the gradients needed in optimization.

### 4. Applications

In this section, two different numerical examples will be given. As a first test case, we consider the Lorenz-96 model, which is a discrete non-linear example, where the parameter is the initial condition. As a second test case, an inductor will be considered. The original equation is obtained by applying the finite element method to the magnetostatic partial differential equations. Then data assimilation is carried out for parameters characterizing the non-linear magnetic material in the iron core. For both examples, the limited memory variant of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [12] callable from *SciPy* [27] is used for optimization. We use the third-order EPIRK-W method derived in [36, Figure 2] in our experiments. The integrator and its adjoint is made callable from *Python* using *f2py* [40] and checkpoints are stored in-memory.

#### 4.1 Lorenz-96 model

The model considered here was originally proposed in [33] and is often used in a data assimilation context [4,14,18,30,31,39,54]. A detailed description of the setup we use is given in [54] and is summarized below to enhance readability. The deterministic model is given by the set of equations:
\[
\frac{dy^j}{dt} = -y^{j-1}(y^{j-2} - y^{j+1} - y^j) + 8, \quad j = 1, \ldots, K,
\]
\[
y^j = y^{K+j} \quad \forall j,
\] (21)
where $K = 40$ and $y = (y^1, \ldots, y^K)^T$. Let $t_0 = 0$, $t_F = 0.3$ and the time interval be $\Delta t = 0.015$ time units.
We ran convergence test for the Lorenz-96 model of both the forward and the adjoint integrator with the Jacobian initialized to a random matrix at the beginning, and using a fixed random seed for the adjoint integrator. Reference solutions for both forward and adjoint problems were computed using the MATLODE [5] ERK forward and adjoint integrators, respectively, with both absolute and relative tolerance set to $1.0 \times 10^{-12}$.

As Figure 1 shows, both forward and adjoint EPIRK-W integrators achieve full third-order convergence in the Euclidean norm, as desired. It should be noted, however, that either the exact Jacobian or a very good approximation of it may be needed for better stability of the integrators (see [24, Sec IV.7, IV.11], [55]). In [36], the convergence behaviour of the forward EPIRK-W integrator was treated in detail. A complete treatment of the convergence behaviour of the adjoint EPIRK-W integrator will be considered in a future work. We continue to describe the example in which we apply these integrators.

In this example, $\theta$ parametrizes the initial condition as $y_{ini} = \theta$, whereas $f$ is a function of $y$, solely. Following [54], the reference $\theta_{true}$ is obtained by integrating $y^i(-10 \Delta t) = 1 + 0.1 \mod(j, 5)$ in time until $t = 0$ (time units). The observation operator $\mathcal{H}$ is defined as

$$\mathcal{H}(y) = \left( y^1, y^3, y^5, \ldots, y^{19}, y^{21}, y^{22}, \ldots, y^{40}, \sum_{j=1}^{10} y^j, \sum_{j=1}^{20} y^j, \sum_{j=21}^{40} y^j, \sum_{j=31}^{40} y^j \right)^T.$$ 

We model the background and observation standard deviation as

$$\sigma_b = 0.03 \theta_{true},$$

$$(\sigma_{obs})_k = c_{obs} \left( \sum_{i=1}^{N_{obs}} \mathcal{H}_k(y^{true}_j) \right) / N_{obs}, \quad k = 1, \ldots, 34,$$

where $y^{true}_j$ refers to the solution at step $j_i = i100$, $i = 1, \ldots, N_{obs} = 10$, with initial condition $\theta_{true}$. In the first part of the experiment, we set $c_{obs} = 0.005$, i.e. we assume a standard deviation of 0.5% of the averaged measured values of the true solution. These standard deviations are used to define the background and observation covariance matrices as

$$B = \alpha I + (1 - \alpha) \sigma_b \otimes \sigma_b e^{-D^2/L^2},$$

$$R_k = R = \text{diag}_{k=1,\ldots,34}((\sigma_{obs})^2_k),$$

with $\alpha = 0.1$, $L = 4$, $(D)_{ij} = \min(|i - j|, K - |i - j|)$. We emphasize that $R$ is modelled to be independent of time in this case. Also, in this section, $I$ refers to the $K \times K$ identity matrix.
The background prior $\theta_b$ is generated as a pseudo-random realization according to the normal distribution of $E_b$ with mean value $\theta_{\text{true}}$ and covariance $B$. Correspondingly, the measurement $z_i$ is generated as a pseudo-random realization around the observation reference according to the normal distribution of $E_{\text{obs}}$, with mean value $\mathcal{H}(y_{ji}(\theta_{\text{true}}))$ and covariance $R$.

We perform data assimilation by solving (10) using the goal function and gradient given in (11) and (12), respectively. The discrete adjoints are computed using AD as outlined in Section 3.2 and supplied to the optimization routine, with a random matrix $T_n = T$ used in the forward and adjoint integrators. The step size used is $\Delta t = 0.0003$ time units. As can be seen from Figure 2, the cost function falls sharply after a few iterations. Nevertheless, about 125 iterations are needed for the gradient to be under $10^{-3}$. This slow convergence is due to the rather small size of the measurement standard deviation $\sigma_{\text{obs}}$. If we repeat the simulation with $c_{\text{obs}} = 0.1$ the convergence is much faster, as expected (see Figure 3). For this setting, $c_{\text{obs}} = 0.1$, Figure 4 depicts the distance between the initial condition at the current iteration and the true initial condition in the Euclidean norm.

### 4.2 Parameter estimation in magnetic field problems

Before describing the data assimilation problem for the inductor, we give some background of the underlying partial differential equations. The magnetquasistatic problem in a bounded computational domain $\Omega$ reads as

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}, \quad \text{in } \Omega,$$

(24a)
Figure 4. Distance between estimated initial condition and true initial condition in the Euclidean norm with $c_{\text{obs}} = 0.1$. Relative error amounts to approximately 5% after optimization.

\[ \nabla \times \vec{H} = \vec{J} + \kappa \vec{E}, \quad \text{in } \Omega, \quad (24b) \]

\[ \vec{B} \cdot \vec{n} = 0, \quad \text{on } \partial \Omega, \quad (24c) \]

and is widely used in the simulation of electrical machines, magnets and transformers. In (24), $\vec{E}$ denotes the electric field, $\kappa$ the conductivity, $\vec{J}$ the imposed source current density and $\vec{n}$ the outer unit normal. Also, the magnetic flux density $\vec{B}$ and the magnet field strength $\vec{H}$ are related through the non-linear material relation $\vec{H} = \nu(|\vec{B}|)\vec{B}$. Here, we restrict ourselves to parametric non-linearities of the type

\[ \nu(|\vec{B}|) = \frac{1}{2|\vec{B}|} \left( \tanh \left( \frac{|\vec{B}|}{\theta_1} \right) + \tanh \left( \frac{|\vec{B}|}{\theta_2} \right)^{30} \right) \left( \theta_3 + \theta_4 |\vec{B}| \right), \quad (25) \]

see [43], where $\nu(0)$ is obtained by taking the limit $|\vec{B}| \to 0$ in the above equation. The parameters $\theta$ of the non-linearity are often deduced from measurements. Equation (24) is discretized in space by applying the finite element method. To this end, as a first step, from (24) we derive the second-order problem

\[ \kappa \frac{\partial \vec{A}}{\partial t} + \nabla \times (\nu(\theta, |\nabla \times \vec{A}|) \nabla \times \vec{A}) = \vec{J}, \quad \text{in } \Omega, \quad (26a) \]

\[ \vec{A} \times \vec{n} = 0, \quad \text{on } \partial \Omega, \quad (26b) \]

with the magnetic vector potential $\vec{A}$ such that $\vec{B} = \nabla \times \vec{A}$ and the initial condition $\vec{A}(0) = \vec{A}_0$ in $\Omega$ [10]. We further simplify the problem by considering a two-dimensional domain. By inserting $\vec{A} = (0, 0, u)^T$ in (26), where the third coordinate represents the direction perpendicular to the two-dimensional domain, we obtain

\[ \kappa \frac{\partial u}{\partial t} - \nabla \cdot (\nu(\theta, |\nabla u|) \nabla u) = f, \quad \text{in } \Omega, \quad (27a) \]

\[ u = 0, \quad \text{on } \partial \Omega. \quad (27b) \]

Difficulties arise in the solution of problem (27), as $\kappa$ vanishes in non-conducting subregions of $\Omega$. A remedy consists in using $\kappa_{\text{reg}} = \max(\kappa, \kappa_{\text{air}})$, with artificial conductivity $\kappa_{\text{air}} > 0$, in (27)
Figure 5. Left: Inductor configuration with airgap, coil (red) and iron core (blue). Due to symmetry only, the right part is used for the simulation. The computational domain (air in gray) with a triangular mesh used in the simulation is depicted. Right: Magnetic flux density in the iron core.

instead. The error of this approximation as a function of $\kappa_{\text{air}}$ is well understood [6]. We assume that $\kappa$ is replaced with $\kappa_{\text{reg}}$ from now on and omit the subscript for simplicity.

To state the weak formulation of (27), we introduce the usual Sobolev space

$$V = \{ u \in L^2(\Omega) \mid \nabla u \in (L^2(\Omega))^2 \text{ and } u|_{\partial\Omega} = 0 \}. \quad (28)$$

Then we seek $u : [t_0, t_F] \rightarrow V$ such that

$$\int_{\Omega} \kappa \frac{\partial u}{\partial t} v \, dx + \int_{\Omega} v(\theta, |\nabla u|) \nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx, \quad \forall v \in V. \quad (29)$$

The finite element space $V_h \subset V$ consists of piecewise linear polynomials on a quasi-uniform triangulation of $\Omega$. This results in an ordinary differential equation

$$M y'_F + K(y_F, \theta)y_F = f_F, \quad (30)$$

where $M, K, y_F, f_F$ refer to the mass and stiffness matrix and to the vector of degrees of freedom and the source vector, respectively. Problem (30) can be recast as

$$y'_F = f(y_F, \theta), \quad y_F = y_{\text{ini}}, \quad (31)$$

where $f(y_F, \theta) = M^{-1}(f_F - K(y_F, \theta)y_F)$. From (31), it is clearly visible that the parameter dependence is bound to $f$ solely.

The example considered here is an inductor, introduced in [34], which is depicted in Figure 5. The observed (scalar) quantity is the flux linkage defined as

$$\mathcal{H} = \frac{N_{\text{turns}}}{|\Omega_{\text{coil}}|} \int_{\Omega_{\text{coil}}} \frac{\partial u}{\partial t} \, dx, \quad (32)$$

where $N_{\text{turns}}$, $\Omega_{\text{coil}}$ and $|\Omega_{\text{coil}}|$ refer to the number of turns in the coil, the coil domain (red in Figure 5) and its area, respectively.
The parameter estimation problem is set up as follows. We assume that the true parameters of the non-linearity (25) are given by

\[ \theta_{\text{true}} = [2.88e+3, 5.99e+7, 4.35e+4, 1.89]. \]

Here, the background and measurement covariance are modelled as

\[ B = \text{diag}(0.09(\theta_{\text{true}})^2) \],

\[ R_i = \sigma_{\text{obs}}^2, \]

i.e. both the parameters and the measurements are assumed to be uncorrelated. We consider a sinusoidal current excitation \( I(t) = I_0 \sin(4\pi t) \) on the time interval \([0, 0.5]\) s. Measurements are taken at time steps \( j_i = 10i, \) for \( i = 1, \ldots, 5 = N_{\text{obs}}\). The measurement standard deviation is modelled to be 0.1% of the effective measurement value:

\[ \sigma_{\text{obs}} = 0.001 \sqrt{\frac{1}{N_{\text{obs}}} \sum_{i=1}^{N_{\text{obs}}} \mathcal{H}(y_{ji}(\theta_{\text{true}}))^2}. \]

Then the background prior \( \theta_{b} \) is generated as a pseudo-random realization according to a normal distribution with mean \( \theta_{\text{true}} \) and covariance \( B \). In a similar manner, measurements \( z_i \) are generated as pseudo-random realizations, according to a normal distribution with covariance \( R_i \) around the true trajectory.

With the goal function and gradient as in Equations (11) and (12), we apply the EPRIK-W method to solve (31). This problem is very stiff and it warrants the use of the exact Jacobian, i.e. \( T_n = J_n \), to enhance stability. The optimization is carried out in the rescaled parameters \( \tilde{\theta}_i \) defined by \( \theta_i = (\theta_{\text{true}})_i (1 + \tilde{\theta}_i) \) and the bounds for the new variables are chosen as \( \tilde{\theta}_i \in [-0.2, 0.2] \). For the simulation, a step size of \( \Delta t = 0.01 \) s is employed. Using symmetry only the right half of the configuration is used in the computations. A current of \( I_0 = 150 \) A per turn, with \( N_{\text{turns}} = 66 \) turns in total, is imposed to the coil. The associated flux distribution in the core is depicted in Figure 5 on the right. Meshing and finite element analysis are carried out with Gmsh [19] and FEniCS [3,32], respectively. To reduce the stiffness of the problem, we set \( \kappa_{\text{air}} = 1.0e+6 \text{Sm}^{-1} \), which has a negligible effect on the flux linkage.

Figure 6 shows the cost function and the gradient over the iteration steps of the optimization, whereas in Figure 7 we plot the difference of parameter \( \theta_4 \) and its true value, during optimization. The error in \( \theta_4 \) decays quickly. It should be noted that \( \theta_4 \) models the saturated range of the non-linearity [43], which contributes significantly to the shape of the waveform depicted in Figure 8.

![Figure 6](image-url) Cost function and projected gradient plotted against iterations of optimization algorithm for a time step size of \( \Delta t = 0.01 \) s.
Figure 7. Absolute error in predicted $\theta_4$ from true $\theta_4^{\text{true}}$. Errors for other components are as follows: $|\theta_1 - \theta_1^{\text{true}}| = 6.496$, $|\theta_2 - \theta_2^{\text{true}}| = 1461.659$, $|\theta_3 - \theta_3^{\text{true}}| = 84.818$, and in a relative sense $2e - 3$, $2.4e - 5$, $2.0e - 3$, respectively.

Figure 8. Measurements and simulation, before and after parameter estimation, of flux linkage over time.

Table 1. Measurements compared to simulation with background parameter and simulation after data assimilation in Wbs$^{-1}$.

| Measurement | Background | Optimum |
|-------------|------------|----------|
| $z_1(t = 0.1)_s$ | 7.656 | 7.450 | 7.660 |
| $z_2(t = 0.2)_s$ | -0.556 | -0.377 | -0.554 |
| $z_3(t = 0.3)_s$ | -9.565 | -9.414 | -9.571 |
| $z_4(t = 0.4)_s$ | -6.092 | -4.004 | -6.092 |
| $z_5(t = 0.5)_s$ | 4.472 | 4.098 | 4.470 |

As this shape is modified during data assimilation to fit the measurements, $\theta_4$ plays a key role here. Figure 8 depicts the flux linkage over the time interval for the prior and the parameters after assimilation together with the measurements. The trajectory after data assimilation is in better accordance with the measurement data. Finally, Table 1 summarizes the measurement values as well as the numerical predictions before and after data assimilation.

5. Conclusion

This work derives the discrete adjoint formulae for a general class of exponential integrators, EPIRK-W. The choice of methods with the W-property allows to use arbitrary approximations of the Jacobian as arguments of the matrix functions while maintaining the overall accuracy of the
forward integration. The use of matrices that do not depend on the model state avoids the complex calculation of Hessians (derivatives of the Jacobian with respect to state variables) in the discrete adjoint formulae. The simplified discrete adjoint can be computed via AD and then supplied to optimization routines for solving the inverse problem at hand. We were able to empirically verify that the convergence order of the discrete adjoint of the EPIRK-W method matches the order of convergence of the forward EPIRK-W integrator. The methodology was applied to estimate the initial condition of the Lorenz-96 model from synthetic measurement data. The methodology was also applied to estimate magnetic material parameters for computational magnetics problems, where a system of stiff ordinary differential equations is obtained by a method of lines approach. Exponential integrators are a promising alternative for this class of problems and this topic merits further investigation.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The work of M. Narayanamurthi and A. Sandu was supported in part by awards NSF Division of Mathematical Sciences DMS-1419003, NSF Division of Computing and Communication Foundations CCF-1613905, Air Force Office of Scientific Research (AFOSR) DDDAS 15RT1037 and by the Computational Science Laboratory at Virginia Tech. Part of the work was performed during U. Römer’s visit at Virginia Tech.

ORCID

Ulrich Römer  
http://orcid.org/0000-0002-1277-7509

Mahesh Narayanamurthi  
http://orcid.org/0000-0002-4410-4499

References

[1] M. Alexe and A. Sandu, *Forward and adjoint sensitivity analysis with continuous explicit Runge–Kutta schemes*, Appl. Math. Comput. 208 (2009), pp. 328–34. Available at https://dx.doi.org/10.1016/j.amc.2008.11.035.

[2] M. Alexe and A. Sandu, *On the discrete adjoints of variable step time integrators*, J. Comput. Appl. Math. 233 (2009), pp. 1005–1020. Available at https://dx.doi.org/10.1016/j.cam.2009.08.109.

[3] M.S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, and G.N. Wells, *The FEniCS project version 1.5*, Arch. Numer. Soft. 3 (2015), pp. 9–23.

[4] J.L. Anderson, *Spatially and temporally varying adaptive covariance inflation for ensemble filters*, Tellus A 61 (2009), pp. 72–83.

[5] A. Augustine and A. Sandu, *MATLODE: A MATLAB ODE solver and sensitivity analysis toolbox*, ACM TOMS in preparation, 2018.

[6] F. Bachinger, U. Langer, and J. Schöberl, *Numerical analysis of nonlinear multiharmonic eddy current problems*, Numer. Math. 100 (2005), pp. 593–616.

[7] J.T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*, SIAM, Philadelphia, 2010.

[8] J.T. Betts and S.L. Campbell, *Discretize then optimize*, Proceedings of the SIAM Conference on Mathematics for Industry, Challenges and Frontiers, Toronto, Ontario, October 13–15, 2003.

[9] J.T. Betts, S.L. Campbell, and A. Engelson, *Direct Transcription Solution of Inequality Constrained Optimal Control Problems*, in American Control Conference, 2004. Proceedings of the 2004, Vol. 2, IEEE, 2004, pp. 1622–1626.

[10] O. Bíró, *Edge element formulations of eddy current problems*, Comput. Methods Appl. Mech. Eng. 169 (1999), pp. 391–405.

[11] J.F. Bonnans and J. Laurent-Varin, *Computation of order conditions for symplectic partitioned Runge–Kutta schemes with application to optimal control*, Numer. Math. 103 (2006), pp. 1–10.

[12] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput. 16 (1995), pp. 1190–1208.

[13] C.M. Campos, S. Ober-Blöbaum, and E. Trélat, *High order variational integrators in the optimal control of mechanical systems*, preprint (2015). Available at arXiv:1502.00325.
The first-order optimality system is obtained by seeking a stationary point of the Lagrangian of the discrete optimal control problem (10). We introduce the compact notation:

\[ Y_n = (Y_{n,1}^T, \ldots, Y_{n,T-1}^T)^T, \quad \hat{Y} = (Y_{0,1}^T, \ldots, Y_{N-1}^T)^T, \quad \hat{y} = (y_{ini,1}^T, \ldots, y_{ini,T}^T)^T. \]  

(A1)

Let \( \Lambda_{n}, \lambda_{n} \) denote the Lagrange multipliers associated to (10b) and (10c), respectively. The Lagrangian of the discrete optimal control problem (10) reads:

\[ \mathcal{L}(\hat{y}, \hat{Y}, \hat{\lambda}, \hat{\theta}) = \sum_{k=0}^{N} q_k(y_k, \theta) - \mathcal{L}_{ext}(\hat{y}, \hat{Y}, \hat{\lambda}, \hat{\theta}) - \mathcal{L}_{int}(\hat{y}, \hat{Y}, \hat{\lambda}, \hat{\theta}), \]

where

\[ \mathcal{L}_{ext} = \lambda_{0}^T(y_0 - y_{ini}^{(\theta)}) \]

\[ + \sum_{k=0}^{N-1} \lambda_{k+1}^T \left( y_{k+1} - y_k - B_{k,1}h\mathfrak{f}(y_k, \theta) - \sum_{j=2}^{s} B_{k,j}h\Delta^{(j-1)}r(y_k, \theta) \right). \]

(A2a)
\[
L_{\text{int}} = \sum_{k=0}^{N-1} \sum_{m=1}^{t-1} A^T_{k,m} \left( Y_{k,m} - y_k - A_{k,m,1} h f(y_k, \theta) - \sum_{j=2}^{m} A_{k,m,j} h \Delta^{(j-1)} r(y_k, \theta) \right).
\] (A2b)

In the above expressions, the forward difference operator \( \Delta^{(j-1)} r(y_k, \theta) \) can be written in a closed-form expression as

\[
\Delta^{(j-1)} r(y_k, \theta) = \sum_{\ell=0}^{j-1} C_{\ell,j} r(Y_{k,j-\ell}, \theta), \quad C_{\ell,j} = (-1)^{j-\ell} \binom{j-1}{\ell}.
\]

where \( r(Y_{k,j-\ell}, \theta) \) is the remainder term of the first-order Taylor expansion of \( f \) around \( y_k \), evaluated at \( Y_{k,j-\ell} \) with the exact Jacobian \( J_k \) replaced by the approximation \( T_k \):

\[
r(Y_{k,j-\ell}, \theta) = f(Y_{k,j-\ell}, \theta) - f(y_k, \theta) - T_k \cdot (Y_{k,j-\ell} - y_k).
\] (A3)

It should be pointed out that in the above expressions \( Y_{k,0} = y_k \) was introduced for brevity in later expressions. Substituting (A3) into (A2a) and (A2b), respectively, we obtain:

\[
L_{\text{ext}} = \lambda_T^T (y_0 - y_n) + \sum_{k=0}^{N-1} \lambda_{k+1}^T \left( y_{k+1} - y_k - B_{k,1} h f(y_k, \theta) 
\right.
\]

\[
- \sum_{j=2}^{t} B_{k,j} \left( \sum_{\ell=0}^{j-1} C_{\ell,j} [f(Y_{k,j-\ell}, \theta) - f(y_k, \theta) - T_k (Y_{k,j-\ell} - y_k)] \right)
\]

\[
L_{\text{int}} = \sum_{k=0}^{N-1} \sum_{m=1}^{t-1} A^T_{k,m} \left( Y_{k,m} - y_k - A_{k,m,1} h f(y_k, \theta) \right.
\]

\[
- \sum_{j=2}^{m} A_{k,m,j} h \left( \sum_{\ell=0}^{j-1} C_{\ell,j} [f(Y_{k,j-\ell}, \theta) - f(y_k, \theta) - T_k (Y_{k,j-\ell} - y_k)] \right).
\]

By enforcing \( \partial L / \partial \lambda_n = 0 \) and \( \partial L / \partial A_{n,i} = 0 \), we obtain the original EPIRK-W method, i.e. (10b), (10c) and (10d).

### A.2 Derivation of the discrete adjoint equation

The adjoint equations (13) are obtained by setting \( \partial L / \partial y_n = 0 \) and \( \partial L / \partial Y_{n,j} = 0 \). For a better readability, we compute various derivatives separately. We thereby adopt the convention that all terms for which subscripts turn out to be negative are automatically dropped or set to zero.

#### A.2.1 Computing \( \partial L_{\text{ext}} / \partial y_n \)

For \( 0 < n < N \), only the terms \( k = n - 1 \) and \( k = n \) in the sum \( \sum_{k=0}^{N-1} \) give non-zero contributions, as follows:

\[
\frac{\partial L_{\text{ext}}}{\partial y_n} = \lambda_n^T \frac{\partial}{\partial y_n} \left\{ y_n - y_{n-1} - B_{n-1,1} h f(y_{n-1}, \theta) \right. 
\]

\[
- \sum_{j=2}^{t} B_{n-1,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} [f(Y_{n-1,j-\ell}, \theta) - f(y_{n-1}, \theta) - T_{n-1} (Y_{n-1,j-\ell} - y_{n-1})]
\]

\[
+ \lambda_{n+1}^T \frac{\partial}{\partial y_n} \left\{ y_{n+1} - y_n - B_{n,1} h f(y_n, \theta) \right. 
\]
respect to \(Y\). The same non-zero terms in the sum remain when we differentiate with respect to \(Y\).

For \(n = 0\), we compute:

\[
\frac{\partial \mathcal{L}_{\text{ext}}}{\partial y_0} = \lambda_0^T + \lambda_1^T \cdot \frac{\partial}{\partial y_0} \left\{ y_1 - y_0 - B_{0,1} h f(y_0, \theta) \right\}
\]

\[
- \sum_{j=2}^s B_{j-1} h \sum_{\ell=0}^{j-1} \nabla L_{\ell,j} \left[ f(Y_{n,j-1,\ell}, \theta) - f(y_0, \theta) - T_n(Y_{n,j-1,\ell} - y_0) \right]
\]

\[
= \lambda_0^T + \lambda_1^T \cdot \left\{ -I - B_{0,1} h J(y_0, \theta) - \sum_{j=2}^s B_{j-1} h \sum_{\ell=0}^{j-2} \nabla L_{\ell,j} \left[ T_n - J(y_0, \theta) \right] \right\}.
\]

A.2.2 Computing \(\frac{\partial \mathcal{L}_{\text{ext}}}{\partial Y_{n,i}}\)

The same non-zero terms in the sum remain when we differentiate with respect to \(Y_{n,i}\) as when we differentiate with respect to \(y_i\). For \(0 \leq n < N\), we obtain:

\[
\frac{\partial \mathcal{L}_{\text{ext}}}{\partial Y_{n,i}} = -\lambda_{n+1}^T \cdot \left\{ \sum_{j=2}^s B_{n,j} h \sum_{\ell=0}^{j-1} \nabla L_{\ell,j} \left[ f(Y_{n,j-1,\ell}, \theta) - f(y_n, \theta) - T_n(Y_{n,j-1,\ell} - y_n) \right] \right\}
\]

\[
= -\lambda_{n+1}^T \cdot \left\{ \sum_{j=2}^s B_{n,j} h C_{j-1,i} \left[ J(y_n, \theta) - T_n \right] \right\}.
\]

For \(n = N\), the derivative is zero.
A.2.3 Computing $\partial L_{\text{int}} / \partial y_n$

The derivation is similar to the one already given in Section A.2.1. Therefore, we omit certain intermediate steps in the following. For $0 \leq n < N$, we obtain:

$$\frac{\partial L_{\text{int}}}{\partial y_n} = \sum_{m=1}^{x-1} A_{n,m}^{T} \cdot \frac{\partial}{\partial y_n} \left[ Y_{n,m} - y_n - A_{n,m,1} f(y_n, \theta) \right]$$

$$- \sum_{j=2}^{m} A_{n,m,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} \left[ f(Y_{n,j-1-\ell}, \theta) - f(y_n, \theta) - T_n(Y_{n,j-1-\ell} - y_n) \right]$$

$$= \sum_{m=1}^{x-1} A_{n,m}^{T} \cdot \left\{ -I - A_{n,m,1} hJ(y_n, \theta) - \sum_{j=2}^{m} A_{n,m,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} \left[ -J(y_n, \theta) + T_n \right] \right\},$$

whereas there is no contribution for $n = N$.

A.2.4 Computing $\partial L_{\text{int}} / \partial Y_{n,i}$

Again, omitting several intermediate results due to the similarity to Section A.2.2 we obtain for $0 \leq n < N$:

$$\frac{\partial L_{\text{int}}}{\partial Y_{n,i}} = \sum_{m=1}^{x-1} A_{n,m}^{T} \cdot \frac{\partial}{\partial Y_{n,i}} \left[ Y_{n,m} - y_n - A_{n,m,1} f(y_n, \theta) \right]$$

$$- \sum_{j=2}^{m} A_{n,m,j} h \sum_{\ell=0}^{j-1} C_{\ell,j} \left[ f(Y_{n,j-1-\ell}, \theta) - f(y_n, \theta) - T_n(Y_{n,j-1-\ell} - y_n) \right]$$

$$= A_{n,i}^{T} - \sum_{m=1}^{x-1} A_{n,m}^{T} \cdot \left\{ \sum_{j=2}^{m} A_{n,m,j} h C_{j-1,i} \left[ J(Y_{n,i}, \theta) - T_n \right] \right\}.$$

Finally, we observe that at the stationary point $(\hat{y}^*, \hat{Y}^*, \hat{\lambda}^*, \hat{\Lambda}^*)$ of the Lagrangian, both $L_{\text{ext}}$ and $L_{\text{int}}$ vanish and hence

$$\hat{\Phi}(\theta) = L(\hat{y}^*, \hat{Y}^*, \hat{\lambda}^*, \hat{\Lambda}^*). \quad (A4)$$

Differentiating this expression with respect to the parameter $\theta$ yields

$$\frac{d\hat{\Phi}}{d\theta} = \sum_{k=1}^{N} \frac{dq_k(y_k, \theta)}{d\theta} - \frac{d}{d\theta} L_{\text{ext}}(\hat{y}^*, \hat{Y}^*, \hat{\lambda}^*, \hat{\Lambda}^*, \theta) - \frac{d}{d\theta} L_{\text{int}}(\hat{y}^*, \hat{Y}^*, \hat{\lambda}^*, \hat{\Lambda}^*, \theta). \quad (A5)$$

from which we obtain the expression for the gradient, given in (14), by inserting the respective expressions for $L_{\text{ext}}$ and $L_{\text{int}}$. 