Adjoint-based space-time adaptive solution algorithms for sensitivity analysis and inverse problems

Mihai Alexe

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Adrian Sandu, Chair
Calvin J. Ribbens
Yang Cao
Eric De Sturler
Jeffrey T. Borggaard

March 18, 2011
Blacksburg, Virginia

Keywords: Inverse problems, Adjoint Method, Adaptive Mesh Refinement, Automatic Differentiation

Adjoint-based space-time adaptive solution algorithms for sensitivity analysis and inverse problems

Mihai Alexe

(ABSTRACT)

Adaptivity in both space and time has become the norm for solving problems modeled by partial differential equations. The size of the discretized problem makes uniformly refined grids computationally prohibitive. Adaptive refinement of meshes and time steps allows to capture the phenomena of interest while keeping the cost of a simulation tractable on the current hardware. Many fields in science and engineering require the solution of inverse problems where parameters for a given model are estimated based on available measurement information. In contrast to forward (regular) simulations, inverse problems have not extensively benefited from the adaptive solver technology. Previous research in inverse problems has focused mainly on the continuous approach to calculate sensitivities, and has typically employed fixed time and space meshes in the solution process. Inverse problem solvers that make exclusive use of uniform or static meshes avoid complications such as the differentiation of mesh motion equations, or inconsistencies in the sensitivity equations between subdomains with different refinement levels. However, this comes at the cost of low computational efficiency. More efficient computations are possible through judicious use of adaptive mesh refinement, adaptive time steps, and the discrete adjoint method.

This dissertation develops a complete framework for fully discrete adjoint sensitivity analysis and inverse problem solutions, in the context of time dependent, adaptive mesh, and adaptive step models. The discrete framework addresses all the necessary ingredients of a state–of–the–art adaptive inverse solution algorithm: adaptive mesh and time step refinement, solution grid transfer operators, *a priori* and *a posteriori* error analysis and estimation, and discrete adjoints for sensitivity analysis of flux–limited numerical algorithms.

This work was supported in part by the US National Science Foundation, through the following awards: NSF-CCF-0635194, NSF OCI-0904397, NSF CCF-0916493, and NSF DMS-0915047.

Dedication

To some truly special people: Silvia, Bianca, and my parents. Thank you for being there.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of this dissertation is to advance the current state of the art adaptive algorithms used in numerical solvers for sensitivity analysis and inversion of space-time dynamical models. This objective is achieved through the derivation of new theoretical results useful towards the definition of a fully adaptive inversion framework based on automatic differentiation, as well as through the development of efficient software tools for sensitivity analysis.

Inverse problems (IPs) consist in using *a priori* available measurements to infer the values of the defining parameters for a given model. IPs arise in various applications of engineering and mathematics, e.g., seismography, meteorology, oceanography, medical imaging, systems biology, and fluid dynamics (see, e.g., [5, 6, 7, 8, 9, 10]). IPs are usually described as constrained optimization problems, where the constraints are ordinary (ODE) or partial differential equations (PDEs) defining the model. Alternatively, the inverse problem can be stated in terms of probability densities. Using Bayes' theorem we arrive at the optimization formulation [11].

State of the art PDE solvers use adjoint-driven adaptive space-time refinement, and other dynamic computational patterns such as upwinding, slope or flux limiting, interpolations, extrapolations, variable order approximations (p-refinement), moving meshes, etc. (see, e.g., [12], and references therein). Space-time adaptivity controls the numerical errors introduced by the spatial and temporal discretizations, preserves the quality of the solution, and helps maximize solver efficiency.

In contrast, most approaches to numerical inversion have so far favored non-adaptive methods. There has been recently a growing trend of research into the use of adaptive inverse problem solvers (see, e.g., [13, 14, 15, 16, 17, 18, 19]), but there remains a considerable gap between the state of the art forward model solvers and the strategies used in inverse problems. This discrepancy is mainly caused by the difficulties with obtaining and using derivative information in adaptive simulations. Consistency of the derivatives computed through the adjoint equations is also an issue, as is the derivation of useful error estimates to judiciously

guide the space-time mesh adaptation during the solution process. This dissertation aims to narrow this gap, using both theoretical and practical arguments to demonstrate the feasibility and efficiency of adaptive solvers for inverse simulations of time and space-dependent dynamical models.

The framework we propose in this dissertation is built around the discrete adjoint method for solving inverse problems.

1.1 The mathematical formulation of inverse problems.

Given the model state $\mathbf{u} \in \mathcal{U}$, the inversion parameters $\mathbf{q} \in \mathcal{Q}$, and the real-valued target functional $\mathcal{J}$, the general continuous formulation of an inverse problem reads as follows:

$$\begin{aligned} \text{Find } \mathbf{q}_* \;\; = \;\; & \arg \min_{\mathbf{q}\in\mathcal{Q},\mathbf{u}\in\mathcal{U}} \mathcal{J}\left[\mathbf{u},\mathbf{q}\right] \;, \\ & \text{subject to } \mathcal{A}\left[\mathbf{u},\,\mathbf{q}\right]\left(\boldsymbol{\psi}_\mathbf{u}\right) = 0 \;,\; \forall\, \boldsymbol{\psi}_\mathbf{u} \in \mathcal{U} \;. \end{aligned} \tag{1.1}$$

Here $\mathcal{A}[\cdot,\,\cdot](\cdot)$ is a semi-linear form (linear in the test functionals $\boldsymbol{\psi}_\mathbf{u}$). We use square brackets for the nonlinear arguments, and round parentheses for the linear arguments. The numerical solution of (1.1) is the objective of the differentiate – then – discretize approach [13]. Here, one leverages the first order necessary conditions for a local optimum [20] to obtain a linearized system of optimality equations that are satisfied by all local solutions to (1.1). Given the optimal solution pair $\{\mathbf{u}_*, \mathbf{q}_*\}$ for (1.1), constrained optimization theory [20] guarantees, under suitable *a priori* assumptions on $\mathcal{J}$ and $\mathcal{A}$ [21], the existence of Lagrange multipliers $\boldsymbol{\lambda}_*$ such that the following Karush–Kuhn–Tucker (KKT) first order necessary conditions hold for the triplet $\boldsymbol{\xi}_* := \{\mathbf{u}_*,\, \boldsymbol{\lambda}_*,\, \mathbf{q}_*\} \in \mathcal{X} := \mathcal{U} \times \mathcal{U} \times \mathcal{Q}$:

$$\mathcal{A}\left[\mathbf{u}_*, \mathbf{q}_*\right](\mathbf{w}) \;\; = \;\; 0 \;,\; \forall\, \mathbf{w} \in \mathcal{U} \;, \tag{1.2a}$$

$$\mathcal{A}_\mathbf{u}\left[\mathbf{u}_*, \mathbf{q}_*\right](\boldsymbol{\lambda}_*)\left(\boldsymbol{\psi}_\mathbf{u}\right) \;\; = \;\; \mathcal{J}_\mathbf{u}\left[\mathbf{u}_*, \mathbf{q}_*\right](\boldsymbol{\psi}_\mathbf{u}) \;,\; \forall\, \boldsymbol{\psi}_\mathbf{u} \in \mathcal{U} \;, \tag{1.2b}$$

$$\mathcal{A}_\mathbf{q}\left[\mathbf{u}_*, \mathbf{q}_*\right](\boldsymbol{\lambda}_*)\left(\boldsymbol{\psi}_\mathbf{q}\right) \;\; = \;\; \mathcal{J}_\mathbf{q}\left[\mathbf{u}_*, \mathbf{q}_*\right](\boldsymbol{\psi}_\mathbf{q}) \;,\; \forall\, \boldsymbol{\psi}_\mathbf{q} \in \mathcal{Q} \;. \tag{1.2c}$$

The subscripts denote partial derivatives of the semi-linear form $\mathcal{A}$.

Assumption. For any admissible parameter function $\mathbf{q} \in \mathcal{Q}^{\text{adm}}$ the forward system has a unique solution. Similarly, the adjoint system has a unique solution. We denote them by:

$$\mathbf{u} = U[\mathbf{q}] \,, \quad \boldsymbol{\lambda} = \Lambda(\mathbf{q}) \;. \tag{1.3}$$

The reduced cost functional depends only on $\mathbf{q}$, as follows:

$$j(\mathbf{q}) = \mathcal{J}\left[U[\mathbf{q}],\, \mathbf{q}\right] \;. \tag{1.4}$$

The continuous optimality equations. Consider the Lagrangian functional $\mathcal{L} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ [20] associated to the constrained problem (1.1):

$$\mathcal{L}[\boldsymbol{\xi}] := \mathcal{J}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\ . \tag{1.5}$$

The KKT system above can be written more compactly as:

$$\mathcal{L}_{\boldsymbol{\xi}}\,[\mathbf{u}_*,\, \boldsymbol{\lambda}_*,\, \mathbf{q}_*]\,(\boldsymbol{\psi}) = 0\ ,\ \forall\, \boldsymbol{\psi} \in \mathcal{X}\ .$$

Here $\mathcal{L}_{\boldsymbol{\xi}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denotes the first variation of the Lagrangian, and is equal to

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{\xi}}\,[\mathbf{u},\, \boldsymbol{\lambda},\, \mathbf{q}]\,(\boldsymbol{\psi}_\mathbf{u}, \boldsymbol{\psi}_{\boldsymbol{\lambda}}, \boldsymbol{\psi}_\mathbf{q}) \;&=\; \mathcal{J}_\mathbf{u}[\mathbf{u},\, \mathbf{q}](\boldsymbol{\psi}_\mathbf{u}) + \mathcal{J}_\mathbf{q}[\mathbf{u},\, \mathbf{q}](\boldsymbol{\psi}_\mathbf{q}) \\
&\quad -\mathcal{A}\,[\mathbf{u}, \mathbf{q}]\ (\boldsymbol{\psi}_{\boldsymbol{\lambda}}) - \mathcal{A}_\mathbf{u}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\,(\boldsymbol{\psi}_\mathbf{u}) - \mathcal{A}_\mathbf{q}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\,(\boldsymbol{\psi}_\mathbf{q}) \\
&=\; -\mathcal{A}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\psi}_{\boldsymbol{\lambda}}) + \{\mathcal{J}_\mathbf{u}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}_\mathbf{u}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\}\ (\boldsymbol{\psi}_\mathbf{u}) \\
&\quad + \{\mathcal{J}_\mathbf{q}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}_\mathbf{q}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\}\ (\boldsymbol{\psi}_\mathbf{q})\ .
\end{aligned}$$

The second variation of the Lagrangian reads:

$$\begin{aligned}
&\mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}\,[\mathbf{u},\, \boldsymbol{\lambda},\, \mathbf{q}]\,(\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\phi}_{\boldsymbol{\lambda}}, \boldsymbol{\phi}_\mathbf{q}\,;\, \boldsymbol{\psi}_\mathbf{u}, \boldsymbol{\psi}_{\boldsymbol{\lambda}}, \boldsymbol{\psi}_\mathbf{q}) \\
&\quad = \mathcal{J}_{\mathbf{u},\mathbf{u}}[\mathbf{u},\, \mathbf{q}](\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_\mathbf{u}) + \mathcal{J}_{\mathbf{u},\mathbf{q}}[\mathbf{u},\, \mathbf{q}](\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_\mathbf{u}) \\
&\quad + \mathcal{J}_{\mathbf{q},\mathbf{u}}[\mathbf{u},\, \mathbf{q}](\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_\mathbf{q}) + \mathcal{J}_{\mathbf{q},\mathbf{q}}[\mathbf{u},\, \mathbf{q}](\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_\mathbf{q}) \\
&\quad - \mathcal{A}_\mathbf{u}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\psi}_{\boldsymbol{\lambda}})\,(\boldsymbol{\phi}_\mathbf{u}) - \mathcal{A}_\mathbf{q}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\psi}_{\boldsymbol{\lambda}})\,(\boldsymbol{\phi}_\mathbf{q}) \\
&\quad - \mathcal{A}_\mathbf{u}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\phi}_{\boldsymbol{\lambda}})\,(\boldsymbol{\psi}_\mathbf{u}) - \mathcal{A}_{\mathbf{u},\mathbf{u}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\,(\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_\mathbf{u}) - \mathcal{A}_{\mathbf{u},\mathbf{q}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\,(\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_\mathbf{u}) \\
&\quad - \mathcal{A}_\mathbf{q}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\phi}_{\boldsymbol{\lambda}})\,(\boldsymbol{\psi}_\mathbf{q}) - \mathcal{A}_{\mathbf{q},\mathbf{u}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\,(\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_\mathbf{q}) - \mathcal{A}_{\mathbf{q},\mathbf{q}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\,(\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_\mathbf{q})\ .
\end{aligned}$$

This equation can be rearranged as follows:

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}\,[\boldsymbol{\xi}]\,(\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\phi}_{\boldsymbol{\lambda}}, \boldsymbol{\phi}_\mathbf{q}\,;\, \boldsymbol{\psi}_\mathbf{u}, \boldsymbol{\psi}_{\boldsymbol{\lambda}}, \boldsymbol{\psi}_\mathbf{q}) \;&=\; \{\mathcal{J}_{\mathbf{u},\mathbf{u}}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}_{\mathbf{u},\mathbf{u}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\}\ (\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_\mathbf{u}) \\
&\quad + \{\mathcal{J}_{\mathbf{u},\mathbf{q}}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}_{\mathbf{u},\mathbf{q}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\}\ (\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_\mathbf{u}) \\
&\quad + \{\mathcal{J}_{\mathbf{q},\mathbf{u}}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}_{\mathbf{q},\mathbf{u}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\}\ (\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_\mathbf{q}) \\
&\quad + \{\mathcal{J}_{\mathbf{q},\mathbf{q}}[\mathbf{u},\, \mathbf{q}] - \mathcal{A}_{\mathbf{q},\mathbf{q}}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\lambda})\}\ (\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_\mathbf{q}) \\
&\quad - \mathcal{A}_\mathbf{u}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\phi}_\mathbf{u}, \boldsymbol{\psi}_{\boldsymbol{\lambda}}) - \mathcal{A}_\mathbf{u}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\phi}_{\boldsymbol{\lambda}}, \boldsymbol{\psi}_\mathbf{u}) \\
&\quad - \mathcal{A}_\mathbf{q}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\phi}_\mathbf{q}, \boldsymbol{\psi}_{\boldsymbol{\lambda}}) - \mathcal{A}_\mathbf{q}\,[\mathbf{u}, \mathbf{q}]\,(\boldsymbol{\phi}_{\boldsymbol{\lambda}}, \boldsymbol{\psi}_\mathbf{q})\ .
\end{aligned} \tag{1.6}$$

The primal, dual, and optimality equations are often solved simultaneously (*all–at–once*), to yield a new search direction for the nonlinear solution algorithm. For example, the Newton update with the solution increment $\boldsymbol{\delta\xi}_k$ at iteration k reads:

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_k](\boldsymbol{\delta\xi}_k\,;\, \boldsymbol{\psi}) \;&=\; -\mathcal{L}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_k](\boldsymbol{\psi})\ ,\ \forall\, \boldsymbol{\psi} \in \mathcal{X}\ , \\
\boldsymbol{\xi}_{k+1} \;&=\; \boldsymbol{\xi}_k + \boldsymbol{\delta\xi}_k\ .
\end{aligned}$$

The advantage of the continuous formulation lies in its flexibility. The complete solution algorithm, i.e., both the model equations, and the nonlinear minimization algorithm for $\mathcal{J}$,

can be formulated in function spaces [13]. This allows arbitrary choices of finite-element type discretizations once the problem has been fully specified in a functional space setting. Space-time meshes can be changed between nonlinear iterations, and convergence can be quantified in a mesh-independent fashion. There are also few restrictions on the mesh types and trial function spaces [13, 14]. The main drawback of this method is its additional complexity in both derivation and implementation, since it is not amenable to automatic code generation.

The *discretize–first* approach starts from the discrete counterpart of (1.1), obtained using the discontinuous Galerkin finite element method [22]:

$$\begin{aligned} \text{Find } \mathbf{q}_*^h &= \arg \min_{\mathbf{q}^h \in \mathcal{Q}_h, \mathbf{u}^h \in \mathcal{U}_h} \mathcal{J}^h[\mathbf{u}^h, \mathbf{q}^h] \,, \\ &\qquad \text{subject to } \mathcal{A}^h[\mathbf{u}^h,\, \mathbf{q}^h]\,(\boldsymbol{\psi}_{\mathbf{u}}^h) = 0 \;,\; \forall\, \boldsymbol{\psi}_{\mathbf{u}}^h \in \mathcal{U}_h \;. \end{aligned} \tag{1.7}$$

The discrete variables and function spaces are denoted by the superscript (or subscript) h. The discrete function spaces are $\mathcal{U}_h \subset \mathcal{U}$, and $\mathcal{Q}_h \subset \mathcal{Q}$. The weak form $\mathcal{A}^h$ is linear in $\boldsymbol{\psi}_{\mathbf{u}}^h$, but may be nonlinear in both $\mathbf{q}^h$ and $\mathbf{u}^h$. The discrete weak formulation of the primal model is *a priori* assumed to be a consistent and stable discretization of the original weak-form PDE in (1.1).

The discrete optimality equations. Again, we assume there exists at least one locally unique solution to (1.7). Such a solution $\boldsymbol{\xi}_*^h := \{\mathbf{u}_*^h,\, \boldsymbol{\lambda}_*^h,\, \mathbf{q}_*^h\} \in \mathcal{X}_h = \mathcal{U}_h \times \mathcal{U}_h \times \mathcal{Q}_h$ is required to satisfy the discrete KKT necessary conditions:

$$\mathcal{A}^h[\mathbf{u}_*^h, \mathbf{q}_*^h](\boldsymbol{\psi}_{\mathbf{u}}^h) = 0 \;,\; \forall\, \boldsymbol{\psi}_{\mathbf{u}}^h \in \mathcal{U}_h \;, \tag{1.8a}$$

$$\mathcal{A}^h_{\boldsymbol{\lambda}^h}[\mathbf{u}_*^h, \mathbf{q}_*^h](\boldsymbol{\psi}_{\mathbf{u}}^h)(\boldsymbol{\lambda}_*^h) = \mathcal{J}^h_{\mathbf{u}^h}[\mathbf{u}_*^h, \mathbf{q}_*^h](\boldsymbol{\psi}_{\mathbf{u}}^h) \;,\; \forall\, \boldsymbol{\psi}_{\mathbf{u}}^h \in \mathcal{U}_h \;, \tag{1.8b}$$

$$\mathcal{A}^h_{\mathbf{q}^h}[\mathbf{u}_*^h, \mathbf{q}_*^h](\boldsymbol{\delta}\mathbf{q}^h) = \mathcal{J}^h_{\mathbf{q}^h}[\mathbf{u}_*^h, \mathbf{q}_*^h](\boldsymbol{\delta}\mathbf{q}^h) \;,\; \forall\, \boldsymbol{\delta}\mathbf{q}^h \in \mathcal{Q}_h \;, \tag{1.8c}$$

or, in more compact notation

$$\mathcal{L}^h_{\boldsymbol{\xi}^h}[\mathbf{u}_*^h,\, \boldsymbol{\lambda}_*^h,\, \mathbf{q}_*^h](\boldsymbol{\psi}^h) = 0 \;,\; \forall\, \boldsymbol{\psi}^h \in \mathcal{X}_h \;,$$

with the discrete Lagrangian functional

$$\mathcal{L}^h[\boldsymbol{\xi}^h] := \mathcal{J}^h[\mathbf{u}^h,\, \mathbf{q}^h] - \mathcal{A}^h[\mathbf{u}^h, \mathbf{q}^h](\boldsymbol{\lambda}^h) \;. \tag{1.9}$$

The subscripts denote partial derivatives with respect to the discrete variables.

While the discretize–first approach lacks the flexibility of its differentiate–first counterpart, the main advantage of the former approach lies in the observation that the KKT system can, even for complicated problems, be generated with relatively low effort using automatic differentiation [23]. The grid transfer operators used for mesh refinement and coarsening in discontinuous Galerkin are also amenable to automatic differentiation [24]. This can significantly reduce the software development time required for a full implementation. Also,

for some practical problems, the solution algorithm interface may need with other numerical optimization or ODE solvers that require the inputs to be discrete mesh variables. If this is the case, then discretize–first is the only feasible approach. Finally, we mention that the discrete adjoint approach is a natural fit for multigrid optimization [25, 26], which may compensate its inability to adapt the mesh between consecutive nonlinear iterations.

As previously indicated in the literature (see, e.g., [27]), the linearization and discretization steps do not generally commute.

1.1.1 Consistency of the DG discretizations

We now define consistency for the primal, dual, and optimality equation discretizations (1.8a)–(1.8c). The primal and dual consistency definitions follow the ones given in [28]. In addition, we consider the consistency of the primal discretization. Consistency of (1.8c) is a crucial requirement for the convergence of the discrete optimal solution $\mathbf{q}^h$ to its analytical counterpart $\mathbf{q}$.

Definition 1.1.1 (Primal consistency)**.** *The primal discretization (1.8a) is said to be consistent if the exact solutions* $\mathbf{u}$ *and* $\mathbf{q}$ *to the weak form primal equation (1.2a) satisfy:*

$$\mathcal{A}^h[\mathbf{u},\mathbf{q}](\boldsymbol{\psi}_\mathbf{u}) = 0 \ , \ \forall \ \boldsymbol{\psi}_\mathbf{u} \in \mathcal{U} \ . \tag{1.10}$$

Definition 1.1.2 (Dual consistency)**.** *The primal discretization (1.8a) is said to be dual consistent, if any triplet* $\boldsymbol{\xi} = \{\mathbf{u},\boldsymbol{\lambda},\mathbf{q}\} \in \mathcal{X}$ *that verifies the primal equation (1.2a), also satisfies:*

$$\mathcal{A}^h_{\mathbf{u}^h}[\mathbf{u},\mathbf{q}](\boldsymbol{\psi}_\mathbf{u},\boldsymbol{\lambda}) = \mathcal{J}^h_{\mathbf{u}^h}[\mathbf{u},\mathbf{q}](\boldsymbol{\psi}_\mathbf{u}) \ , \ \forall \ \boldsymbol{\psi}_\mathbf{u} \in \mathcal{U} \ . \tag{1.11}$$

Definition 1.1.3 (Optimality equation consistency)**.** *The discretization (1.8c) is said to be consistent, if any triplet* $\boldsymbol{\xi} = \{\mathbf{u},\boldsymbol{\lambda},\mathbf{q}\} \in \mathcal{X}$ *that verifies (1.2c), also satisfies the equation:*

$$\mathcal{A}^h_{\mathbf{q}^h}[\mathbf{u},\mathbf{q}](\boldsymbol{\delta}\mathbf{q}) = \mathcal{J}^h_{\mathbf{q}^h}[\mathbf{u},\mathbf{q}](\boldsymbol{\delta}\mathbf{q}) \ , \ \forall \ \boldsymbol{\delta}\mathbf{q} \in \mathcal{Q} \ . \tag{1.12}$$

The definitions above can be extended to time–dependent problems, where the time dimension is discretized using a Runge–Kutta quadrature [24].

The strong form of the primal equations.

For reference in Chapter 5, we also give the strong formulations of the primal equations in (1.1) and (1.7). The inclusion of the time dimension is explicitly indicated through the addition of the time variable t, or the time superscript n (for discrete functions). The strong

form of the primal constraint in (1.1) reads:

$$\begin{array}{lcll} \mathcal{F}\left(\mathbf{u},\, \mathbf{u}_t,\, \mathbf{u}_{\mathbf{x}},\, \mathbf{u}_{\mathbf{xx}},\, \ldots,\, \mathbf{q},\, t,\, \mathbf{x}\right) & = & 0\,, & \mathbf{x} \in \Omega\,,\; t^0 \leq t \leq t^N \\ \mathcal{F}_B(\mathbf{u}, \mathbf{u}_{\mathbf{n}}, \ldots, \mathbf{q},\, t,\, \mathbf{x}) & = & 0\,, & \mathbf{x} \in \Gamma := \partial\Omega \\ \mathbf{u}(t^0,\, \mathbf{x}) & = & \mathbf{u}^0(\mathbf{q})\,, & \mathbf{x} \in \Omega\,. \end{array}$$

The discretization of (1.13) can be written as:

$$\begin{array}{lcll} \mathcal{F}^{h,n}\left(\mathbf{u}^{h,n}(\mathbf{x}^h),\, \mathbf{u}^{h,n-1}(\mathbf{x}^h),\, \mathbf{q}^h,\, t^n\right) & = & 0\,, & \mathbf{x}^h \in \Omega_n^h\,,\, 1 \leq n \leq N \\ \mathbf{u}^{h,0}(t^0,\, \mathbf{x}^h) & = & \mathbf{u}^{h,0}\,, & \mathbf{x}^h \in \bar{\Omega}^h{}_0 := \Omega_0^h \cup \Gamma_n^h\,. \end{array}$$

Note that $\mathcal{F}^{h,n}$ incorporates both volume, and boundary residuals. Let the discrete time step be denoted by $\tau^n := t^{n+1} - t^n$. We assume that the discrete primal solution $\mathbf{u}^{h,n}$ is $\mathcal{L}^2$-convergent to $\mathbf{u}$ on Ω in the limit of the discretization, i.e.:

$$\lim_{h \to 0, \tau^n \to 0} \left\| \mathbf{u} - \mathbf{u}^{h,n} \right\|_{\mathcal{L}^2(\Omega)} = 0\,.$$

For compatible cost functionals, which will be formally defined in Chapter 5, the strong forms can be given for the dual and optimality condition equations (1.2b)–(1.2c).

1.2 Research accomplishments

We summarize the main contributions of this dissertation in the following paragraphs.

Adjoint sensitivity analysis of flux-limited finite volume solvers. As previously noted by other authors [29], the discrete adjoints of computationally adaptive schemes, such as the well-known upwind methods, can display spurious oscillations resulting from the black-box automatic differentiation of the upwind switching mechanism. For certain types of physical problems that model naturally occurring phenomena, the numerical solution must lie within *a priori* defined bounds to be considered admissible. For example, concentrations of pollutants or tracers in the Earth's atmosphere or oceans must remain positive throughout the whole numerical simulation. Discontinuities in the analytical solution profiles are also common: one can think of interfaces between regions of distinct saline concentrations in the Earth's oceans, or between layers of rock with different porosity or permeability in the Earth's crust. The finite volume method (FVM) is a very good fit for conservation laws or other type of transport phenomena. It is in widespread use in numerical simulations. However, simple finite volume schemes may introduce unacceptable oscillations in the numerical solution around these regions of discontinuity. To prevent such numerical undershoots or overshoots from occurring, the simulation must use either a discretization which is first order accurate in space and does not introduce new extrema (but suffers from excessive diffusivity and low

accuracy), or employ a flux limiter function that reduces the order locally in non-smooth regions. Sufficiently far from the discontinuity the solution the limiter function is not active, and the numerical solution remains highly accurate, as desired. *This dissertation analyzes the discrete adjoints of flux limited finite volume codes used in sensitivity analysis. We find that discrete adjoints implemented by–the–book are not consistent with their continuous counterparts. Adjoining the limiting mechanism introduce unstable oscillations in the adjoint sensitivities. However, if the forward code is carefully re-factored before the discrete adjoint derivations, stability and accuracy of the sensitivity solutions is restored.* This result proves very important in practice since discrete adjoints can be generated with low effort using automatic differentiation from their primal model counterparts.

Continuous sensitivity analysis of adaptive time stepping methods. This dissertation also considers inverse problems with time adaptivity, from both *differentiate–first* and *discretize–first* perspectives.

The *differentiate–first* approach allows the highest level of flexibility in the sensitivity computations. The primal and tangent linear equations can be integrated simultaneously with adaptive time stepping, as a coupled system of ODEs. This approach can be expected to work well in practice, since both problems share the same eigen-structure through the Jacobian of the right–hand side term. The adjoint final-value problem can be solved separately on a different mesh for maximum computational efficiency. However, for general nonlinear problems, independent adaptation of the adjoint temporal mesh requires on–the–fly interpolation of the primal and tangent linear solutions in the dual ODE solver. *For the high order accuracy that may be necessary for some problems in sensitivity analysis, this dissertation proposes the use of high–order Runge–Kutta embedded methods, together with dense output interpolation for the primal and tangent linear states.* Use of high–order interpolants avoids the order reduction in the dual solution that would be noticed when lower–order Hermite interpolants are used. The advantages of dense output are thus three–fold: high accuracy, ease of implementation (since the mechanism is already built in the Runge–Kutta pair), and low overhead (it only requires a small number of additional stages, or function evaluations, per time step).

We implement the Fortran 90 library DENSERKS for high accuracy sensitivity and (first/second order) adjoint computations. The library comprises several explicit Runge-Kutta codes using embedded schemes up to 8th order. Cubic and quintic Hermite interpolants are implemented for use with lower-order methods. The memory space requirements for reversals are drastically reduced by the use of a two-level checkpointing mechanism. *We find that the overall performance (accuracy and efficiency) obtained with DENSERKS compared very well against those of established sensitivity and adjoint ODE solvers like Sundials [30] or DASPKADJOINT [31].*

Discrete adjoints of adaptive time integration algorithms. The alternative approach starts, as discussed above, with the discrete formulation of the inverse problem. The adjoint model is then obtained through a transpose operation of the linearized equations, which may be performed automatically with the help of automatic differentiation. A very important property of both implicit and explicit fixed-step Runge–Kutta methods is that they retain their temporal order of accuracy p under the linearization and adjoining operations (for arbitrary $p \geq 1$).

However, use of black–box AD does introduce several complications. First examined by Eberhard and Bischof [4], the forward linearizations of discrete time–adaptive adaptive models are not consistent discretizations of the continuous sensitivity equations. This is due to the automatic generation of unphysical derivatives stemming from the dependence of the solution on the current (variable) time step size and previous time point, through the error estimation formulas. Post processing of the TLM code is required to restore accuracy in the tangent linear solution.

We extend the tangent linear consistency analysis in [4], to cover discrete adjoints of general one-step explicit methods. The adjoint method is much more economical than the forward sensitivity approach when the size of the control space is much larger than the number of objective functionals (which is very often the case in practice). Similar to forward differentiation, reverse mode black-box AD compromises the derivative values. *Our numerical analysis shows that AD engine detects and differentiates the dependence between the time step and the primal solution at previous integration times. Non-physical gradients compromise the discrete adjoint solution. However, we show that the discrete adjoint solution can be made fully accurate by post-processing the adjoint code. Moreover, our derivations indicate that a simple final time step adjustment in the forward simulation does not restore full accuracy in the discrete tangent linear or adjoint solutions, contrary to what was argued in [4].* We also consider second order adjoint models obtained by *forward–over–reverse differentiation*. The theoretical findings are supported with comprehensive numerical results for several adaptive ODE models.

Inverse problems with space–time adaptivity. Apart from temporal mesh refinement, space adaptivity is the other key ingredient in fully adaptive inverse simulations. Adaptivity in the space dimension raises several complications in the inversion process that are not encountered when using exclusively static meshes. The prototypical numerical method used in this dissertation for the space discretization is the discontinuous Galerkin finite element method (DG-FEM) [22]. The method is extremely well suited for h/p-refinement and parallel computation, due to the low degree coupling between elements of the spatial triangulation. The dual consistency theory for spatial DG discretizations is well developed [28, 32]. However, these consistency definitions and results cannot be immediately extended to space-time discretization, unless the time dimension is also discretized by DG. For the more popular case of Runge–Kutta DG (RK–DG) methods, extensions of the definition of consistency to

the full space-time weak discretization are not immediately apparent. *Using tools from the consistency theory of RK temporal discretizations, and from the formal RK adjoint theory [33], this dissertation proves that RK–DG discretizations that are dual consistent in space, are also adjoint consistent in time. Moreover, these discretizations retain the temporal order of accuracy of their primal counterparts.* Empirical order of accuracy experiments for a two–dimensional inverse problem back up the theoretical derivations. The discrete dual solution is used for the determination of a descent direction for the discrete target functional in a robust truncated–Newton optimization algorithm. Good quality analysis solutions are retrieved even with significant noise levels in the observation data.

Discrete adjoints of mesh transfer operators with the DG and finite volume methods. The adaptive solution algorithm also introduces the intergrid transfer operators that project the primal approximation between meshes at every refinement step. The linearized transposes of these mesh transfer operators are then reused in the adjoint time reversal simulation for back–projection of the discrete dual solution. This immediately raises the question of whether the intergrid transfer operators are dual consistent, in the sense that they do not negatively impact the quality of the dual solution (i.e., all reduction in accuracy is due to local coarsening, and not to inconsistent interpolation). *This dissertation analyzes the discrete adjoints of h/p-mesh transfer operators for both hierarchical refinement (common technique in practice, due to high performance data structure implementations), and unstructured mesh adaptation (where the domain is completely re-meshed at each refinement step). We find the mesh transfer operators for DG to be dual consistent.* However, this property is not valid for all types of discretizations. *Finite volumes intergrid interpolations and restrictions are found herein not to satisfy this transpose requirement. It is shown that the use of the inconsistent dual mesh transfer operators will reduce the local order of the dual solution, and introduce first order perturbations in the neighbors of coarsened elements.*

***A priori* error analysis of the discrete KKT equations.** Given the optimality systems (1.2a)–(1.2c), and (1.8a)–(1.8c), this dissertation examines the differences between the continuous and discrete dual and optimality conditions. As discussed above, the dual consistency theory for (1.8b) has been previously examined by other authors. *This dissertation extends the consistency concept to the discrete optimality equation, i.e., the third equation in the KKT set.* Since (1.8c) is linear in the unknown discrete optimal variables $\mathbf{q}^h$, stability and consistency implies convergence of the discrete optimal solution to the analytical value, by the Lax–Richtmyer equivalence theorem. *Similar to the case of adjoint equations, consistency of (1.8c) is dependent on the particular choice of primal discretization.* The model problem considered in Chapter 6 illustrates this problem. This dissertation proves that, while equation (1.8c) may not be *a priori* consistent, *consistency and stability can be retrieved through the introduction of stabilization terms via a consistent modification to the discrete target functional*, in a manner similar to that described in [28] for ensuring dual consistency. *We give a priori bounds for the additional terms introduced in the optimality*

condition through the linearization of the DG interface terms in primal discretization. The discrete optimality equation is proven to be asymptotically consistent. Numerical results with a DG implementation confirms the theoretical derivations.

***A posteriori* error analysis for parameter identification problems.** Another key ingredient in a space–time adaptive inversion algorithm is error estimation. An element–wise error indicator needs to guide the coarsening and refinement in the primal/dual and parameter meshes, *on–the–fly* during the solution process. The framework for optimal control of target functionals using the adjoint method is very well developed, see, e.g., the survey by Becker and Rannacher [34], and references therein. The dual–weighted residual approach (DWR for short) [35] has been successfully used for error estimation and grid adaptation in a variety of numerical studies (see [36, 37, 38, 39, 40, 41], to name just a few). However, the progress with error estimators for the error in the optimal solution has been so far much slower. Several energy norm estimates for the optimal solution error $\left\|\mathbf{q}-\mathbf{q}^h\right\|$ have been derived in the literature based on a coercivity estimate for the KKT saddle-point problem [42, 43], but their practical use is severely limited by their reliance on generally unknown stability constants [35]. *This dissertation extends the estimates for the control error from Becker and Vexler [25] to the general case of an infinite dimensional control space. To keep the computational cost of error estimation low, the Hessian $j_{\mathbf{q},\mathbf{q}}$ is computed using a reduced-space BFGS approximation.*

1.3 Dissertation layout

This dissertation is structured as follows.

Chapter 2 discusses the background of the work presented in this dissertation, and gives a short literature review on the topic of adaptive solutions of inverse problems. It also contains a short introduction to derivative computations for inverse problems, including the automatic code generation approach using AD. Chapter 3 considers the stability and consistency of discrete adjoints for flux-limited finite volume schemes, where physical considerations require the solution to be within a given admissible domain (e.g., positive). In chapter 4 we study the problem of time adaptivity for both continuous and discrete formulations of inverse problems, together with the complications that arise in forward and adjoint sensitivity calculations from variations in the time step size in a ordinary differential equation (ODE) or partial differential equation (PDE) temporal integration. Spatial (mesh) adaptivity is the subject of chapter 5. Here we discuss the adjoint formulation of spatially adaptive PDE problems. We consider both the formulation of adjoint problems, and the computation of gradients in function spaces (with the standard $\mathcal{L}^2$ inner products), for general space-time differential operators. Then, we investigate adjoint (dual) consistency for two common types of discretizations particularly amenable to h-refinement: finite volumes [44], and discontinuous Galerkin [22].

Chapter 6 investigates the first-order discrete optimality system (also called the KKT system) for adaptive problems, and relates it to the continuous formulation of the problem. *A priori* and *a posteriori* error estimates are obtained for the optimal solution. Finally, chapter 7 discusses the conclusions and implications of this work. Also, we outline several research directions that appear promising for future investigations.

Chapter 2

Fundamentals of sensitivity analysis

Introduction

In this section we review fundamental elements of forward and adjoint sensitivity analysis in the context of system described by differential equations.

The objective of sensitivity analysis is to obtain qualitative and quantitative information about the variations in a given dynamical model output, that are caused by perturbations in the model inputs. Most frequently the information collected is in the form of derivatives (also called sensitivities) of output functionals, with respect to a selected set of model inputs. This derivative information leads to an approximation of the output variation, that is accurate to first order in the (small) input perturbations. Probabilistic methods [45] are another approach to performing sensitivity analysis; however, they fall outside the scope of this dissertation.

There are two approaches to computing sensitivities: direct (or forward) and adjoint. The *forward* sensitivity approach relies on the solution to the linearized model equations. The linearized is performed with respect to the control variables of interest, resulting in the so-called *tangent linear equations*. The analytical form of these equations can be derived using the chain rule of differential calculus (given below). Alternatively, in the discrete approach, automatic differentiation of the primal discretization may be used to construct the discrete tangent linear equations. Section 2.1.1 gives further details on algorithmic differentiation, and discusses its uses in sensitivity analysis. The forward sensitivity approach is most economical when the number of functional outputs of interest is significantly larger than the number of inputs. For the class of problems that we will consider in this dissertation, this requirement is not satisfied; indeed, the converse is valid. One then resorts to the *adjoint* approach.

Continuous adjoint sensitivity analysis [46, 47, 48, 49, 50, 51] relies on the concept of duality for differential operators. The derivation of the adjoint system for a general PDE constrained

optimization problem fits in the scope of calculus of variations and functional analysis (see, e.g., [27] and references therein). The cost functional and the required derivatives may be expressed in terms of the adjoint (also called dual) variables. For time dependent systems, the adjoint equation is a final–value problem. It is also coupled with the primal equation. Additional constraints may need to be imposed on the model differential operators, and cost functional, to guarantee a well posed adjoint problem. These complications are very important in practice, and will be discussed at length in chapters 4 and 5. The computational process may be obtained through the discretization of the adjoint PDE system. It is important to note that the derivatives obtained with the continuous adjoint approach are only approximations to the true gradients of the analytical cost functional. They become exact only in the limit of the space and time discretizations.

Alternatively, in the *discrete adjoint sensitivity* approach, the adjoint problem may be derived directly from the primal discretization. The discrete adjoint model code can be generated by automatic differentiation. With this method, one obtains the exact derivatives of the discrete cost functional. The discrete adjoint approach is the method we advocate for use in adaptive inverse problems throughout this dissertation.

The adjoint method is most beneficial from a computational point of view, when the model parameter (input) space size is much larger than that of the number of functional outputs of interest. This is the case for a large class of practical applications, such as data assimilation, model calibration, or parameter identification problems.

This dissertation advocates the use of the discrete adjoint method to calculating sensitivities. The tangent linear equations will be discussed in Chapter 4. The adjoint PDE framework will be revisited in Chapter 5.

2.1 Derivative computations for sensitivity analysis

This section gives a introduction to the numerical computation of derivatives for sensitivity analysis and inverse problems. The theoretical concepts introduced here will be used frequently in the subsequent chapters of the dissertation.

2.1.1 Derivatives over discrete spaces

Recall from previous chapters that inverse problems may be formulated as discrete numerical optimization problems (equation (1.7)). Some form of derivative information for the discrete target functional $\mathcal{J}_h$ is required in the solution process. The same derivative computation problem arises in sensitivity analysis studies, where the tangent linear and adjoint equations (see 1.8a–1.8c) are obtained by differentiation of the primal constraints around specified points in the discrete space $\mathcal{U}_h$.

We begin with a few definitions [52, Section 5.3].

Definition 2.1.1 (The Fréchet derivative)**.** *Let $K \subset \mathbb{R}^n$, with $\mathbf{x}_0$ as an interior point, and define $\mathbf{F} : K \to \mathbb{R}^m$. We say $\mathbf{F}$ is Fréchet differentiable at $\mathbf{x}_0$ if there exists a matrix (linear operator) $\mathbf{J} \in \mathbb{R}^{m \times n}$ such that*

$$\mathbf{F}(\mathbf{x}_0 + \mathbf{h}) = \mathbf{F}(\mathbf{x}_0) + \mathbf{J}\,\mathbf{h} + o(\|\mathbf{h}\|)\ ,\ \textit{as}\ \|\mathbf{h}\| \to 0\ .$$

One can prove that $\mathbf{J} := \nabla \mathbf{F}(\mathbf{x}_0) = \left(\dfrac{\partial \mathbf{F}_i}{\partial \mathbf{x}_j} \right)_{i=1\ldots m, j=1\ldots n}$. For $m = 1$, we refer to $\mathbf{J} \in \mathbb{R}^{1 \times n}$ as the *gradient* of $\mathbf{F}$, otherwise $\mathbf{J}$ is the Jacobian matrix of the vector-valued function $\mathbf{F}$.

Definition 2.1.2 (The Gâteaux derivative)**.** *Let $K \subset \mathbb{R}^n$, with $\mathbf{x}_0$ as an interior point, and define $\mathbf{F} : K \to \mathbb{R}^m$. Also, let $\mathbf{h}$ be a fixed vector in $\mathbb{R}^n$. We say $\mathbf{F}$ is Gâteaux differentiable at $\mathbf{x}_0$ if there exists a matrix $\mathbf{J} \in \mathbb{R}^{n \times m}$ such that*

$$\boldsymbol{\delta}\mathbf{F}[\mathbf{x}_0](\mathbf{h}) := \lim_{t \to 0} \frac{\mathbf{F}(\mathbf{x}_0 + t\,\mathbf{h}) - \mathbf{F}(\mathbf{x}_0)}{t} = \mathbf{J}\,\mathbf{h}\ .$$

For $\|\mathbf{h}\| = 1$, the Gâteaux derivative is also referred to as the directional derivative (or first variation) in the literature. Note that Fréchet differentiability at $\mathbf{x}_0$ implies Gâteaux differentiability, but the reverse is not true.

The most frequently used techniques for derivative calculations are described below. We assume that $\mathbf{F}$ is at least twice Fréchet differentiable at all $\mathbf{x} \in K$.

Hand coded differentiation.

For some codes of relatively low complexity, manual differentiation is possible. Starting from the implementation of $\mathbf{F}$, new code is written that calculates the Fréchet or Gâteaux derivatives of $\mathbf{F}$ at the points of interest. However, the approach is extremely bug–prone, and usually very slow to implement. Hence, we do not recommend it for practical problems.

Finite differences.

The method of finite differences relies on Taylor's theorem for smooth functions [20]. By considering small perturbations in the arguments of $\mathbf{F}$, we can approximate the Jacobian $\mathbf{J}$ component–wise:

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_j} \approx \frac{\mathbf{F}_i(\mathbf{x} + h\,\mathbf{e}_j) - \mathbf{F}_i(\mathbf{x})}{h}\ , \tag{2.1}$$

Here $h > 0$ is sufficiently small. Here $\mathbf{e}_j \in \mathbb{R}^m$ is the j-th unit vector. Note that we require $m \cdot n + 1$ function evaluations to approximate the entire $\mathbf{J}$ matrix. Also, the optimal choice for h is usually unknown. Large values for h yield poor derivative approximations, while for small h the computation results can be corrupted by truncation errors. A frequently used *a priori* estimate for the optimal value of h is

$$h = \sqrt{\epsilon}\,,$$

where ϵ denotes the machine precision [20]. One can immediately generalize (2.1) to approximate directional derivatives along $\mathbf{d} \in \mathbb{R}^n$, for sufficiently small h:

$$\nabla \mathbf{F}[\mathbf{x}_0](\mathbf{d}) \approx \frac{\mathbf{F}(\mathbf{x} + h\,\mathbf{d}) - \mathbf{F}(\mathbf{x})}{h}\,.$$

Sparse Jacobians can be approximated at a much lower cost by merging multiple derivative approximations into a single computation (see, e.g., [20, Chapter 6]). Similarly, higher order derivative tensors can be built on top of lower-order terms. For example, Hessian–vector products for $\mathbf{F}$ can be computed as a finite difference of two first-order directional derivatives:

$$\left(\frac{\partial^2 \mathbf{F}}{\partial \mathbf{x}^2} \otimes \mathbf{d}\right) \cdot \widehat{\mathbf{d}} \approx \frac{\nabla_{\mathbf{d}} \mathbf{F}(\mathbf{x} + h\,\widehat{\mathbf{d}}) - \nabla_{\mathbf{d}} \mathbf{F}(\mathbf{x})}{h}\,.$$

Here the operator $\otimes$ indicates the Hessian-vector tensor product operation. The Hessian tensor of $\mathbf{F}$ is denoted by $\mathbf{H}(\mathbf{x}) := \dfrac{\partial^2 \mathbf{F}}{\partial \mathbf{x}^2}$.

Symbolic differentiation.

The symbolic differentiation approach (implemented in software products such as MAPLE or MATHEMATICA) relies on algorithmic manipulation of algebraic symbols in the definition of $\mathbf{F}$. Complex algebraic expressions (that may or may not be simplified) are produced for each derivative component. Symbolic differentiation sometimes leads to expression blow-up, due to its inability to eliminate common sub-expressions through intermediate variables. Also, it is relatively slow to produce a result, and the evaluation of the resulting expressions is usually not efficient. While both automatic and symbolic differentiation rely on the same differentiation rules (stemming from the chain rule from basic differential calculus), they do differ in one crucial aspect. Symbolic differentiation works on *the formula* for $\mathbf{F}$, hence the results are additional formulas instead of numerical values. Automatic differentiation, on the other hand, works on the algorithmic implementation of the function, and yields point-wise (floating-point) values for the required derivatives.

Automatic differentiation.

Automatic (or algorithmic) differentiation [23] is our method of choice when performing adaptive inversions. It is a technique for computing derivative information for functions defined by computer programs. Suppose we are given the source code for the implementation of:

$$\mathbf{y} = \mathbf{F}(\mathbf{x}) \,. \tag{2.2}$$

Automatic differentiation tools rely on the observation that the code to compute $\mathbf{F}$ can be viewed as a sequence of elementary and differentiable arithmetic operations. One can then apply the basic chain rule of differentiation to obtain derivatives. AD takes the original function code as input, and generates a transformed code that calculates derivatives of $\mathbf{F}(\mathbf{x})$. Naturally, the function $\mathbf{F}(\mathbf{x})$ needs to be mathematically differentiable at the points of interest for automatic differentiation to yield correct results. However, experimental support for calculating left and right derivatives at a discontinuity point of $\mathbf{F}$ exists in some automatic differentiation tools [53].

Comprehensive support for automatic differentiation is available for most of the programming languages used in scientific computing [53, 54, 55, 56, 57, 58, 59, 60]. It is useful to note the following strong points of the AD approach. First order derivative information can be computed at a very low cost (a very important result that motivated the widespread use of AD in the numerical modeling community is the *cheap gradient theorem* in [23]). The derivatives computed by AD are usually as accurate as the function computation itself, thus surpassing traditional approaches like finite differencing (which suffer from truncation errors). There is no code size blow-up like in the case of symbolic differentiation [61]. Finally, note that automatic code generation is also much less bug prone than hand coded differentiation.

Forward and reverse mode AD. Following Griewank [23], consider the nonlinear equation (2.2), with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$. We assume $\mathbf{F}$ is at least twice continuously differentiable in $\mathbf{x}$. The *forward mode* of AD with $\mathbf{x}$ as input and $\mathbf{y}$ as output yields

$$\boldsymbol{\delta}\mathbf{y} = \mathbf{J}(\mathbf{x})\, \boldsymbol{\delta}\mathbf{x} \,, \tag{2.3}$$

whereas the *reverse mode* computes

$$\begin{aligned} \bar{\mathbf{x}} &= \bar{\mathbf{x}} + \bar{\mathbf{y}}\, \mathbf{J}(\mathbf{x}) \\ \bar{\mathbf{y}} &= 0 \,. \end{aligned} \tag{2.4}$$

Here the *tangent linear* variable and the *adjoint* variable corresponding to $\mathbf{x}$ are denoted by $\boldsymbol{\delta}\mathbf{x} \in \mathbb{R}^n$ and $\bar{\mathbf{x}} \in \mathbb{R}^{1\times n}$, respectively.

The adjoint (or *co-state*) variables are defined as row vectors. For $n \gg m$, the reverse mode of AD is significantly more economical in practice than forward mode differentiation. This is important in inverse problems, where the cost functional that needs to be differentiated

is single-valued ($m = 1$), whereas the independent variables (the inversion parameters) may run in the millions or more. However, the adjoint mode of differentiation introduces several complications over the (more straightforward) forward approach. An automatic reversal of the original program control flow is required for reverse mode differentiation. Additional storage is needed for the intermediate variables in the forward code that may need to be reused in the reversed code. Recently, there has been significant progress in both off-line and on-line reversal schemes for computer programs [23, 62, 63, 64, 65, 66], that make reversal of complex codes computationally feasible.

Higher order derivatives. Higher order derivative information can be computed by AD through multiple differentiations. Since the dimensions of derivative tensors scales up very rapidly with both m and n, as their order increases (the curse of dimensionality), higher order derivative codes are more complex, and take longer to run. There is no encouraging upper bound for higher order Taylor coefficients. However, economical schemes exist [67, 68], which may make the computation of higher order derivative tensors computationally feasible in numerical simulation as computer speeds scale up. Second order information obtained via AD has been successfully used in various studies, such as data assimilation (see, e.g., [69, 70, 71, 72], among others). Various studies have found that the convergence of optimization algorithms can benefit from the addition of second order information [69, 70, 72, 73, 74, 75, 76]. Second order code optimizations, judicious computations of low-rank higher-order tensors, and reductions in the number of optimization iterations required to reach convergence, may offset the additional development effort and computational overhead of higher order derivative calculations.

Optimization algorithms were found to benefit from second order information up to the point that the additional cost of obtaining these derivatives is offset by the increases in performance and accuracy. We give a short overview of how to obtain Hessian-vector products via AD below.

From (2.3), forward mode differentiation yields:

$$\boldsymbol{\delta}\bar{\mathbf{x}} = \boldsymbol{\delta}\bar{\mathbf{y}}\,\mathbf{J}(\mathbf{x}) + \bar{\mathbf{y}}\,\mathbf{H}(\mathbf{x})\,\boldsymbol{\delta}\mathbf{x}\ .$$

Note first that, for a symmetric Jacobian matrix $F'(\mathbf{x}) = F'(\mathbf{x})^T$, we have

$$\bar{\mathbf{y}}\,\mathbf{J}(\mathbf{x}) = (\mathbf{J}(\mathbf{x})\,\boldsymbol{\delta}\mathbf{x})^T\ , \qquad \text{with} \qquad \boldsymbol{\delta}\mathbf{x} = \bar{\mathbf{y}}^T\ .$$

Thus the adjoint vector $\bar{\mathbf{x}} = \bar{\mathbf{y}}\,\mathbf{J}(\mathbf{x})$ can be computed by forward differentiation if $\mathbf{J}(\mathbf{x})$ is symmetric (i.e., it is the Hessian $\nabla^2\psi(\mathbf{x})$ of some scalar function $\psi(\mathbf{x})$ with the gradient $\nabla\psi(\mathbf{x}) = \mathbf{F}(\mathbf{x})$). Thus, it is never efficient to apply the reverse mode to vector functions that are gradients. This is important for symmetric Jacobians, because if we apply reverse mode differentiation to the forward model, we get:

$$\left[\mathbf{F}(\mathbf{x})^T,\ \ \bar{\mathbf{y}}\,\mathbf{J}(\mathbf{x})\right]^T = \nabla_{\bar{\mathbf{y}},\mathbf{x}}\ \bar{\mathbf{y}}\,\mathbf{F}(\mathbf{x})\ .$$

If we view the forward model (2.2) as an optimization problem, the result of reverse mode differentiation is *the gradient of the Lagrangian* functional $\bar{\mathbf{y}}\,\mathbf{F}(\mathbf{x})$. Hence, a second differentiation can be carried out *without loss of generality* in the forward mode. This is always computationally cheaper than adjoining, since no flow control reversals or checkpointing algorithms are required.

2.1.2 Derivatives over function spaces

We now give the formal definitions of derivatives for smooth nonlinear operators over function spaces [52].

Let $\mathbf{F} : K \subset \mathcal{V} \to \mathcal{W}$, where $\mathcal{V}$ and $\mathcal{W}$ are normed spaces. Denote by $\mathcal{L}(\mathcal{V}, \mathcal{W})$ the space of linear functionals from $\mathcal{V}$ onto $\mathcal{W}$. We define differentiability of $\mathbf{F}$ at an interior point $\mathbf{x}_0 \in \mathcal{V}$.

Definition 2.1.3 (The Fréchet derivative)**.** *The operator* $\mathbf{F}$ *is Fréchet differentiable at* $\mathbf{x}_0$ *if and only if there exists* $\mathbf{J} \in \mathcal{L}(\mathcal{V}, \mathcal{W})$ *such that*

$$\mathbf{F}(\mathbf{x}_0 + \mathbf{h}) = \mathbf{F}(\mathbf{x}_0) + \mathbf{J}(\mathbf{h}) + o(\|\mathbf{h}\|) \ , \ \textit{as}\ \mathbf{h} \to 0 \ .$$

The linear mapping $\mathbf{J} := \mathbf{F}'[\mathbf{x}_0]$ is called the Fréchet derivative of $\mathbf{F}$ at $\mathbf{x}_0$. Similarly, $\mathbf{F}'[\mathbf{x}_0](\mathbf{h})$ is the Fréchet *differential* of $\mathbf{F}$ at $\mathbf{x}_0$ along the direction $\mathbf{h}$. If $\mathbf{F}$ is Fréchet differentiable at all points in K, then the (unique) Fréchet derivative of $\mathbf{F}$ on K is

$$\mathbf{F}' : K \to \mathcal{L}(\mathcal{V}, \mathcal{W}) \ . \tag{2.5}$$

Definition 2.1.4 (The Gâteaux derivative)**.** *The operator* $\mathbf{F}$ *is Gâteaux differentiable at* $\mathbf{x}_0$ *if and only if there exists* $\mathbf{J} \in \mathcal{L}(\mathcal{V}, \mathcal{W})$ *such that*

$$\lim_{t \to 0} \frac{\mathbf{F}(\mathbf{x}_0 + t\,\mathbf{h}) - \mathbf{F}(\mathbf{x}_0)}{h} = \mathbf{J}(\mathbf{h}) \ , \ \forall\, \mathbf{h} \in \mathcal{V} \ .$$

Similar definitions and notations as above apply for the Gâteaux derivative. The Gâteaux and Fréchet derivatives obey the classical sum and product rules of differentiation. An important result is the chain rule of differentiation, on which many of the derivations in this dissertation will be based. It is stated below; for a proof, see [77].

Proposition 2.1.1 (The chain rule of differentiation)**.** *Let* $\mathcal{V}$*,* $\mathcal{W}$*, and* $\mathcal{U}$ *be normed spaces, and* $\mathbf{F} : K \subset \mathcal{V} \to \mathcal{W}$*,* $\mathbf{G} : L \subset \mathcal{W} \to \mathcal{U}$*, with* $\mathbf{F}(K) \subset L$*. Assume* $\mathbf{x}_0$ *is an interior point of* K*, and* $\mathbf{F}(\mathbf{x}_0)$ *is an interior point of* L*. Then:*

(a) If $\mathbf{F}'[\mathbf{x}_0]$ *and* $\mathbf{G}'[\mathbf{F}(\mathbf{x}_0)]$ *exist as Fréchet derivatives, then* $\mathbf{G} \circ \mathbf{F}$ *is Féchet differentiable at* $\mathbf{x}_0$*, and the chain rule holds:*

$$(\mathbf{G} \circ \mathbf{F})'[\mathbf{x}_0] = \mathbf{G}'[\mathbf{F}(\mathbf{x}_0)]\,\mathbf{F}'[\mathbf{x}_0] \ .$$

(b) If $\mathbf{F}'[\mathbf{x}_0]$ *exists as a Gâteaux derivative, and* $\mathbf{G}'[\mathbf{F}(\mathbf{x}_0)]$ *exists as a Fréchet derivative, then* $\mathbf{G} \circ \mathbf{F}$ *is Gâteaux differentiable at* $\mathbf{x}_0$*, and the chain rule given above holds.*

2.2 A brief literature review

This section gives a short overview of prior research in sensitivity analysis for the solution of inverse problems.

As stated above, the adjoint approach to calculating sensitivities is the method of choice in most practical problems. Let us consider the continuous approach. There are a few complications that arise when making use of the duality framework over function spaces. The theory of calculus of variations can lead to ill-posed adjoint systems for certain *inadmissible* cost functionals [78, 79, 80]. Giles et al. [50] give several conditions that need to be satisfied for a specific type of functional to be admissible. Here admissibility of the cost functional implies the existence of a well posed adjoint final boundary value problem. Arian and Salas [79] add additional boundary conditions (based on higher-order solution derivatives) to the forward model equations to obtain well posed adjoints. *In chapter 5, we formulate a more general set of compatibility conditions that need to be satisfied by both the primal differential operators, and the target functional, for the adjoint PDE to be well posed.* These considerations, important in the function space adjoint formulation, do not apply to the discrete duality framework. The target functional in the discrete adjoint approach is by construction compatible with the adjoint problem [27].

An important point is whether the discrete adjoint variables converge to the solution of the continuous adjoint equation (i.e., if the discrete adjoint is a stable and consistent discretization of the continuous problem). Spatial consistency is not automatically inherited by the adjoint method: the DA solution can display strong non-physical behavior, as it has been noticed in [81, 82]. Spurious spikes in the DA may appear near the domain boundary due to the strong boundary conditions imposed in the forward system [50, 83]. Other inconsistencies stem from the use of a smaller finite difference stencil, or from changes in upwinding [29].

Previous research efforts in adjoint sensitivity analysis for adaptive mesh refinement include the work of Li and Petzold [15]. The authors attempt to combine the advantages of both approaches to adjoint sensitivity, DA (discretization of the adjoint) and AD (adjoint of the discretization) into their ADDA method. However, their approach uses discrete adjoints only for a fixed grid inside the boundary layer, where the DA is assumed to be consistent, while employing the continuous adjoint method in the interior of the domain [15]. This avoids the difficult determination of suitable boundary conditions for the adjoint PDE. Anderson and Venkatakrishnan [78] describe a continuous adjoint sensitivity analysis approach on unstructured grids. It has also been shown that adaptive mesh refinement can also improve discrete adjoints in the presence of shocks, as pointed out by Giles *et al.* [84, 85, 86].

Marta [87] applies automatic differentiation on selected parts of two 3D CFD solvers (on unstructured meshes) to derive a consistent discrete adjoint model.

Recent work on four–dimensional data assimilation (4D-Var) for discrete models on nested grids includes that of Simon *et al.* [88, 89, 90]. The nested meshes are obtained through local hierarchical mesh refinement. The feature of their work is the inclusion of information from multiple grid length scales in the discrete adjoint model, the 4D-Var objective functional, as well as the background covariance matrices. Their numerical results of a shallow–water inversion problem on nested grids prove to be significantly better than those obtained through a single-grid approach [88].

Although the discrete adjoint approach is attractive in practice, most of the research effort for large scale inverse problems has gone into analyzing the continuous sensitivity approach. Bangerth [13] introduces a framework for the solution of parameter estimation problems on adaptive grids using finite element discretizations. The optimality system for the Lagrangian is solved using an *all-at-once* approach in a function space setting. This formulation allows maximum flexibility of the inversion algorithm, through the use of separate meshes for the primal, dual, and inversion variables. Moreover, convergence can be quantified in a mesh-independent fashion. Fang *et al.* have developed a fully adaptive 3D finite-element ocean model equipped with adjoint sensitivity analysis [16, 18, 19, 91, 92, 93]. The adjoints of the forward dynamics equations are discretized on fully adaptive meshes, with the forward variables interpolated onto the adjoint mesh *on–the–fly* wherever needed. For faster forward and backward simulations, the POD method [91] is used for reducing the state space of the model and its adjoint, while capturing their essential dynamics. Use of separate meshes for the primal and dual variables allows for maximum solver efficiency. However, numerical errors are introduced through the interpolation of forward variables onto the dual mesh in the time reversal procedure. The effect of such errors on the adjoint solution has not yet been fully analyzed. *We advocate the use of discrete adjoint sensitivities throughout the inversion process, due to their accuracy and ease of implementation (through automatic differentiation). However, if continuous sensitivities cannot be avoided, this dissertation provides a framework for the derivation of the tangent linear and adjoint PDE for general nonlinear time dependent models and cost functionals* (see Chapter 4, and [24]).

Time consistency of the adjoint discretization has also been investigated by several authors. Discrete adjoints of Runge-Kutta methods of arbitrary order were proven to be consistent with the original continuous adjoint equations [94, 95]. Moreover, the discrete adjoint Runge–Kutta quadratures retain the order of accuracy of the primal discretizations. Multistep methods, however, are not dual consistent [96]. Sensitivity analysis for ODEs with multistep methods has made exclusive use of the continuous approach [30]. Adaptive time integration has also received some attention. Eberhard and Bischof [4] have shown that black–box application of forward–mode AD yields unphysical gradients due to the dependency of the time step on the forward solution. The spurious sensitivities can be removed by code post-processing [4]. *This dissertation extended the work in [4] to discrete adjoints of one-step explicit methods. We show in chapter 4 that dual consistency is lost due to the emergence of*

spurious gradients; however, accuracy may be restored through adjoint code post-processing (see also [97]). The analysis also proves that a simple final time step adjustment in the forward simulation does not restore full accuracy in the TLM or DA solution, contrary to what was argued in [4]. A full adjoint trajectory correction is necessary.

Consistent adjoint sensitivities have been successfully used in spatial and temporal mesh refinement by several authors. Dual weighted residuals and error estimation [34] were shown to significantly improve grid adaptation and efficiency for types of model calibration problems with ODE or PDE constraints [98, 99, 100, 101, 102, 103, 104], including in the presence of shock discontinuities [105]. Cao and Petzold describe in [106] a duality framework for ODE global error control. Recent work includes the space–time adaptive mesh refinement strategy set forth by Carey *et al.* [12]. In their study, the refinement in the primal simulation is guided by coarse–scale dual solutions.

The framework for adjoint-based error control in optimal control problems, where the functional $\mathcal{J}$ is the quantity of interest, is well established (see, e.g., [34, 107] and references therein). Moreover, super-convergence in the functional approximations can be achieved through duality-based post-processing [108, 109].

There has been comparatively little work in error estimation for parameter identification problems. Previous research in this area includes that of Feng *et al.* [42, 43], Becker and Vexler [110, 111], and the recent survey by Rannacher and Vexler [35]. The energy norm estimates that have been derived for the optimal solution error [42, 43] are of limited use in practice due to their dependence on unknown stability or coercivity constants. A more practical approach is the one given by Becker and Vexler [110, 111]. Their error estimator is built for a modified problem that is defined in terms of a suitably chosen cost functional. *In this dissertation, we generalize the approach in [110, 111] to the case of an infinite dimensional control space. The Hessian of the reduced problem is replaced by a quasi-Newton approximation. For the time dimension, we advocate the use of dual–consistent adaptive Runge–Kutta discretizations.*

.

Chapter 3

Adjoint sensitivity analysis with finite volume models

This chapter investigates the behavior of discrete adjoint for high resolution finite volume codes. Equations modeling natural phenomena (such as basic advection or diffusion) often require that their solution belong to a certain physically admissible domain. For example, solutions to transport equations modeling the advection of pollutants in the Earth's atmosphere, must remain positive at any given time of the simulation. To add to this complication, the solutions of the analytical are often discontinuous, due to e.g., physical interfaces between regions of distinct concentrations in a certain pollutant or tracer. The discretization methods may introduce numerical undershoots or overshoots, like the Gibbs phenomenon in truncated polynomial approximations for non-smooth functions. Such under-, or overshoots, are unacceptable in numerical simulations of some phenomena, since they may render the numerical solution physically unfeasible.

Simple discretizations that are monotonic and do not lead to overshoots, have only first order accuracy [112], and suffer from excessive diffusivity. To enforce the positivity requirement for numerical solutions, and retain high temporal and spatial accuracy in smooth flow regions, one must employ *high resolution* schemes [1], that make use of solution or flux limiting functions to strictly damp oscillations in regions of discontinuity. This comes at the expense of local order reduction in regions of non-smooth flow. In smooth areas of the domain, however, the order of accuracy can be kept arbitrarily high.

Limiters are frequently used with the finite volume [1], discontinuous Galerkin [22], and other types of spectral methods (see, e.g., [113] and references therein). The following analysis concerns the finite volume method. We find that the discrete adjoints of forward models that use limiter functions leads to numerically unstable adjoint implementations. A detailed analysis of the forward model code structure reveals that refactoring is required before the derivation of the discrete adjoint equations. This strategy yields a stable adjoint solution. Numerical examples illustrate the behavior of discrete adjoints obtained through black-box

AD on two prototypical advection problems, before and after stabilization.

3.1 A brief overview of the finite volume method

Consider the following time dependent hyperbolic initial boundary value problem (IBVP):

$$\begin{array}{lcll} \mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) & = & \mathbf{f}\,, & \mathbf{x} \in \Omega\,,\, t \in [0,\, T] \\ \mathbf{u}(t, \mathbf{x}) & = & \mathbf{g}(t, \mathbf{x})\,, & \mathbf{x} \in \partial\Omega_{\text{in}} \\ \mathbf{u}(t = 0, \mathbf{x}) & = & \mathbf{u}^0(\mathbf{x})\,. & \end{array}$$

The finite volume method (FVM) [1] is built on the physical properties of the hyperbolic conservation law (3.1). It can be used on structured and unstructured meshes, and has excellent geometric flexibility. The formulation of the FVM relies on conservation of *mass* (in the general sense, some solution average) principles inside a given control volume (or cell). Our discrete spatial mesh at t^n is now defined as the union of K distinct control volumes: $\Omega_n^h = \bigcup_{k=1}^{K_n} C_n^k$. We define the approximate solution average inside a cell C_n^k at time t^n to be:

$$\mathbf{U}^{k,n} \approx \frac{1}{\text{Vol}(C_n^k)} \int_{C_n^k} \mathbf{u}(\mathbf{x},\, t^n)\, \mathrm{d}\mathbf{x}\,. \tag{3.1}$$

From (5.18) we obtain:

$$\frac{1}{\text{Vol}(C_n^k)} \left(\frac{\partial}{\partial t} \int_{C_n^k} \mathbf{u}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} + \int_{\partial C_n^k} \mathbf{F} \cdot \vec{\mathbf{n}}\, \mathrm{d}\mathbf{s} - \int_{C_n^k} \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} \right) = 0\,.$$

Let σ be a boundary edge (or face) for the control cell C_n^k. Using a conservative and consistent approximation $\mathbf{F}_\sigma^{\text{FVM}}(\mathbf{x}^h, t^n)$ for the analytical flux $\mathbf{F}(\mathbf{x}, t^n)$ through σ, we arrive at the semi-discrete finite-volume formulation of (5.18):

$$\frac{\mathrm{d}\mathbf{U}^{k,n}}{\mathrm{d}t} + \frac{1}{\text{Vol}(C_n^k)} \left(\sum_{\sigma \in \partial C_n^k} \mathbf{F}_\sigma^{\text{FVM}}(\mathbf{x}^h, t^n) - \int_{C_n^k} \mathbf{f}\left(\mathbf{x}^h, t^n\right) \mathrm{d}\mathbf{x} \right) = 0\,.$$

By virtue of the conservation principle, the net flow through the boundaries of the control volume must be zero in the absence of any forcing terms $\mathbf{f}$. The numerical reconstruction of the solution at the control volume interfaces is done locally. Hence, generalizations to unstructured grids and higher dimensions are straightforward. The various choices of numerical fluxes lead to different finite volume methods [1]. However, the FVM in two or three space dimensions is not easily amenable to p-refinement, since higher order solution

approximations rely on polynomial reconstructions over multiple cells. In d dimensions, one needs at least

$$\widehat{J} = \frac{(J+d-1)!}{(J-1)!\,d!} \tag{3.2}$$

finite volume cells to build a J-th degree polynomial approximation to $\mathbf{u}(\mathbf{x}, t)$. Structured meshes make p-refinement easier by allowing straightforward stencil size adjustments. However, general unstructured grids are more complex to handle [114, 115, 116]. Another drawback of larger stencils is the order reduction in the primal and dual solutions around the boundaries of the domain [29], where the stencil cannot be extended to allow sufficient accuracy (except in the case of periodic boundary conditions).

We are concerned with the accuracy of intergrid operators for finite volume solvers on structured meshes. For simplicity, we first consider one-dimensional problems; the analysis will be extended to higher dimensions in section 5.3.5. This one-dimensional discussion fully illustrates the accuracy issues encountered with adjoint (transposed) intergrid operators. Furthermore, we omit the time dependent notation, for ease of notation. The kth finite volume cell is $C^k = \left[x^{k-1/2},\, x^{k+1/2}\right]$, where $x^{k+1/2} = (x^{k+1} + x^k)/2$. Let $h^k = \mathrm{Vol}(C^k) := x^{k+1} - x^k$. This leads to the finite volume scheme for equation (5.18):

$$h^k \frac{\mathrm{d}\mathbf{U}^k}{\mathrm{d}t} + \mathbf{F}^{\mathrm{FVM}}_{k+1/2} - \mathbf{F}^{\mathrm{FVM}}_{k+1/2} = h^k\,\mathbf{f}^k \;,\; \forall\; k = 1 \;\ldots\; K \;. \tag{3.3}$$

The polynomial reconstruction approach approximates the solution through the interface between two control volumes as a polynomial with unknown coefficients:

$$\mathbf{u}^h(x) := \sum_{j=0}^{J} a_j (x - x^k)^j \;. \tag{3.4}$$

Note that we dropped the explicit time dependency in the numerical solution, since we assume this holds at some particular time point t^n. The a_j coefficients are determined by requiring that

$$\int_{C^j} \mathbf{u}^h(x) \;\; = \;\; \mathrm{Vol}(C^j)\,\mathbf{U}^j \;,\; \forall\, j = 0, \ldots, J \;.$$

The reconstruction may be either centered, or biased (upwind). We will show that the transpose relationship between the interpolation and restriction operators does not hold for operators with orders of accuracy higher than one. The adjoints of quadratic or higher order flux interpolations reduce to a simple first order conservative reconstruction. Hence, the adjoint solution cannot be expected to retain the order of accuracy of the forward discretization, due the order reduction in the intergrid solution transfer process. This is an additional drawback of the finite volume method, which negatively impacts the accuracy of black-box generation (via AD) of h-refinement operators in the discrete adjoint solver.

3.2 The model problems

We focus on the following two model problems:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial (\boldsymbol{\beta}\, \mathbf{u})}{\partial x} &= 0, \;\; x \in \Omega := [-1, 1] \; , \\ \mathbf{u}(t, -1) &= 0 \; , \; 0 \leq t \leq T \; , \\ \mathbf{u}(t, 1) &= 0 \; , \; 0 \leq t \leq T \; , \\ \mathbf{u}(0, \mathbf{x}) &= \mathbf{u}_1^0(\mathbf{x}) \; , \; \mathbf{x} \in \Omega \; . \end{aligned} \tag{3.5}$$

and

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\vec{\boldsymbol{\beta}} \mathbf{u}) &= 0 \; , \; \mathbf{x} \in \Omega := [0, 1] \times [0, 1] \; , \; 0 \leq t \leq T \; , \\ \mathbf{u}(t, \mathbf{x}) &= 0 \; , \; \mathbf{x} \in \Gamma := \partial \Omega \; , \; 0 \leq t \leq T \; , \\ \mathbf{u}(0, \mathbf{x}) &= \mathbf{u}_2^0(\mathbf{x}) \; , \; \mathbf{x} \in \Omega \; . \end{aligned} \tag{3.6}$$

The unknown primal solutions $\mathbf{u}(\mathbf{x}) \in \mathcal{L}^2([0, T] \times \Omega)$ are in both cases assumed to be continuously differentiable a.e. on Ω. We require for both (3.5) and (3.6) that

$$\mathbf{u}(t, \mathbf{x}) \geq 0 \; , \; \forall \mathbf{x} \in \Omega \; , \; \forall \; 0 \leq t \leq T \; . \tag{3.7}$$

with the possible exception of very small undershoots (which may be treated as zero values).

We have the following definition [117]:

Definition 3.2.1. *A numerical scheme for (3.1) is called positive if for any positive initial condition* $\mathbf{U}^{h,0} \geq 0$*, the numerical solution* $\mathbf{U}^{h,n}$ *remains non-negative at all times* $0 \leq t^n \leq T$*.*

Positivity of a numerical scheme is intimately connected with phenomena of overshoot and undershoot of the numerical solution (see [117] for details). Apart from consistency and stability, positivity is the essential property that we require from any numerical scheme used to discretize (3.5) and (3.6). Thus, any acceptable scheme may not allow the development of new local extrema in the numerical solution (other than the ones present in the initial condition $\mathbf{u}^{h,0}$).

Definition 3.2.2. *A scheme is monotonicity preserving if, for all time levels* n*,*

$$\mathbf{U}_n^k \geq \mathbf{U}_n^{k+1} \; , \; \forall \, k \implies \mathbf{U}_{n+1}^k \geq \mathbf{U}_{n+1}^{k+1} \; , \; \forall \, k \; . \tag{3.8}$$

The celebrated Godunov theorem restricts the spatial order of monotonicity preserving schemes to one.

Theorem 3.2.1 (*Godunov's order barrier theorem*)**.** *Consistent, stable, and monotone linear schemes for linear partial differential equations are at most first-order accurate.*

Proof. See, e.g., [112, Section 4.4]. □

When spatial and temporal accuracy higher than first order are necessary in the numerical solution algorithm, condition (3.7) and Theorem 3.2.1 impose the use of a high resolution method with flux limiting [1]. The flux limiter function becomes active in regions of high variation in the primal solution $\mathbf{u}^{h,n}$, preventing the occurrence of new extrema in the numerical solution. Flux limiter functions can be either active or inactive on a per-cell basis, as determined by the limiting algorithm.

We now proceed to investigate the effect of black-box automatic differentiation on flux limiting functions. The exact form of the boundary flux $F_{k+\frac{1}{2}}$ in equation (3.3) depends on the mesh stencil used for spatial discretization. Let $\mathbf{f}_k := \boldsymbol{\beta}\mathbf{U}^k$. The upwind ratio of two consecutive flux gradients can be then computed as:

$$\boldsymbol{\theta}_{k+\frac{1}{2}} = \frac{\mathbf{f}_{k+1} - \mathbf{f}_k + \boldsymbol{\epsilon}}{\mathbf{f}_k - \mathbf{f}_{k-1} + \boldsymbol{\epsilon}} \,, \tag{3.9}$$

with $\boldsymbol{\epsilon} \approx 10^{-14}$ a very small mesh-independent constant, that prevents division by zero in regions with uniform flow.

Let $\boldsymbol{\phi}(\theta)$ denote the limiter function. The general flux expression at the inflow boundary of the k-th finite volume cell reads [1]:

$$F_{k-\frac{1}{2}} = \mathcal{F}_L(f_k, f_{k-1}) + \boldsymbol{\phi}(\theta_{k-\frac{1}{2}})(\mathcal{F}_H(f_k, f_{k-1}) - \mathcal{F}_L(f_k, f_{k-1})) \,. \tag{3.10}$$

Here $\mathcal{F}_L$ is the flux given by a positive low-order scheme (such as first order upwind); $\mathcal{F}_H$ is the flux of a high-order numerical scheme, such as Lax–Wendroff [1], or third order upwind [117].

3.3 The discrete adjoint model

The one–dimensional problem If we consider the third order upwind scheme and the limiter function $\boldsymbol{\phi}(\theta)$ proposed by Hundsdorfer *et al.* [117], then the numerical solution of (3.5) is positive throughout the domain, with only some negligible $\mathcal{O}(\epsilon)$ undershoots. However, the discrete adjoint solution behaves unexpectedly. Denote the continuous adjoint variable $\boldsymbol{\lambda}(t, \mathbf{x})$. Then, $\boldsymbol{\lambda}$ satisfies the following final–value problem:

$$\begin{aligned} \frac{\partial \boldsymbol{\lambda}}{\partial t} + \boldsymbol{\beta}\frac{\partial \boldsymbol{\lambda}}{\partial x} &= 0 \,,\ \mathbf{x} \in [-1, 1] \ ,\ T \geq t \geq 0 \,, \\ \boldsymbol{\lambda}(t, -1) = \boldsymbol{\lambda}(t, 1) &= 0 \,,\ \forall\, T \geq t \geq 0 \,, \\ \boldsymbol{\lambda}(T, \mathbf{x}) &= \boldsymbol{\lambda}^F(\mathbf{x}) \,,\ \forall\, \mathbf{x} \in \Omega \,. \end{aligned} \tag{3.11}$$

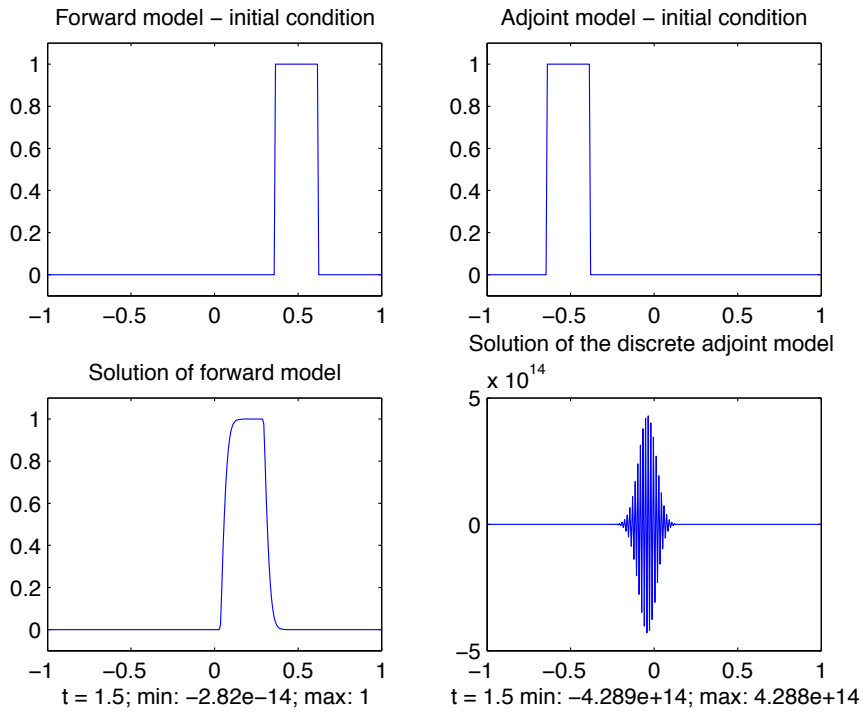


Figure 3.1: Unstable behavior for the discrete adjoint of flux-limited schemes implemented *by the book*.

Choose $\boldsymbol{\lambda}^F = \mathbf{u}(t^F, x)$, $\forall\, x \in [-1, 1]$. For constant $\boldsymbol{\beta}(\mathbf{x})$, we have $\boldsymbol{\lambda}(t, \mathbf{x}) = \mathbf{u}(t, \mathbf{x})$ for all times t.

Consider the discrete adjoint solutions. Let $\mathbf{u}(0, \mathbf{x})$ in (3.5) be a square wave of unit amplitude (see Figure 3.1). The advection coefficient $\boldsymbol{\beta} = -2$. The time steps are fixed through a mesh Courant number $\boldsymbol{\nu} = 0.76$. Unexpectedly, as Figure 3.1 shows, the discrete adjoint solution develops unstable oscillations after just a few time steps.

Let us seek the source of this instability. Let $\text{phi}_{\boldsymbol{\theta}}$ be the value of the limiter function for $\boldsymbol{\theta}$ given in (3.9). Similarly, adf_k is the adjoint of the $\mathbf{f}_k$ variable, and adflux_k is the adjoint of the boundary flux $\mathbf{F}_{k+1/2}$. The flux limiter function has the following form [117]:

$$\boldsymbol{\phi}(\theta) = \max\left(0, \min\left(2\,\boldsymbol{\theta},\, \min\left(2, \frac{1}{3} + \frac{2}{3}\,\boldsymbol{\theta}\right)\right)\right) \;. \tag{3.12}$$

With our notation

$$\text{phi}_\theta = \boldsymbol{\phi}\left(\frac{\mathbf{f}_k - \mathbf{f}_{k+1} + \epsilon}{\mathbf{f}_{k+1} - \mathbf{f}_{k+2} + \boldsymbol{\epsilon}}\right) \;. \tag{3.13}$$

This formulation is not suitable for direct differentiation. Since differentiation of intrinsic functions such as `max`, `min` or `abs` can result in incorrect code [118], we write (3.12) as an

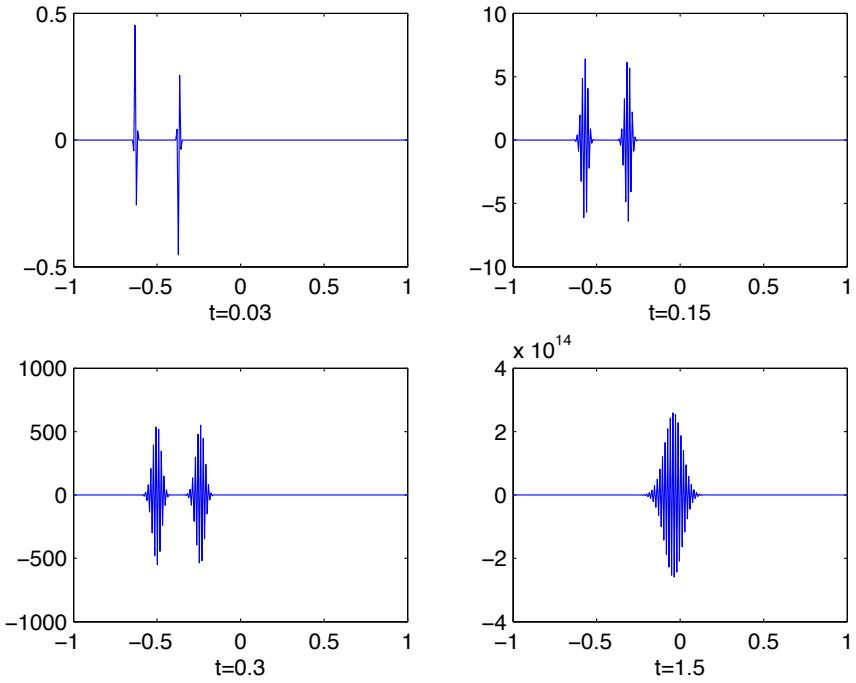


Figure 3.2: Instabilities in the adjoint fluxes at various time moments. The spurious modes are not damped by the discrete adjoint scheme. $K_n = 128$, $\tau^n = 0.03$, and Courant number $\nu = 0.765$.

if–then–else expression, and differentiate it in this form. Assume a negative velocity field u (the case when $u > 0$ is similar). If we examine the differentiated code, we notice the following statements that compute the adjoints of $\mathbf{f} \equiv \boldsymbol{\beta}\,\mathbf{U}$:

$$\begin{aligned} \mathrm{adf}_{k+2} &= \mathrm{adf}_{\mathrm{k+2}} - 0.5 \cdot \mathrm{adflux}_{\mathrm{k}} \cdot \mathrm{phi}_{\boldsymbol{\theta}} \\ \mathrm{adf}_{k+1} &= \mathrm{adf}_{\mathrm{k+1}} + \mathrm{adflux}_{\mathrm{k}} \cdot (1 + 0.5 \cdot \mathrm{phi}_{\boldsymbol{\theta}}) \; . \end{aligned}$$

Initially $\mathrm{adf}_k = 0$, $\forall\ k$. At the first time step the values are updated twice, since we are using a two–stage Runge-Kutta integrator. It follows that for the cells where $\mathrm{adflux}_k \neq 0$, the variations in the adjoint variable adf are amplified by the multiplicative term containing phi_θ. This happens in the non-uniform regions of the adjoint model solution. High-amplitude spurious spikes are noticeable after a single time step around the discontinuity points of $\mathbf{u}^{h,n}$ and $\boldsymbol{\lambda}^{h,n}$, as shown in Figure 3.2. This excitation is being added at each subsequent time step. The discrete adjoint $\boldsymbol{\lambda}^{h,n}$ is computed using the adf values so it will essentially behave in the same way.

Refactoring of the forward code to avoid discrete adjoint instability. The discrete adjoint model is therefore unstable, failing to provide a positive, non-oscillatory solution sim-

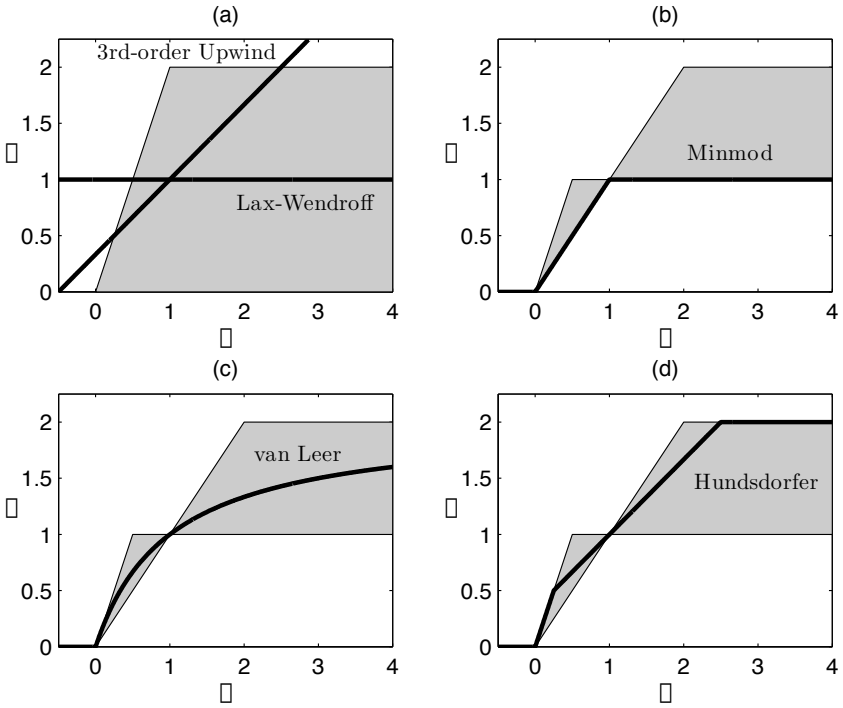


Figure 3.3: Limiter functions $\phi(\theta)$. *(a)* The shaded region is the TVD region of the θ-ϕ plane. Both Lax–Wendroff and the 3rd order upwind method have a $\phi(\theta)$ that are not contained in this region. *(b)* The shaded region is the second-order TVD region of Sweby [1]. The limiter function stays within this region (on its lower boundary). *(c),(d)* Both limiters are smooth at $\theta = 1$. This ensures full second-order accuracy.

ilar to the one obtained by running the forward code. To eliminate this undesired behavior, we need to examine the limiter function $\phi(\theta)$ in more detail. The goal is to remove the explicit dependence on θ (and implicitly on ϕ) of the forward model code. To do so, consider the graph of $\phi = \phi(\theta)$ for various values of θ in Figure 3.3. The solver is built upon the Hundsdorfer flux limiter function [117], and the third order upwind method.

Since $\phi(\theta)$ is a piecewise linear function, one can refactor the forward model code to take advantage of this property. Consider each interval on which $\phi(\theta)$ is linear. The exact expression of the flux function inside that interval, is straightforward to compute as a weighted sum of f_k. We are thus able to remove the explicit dependency on θ that causes the numerical instability in the discrete adjoint solution $\boldsymbol{\lambda}^{h,n}$.

If $\mathbf{u} > 0$, we obtain the following:

$$\begin{aligned}
&\text{If } \theta \le 0 && \Rightarrow \quad \mathbf{F}_{k+\frac{1}{2}} = f_k \\
&\text{Else if } \theta \le 0.25 && \Rightarrow \quad \mathbf{F}_{k+\frac{1}{2}} = f_{k+1} \\
&\text{Else if } \theta \le 2.5 && \Rightarrow \quad \mathbf{F}_{h+\frac{1}{2}} = f_h + \tfrac{f_k - f_{k-1}}{6} + \tfrac{f_{k+1} - f_k}{3} \\
&\text{Else} && \Rightarrow \quad \mathbf{F}_{k+\frac{1}{2}} = 2\, f_k - f_{k-1}\ .
\end{aligned}$$

It is useful to examine the adjoint code generated by a source–to–source AD tool such as TAMC [58]. Using the same notation as in (3.14), the discrete adjoint code reads:

$$\begin{aligned}
&\text{If } \theta \le 0 && \Rightarrow \quad \mathrm{adf}_k \leftarrow \mathrm{adf}_k + \mathrm{adflux}_k \\
&\text{Else if } \theta \le 0.25 && \Rightarrow \quad \mathrm{adf}_{k+1} \leftarrow \mathrm{adf}_{k+1} + \mathrm{adflux}_k \\
&\text{Else if } \theta \le 2.5 && \Rightarrow \quad \mathrm{adf}_{k-1} \leftarrow \mathrm{adf}_{k-1} - \tfrac{1}{6}\,\mathrm{adflux}_k \\
&&& \qquad\ \ \mathrm{adf}_{k+1} \leftarrow \mathrm{adf}_{k+1} + \tfrac{1}{3}\,\mathrm{adflux}_k \\
&&& \qquad\ \ \mathrm{adf}_k \leftarrow \mathrm{adf}_k - \tfrac{5}{6}\,\mathrm{adflux}_k \\
&\text{Else} && \Rightarrow \quad \mathrm{adf}_{k-1} \leftarrow \mathrm{adf}_{k-1} - \mathrm{adflux}_k \\
&&& \qquad\ \ \mathrm{adf}_k \leftarrow \mathrm{adf}_k + 2\,\mathrm{adflux}_k\ .
\end{aligned}$$

The second branch (for $\theta \in (0, 0.25]$) in (3.14) gives an unstable *downwind* approximation of the flux. However, the influence of this downwind flux is only local, and any oscillations that arise are quickly damped by the scheme. The final solution of the forward model has the desired properties: $\mathbf{u}^{h,n}$ both sharp and positive: any undershoots or overshoots that occur have an amplitude of order $\mathcal{O}(\epsilon)$. The same holds true for the discrete adjoint model: branches 2 and 4 are unstable, but due to the fact that the stable branch 3 is taken most frequently (in uniform flow regions we have $\theta \approx 1$, so branch 3 is favored), no instabilities develop and we obtain a good approximation to the exact adjoint model solution.

After the forward model has been differentiated, we notice that the explicit dependency of the adjoint code on $\mathrm{phi}_{\boldsymbol{\theta}}$ has disappeared. This is reflected in the results of a forward and adjoint model run in Figure 3.4: no instability is present and the adjoint solution is very similar to the forward model solution of an *unlimited* third order upwind scheme. Since the adjoint flux formulation depends on the value of $\boldsymbol{\theta}$ from the forward model formulation, and not on the differentiated values of the gradient ratio, the flux limiter will be active only in regions of strong variation of the forward solution. In regions of smooth flow, branch number

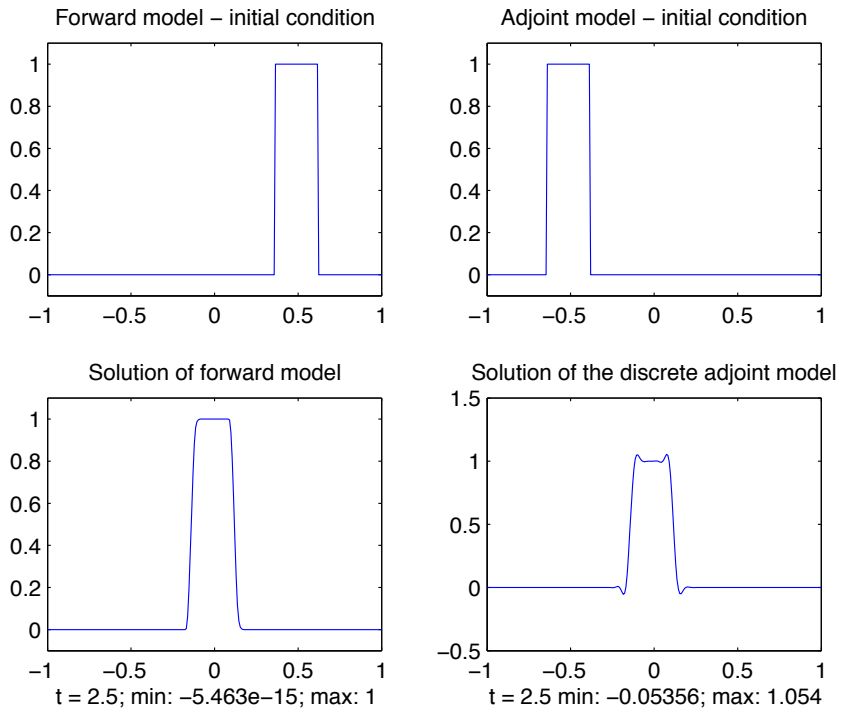


Figure 3.4: After the reorganization of the forward code, the discrete adjoint solution of the flux-limited scheme is stable and similar to an unlimited forward model solution. $K_n = 256$, and $\tau^n = 0.01$.

3 ($\boldsymbol{\theta} \in (0.25, 2.5]$) is taken. This yields a stable discrete adjoint solution shown in Figure 3.4.

Examine the variation of adf in this case. Figure 3.5 shows that oscillations in adf remain bounded as $t \to 0$. This does not come as a surprise, since we already noticed a well-behaved solution of the adjoint equation (3.11).

The two dimensional problem Consider now the two-dimensional problem (3.6). Our test scenario consists of the Molenkamp–Crowley test of solid body rotation. In equation (3.6), let $\vec{\boldsymbol{\beta}}$ be a rotating velocity field with angular velocity $\boldsymbol{\omega} = \pi$, described by the following equation:

$$\vec{\boldsymbol{\beta}}(x, y) = \begin{bmatrix} \boldsymbol{\beta}_x(x, y) \\ \boldsymbol{\beta}_y(x, y) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega} y \\ -\boldsymbol{\omega} x \end{bmatrix} . \tag{3.14}$$

We use the same initial conditions for both discrete forward and adjoint models (see Figure 3.6). If the boundary fluxes used in the forward numerical scheme depend explicitly on the gradient ratios $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$, the adjoint model solution develops strong oscillations after just a few time steps. These oscillations rapidly increase in amplitude at every time step.

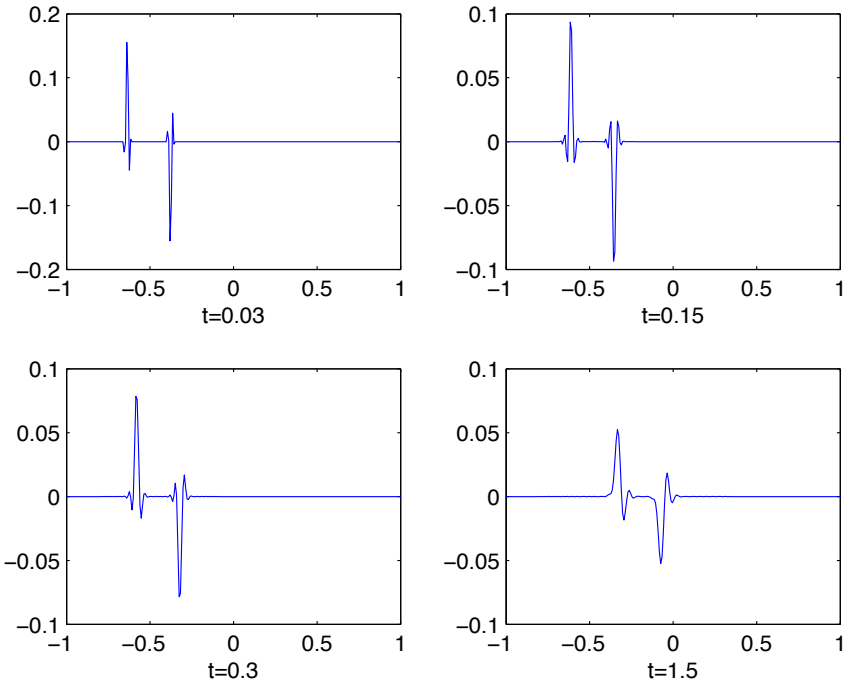


Figure 3.5: Stable adjoint fluxes. The adjoint model damps out all oscillations. $K_n = 128$, $\Delta t = 0.03$, and $\nu = 0.765$.

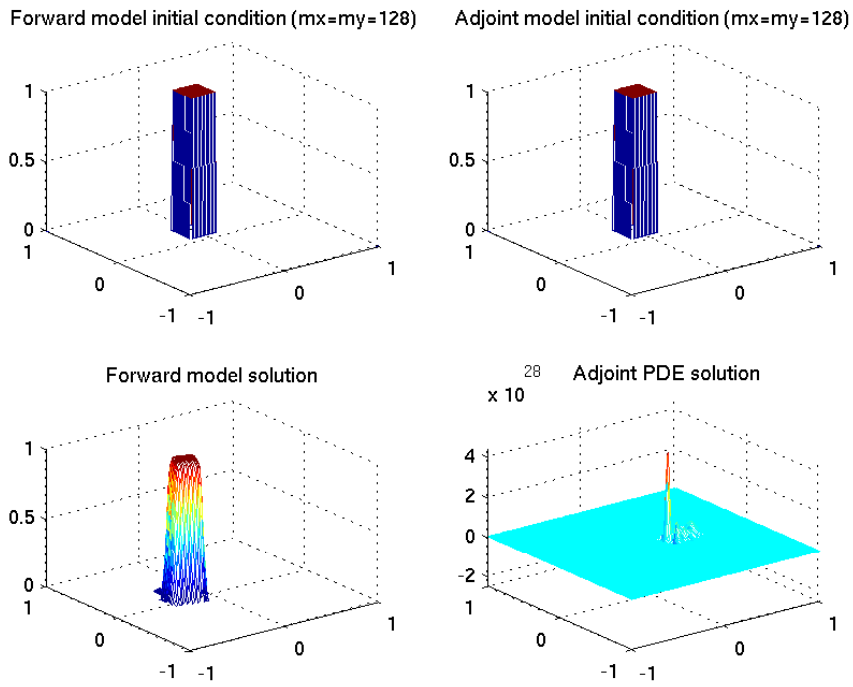


Figure 3.6: Unstable discrete forward and adjoint solutions to (3.6). $K_n = 128^2$, and $t = 0.25$. Unstable modes in the discrete adjoint $\boldsymbol{\lambda}^{h,n}$ display a very fast growth.

If we once more eliminate the explicit dependence $\mathbf{F} = \mathbf{F}(\boldsymbol{\theta})$ through the same procedure presented above, the results in Figure 3.7 illustrate the expected behavior: the solution of 3.6 remains positive and the initial profile is reproduced quite accurately, while the adjoint solution is strongly affected by spurious highly oscillatory modes. This is due to the fact that the adjoint model ceases to be a consistent approximation of the continuous adjoint equation in regions where the velocity field changes pattern [119]. Nevertheless, these oscillations do not grow unbounded, and one can still retrieve useful sensitivity information from the discrete adjoint profile.

The next numerical experiment uses the *minmod* limiter in Figure 3.3(b) ([1]) with a second-order Lax–Wendroff numerical scheme for (3.5). For $\mathbf{u} \geq 0$, the flux at the right boundary of the kth cell is a function of the Courant number ν:

$$\mathcal{F}_{LW} = \mathbf{f}_k + 0.5(1 - \boldsymbol{\nu})\mathbf{f}_{k+1} - 0.5(1 - \boldsymbol{\nu})f_k \ . \tag{3.15}$$

Using the piecewise linearity of the *minmod* function one can write the following implementation of the limiting algorithm:

$$\begin{aligned}
&\text{If } \theta \leq 0 && \Rightarrow \quad F_{k+\frac{1}{2}} = \mathbf{f}_k \\
&\text{Else if } \theta \leq 1 && \Rightarrow \quad F_{k+\frac{1}{2}} = \mathbf{f}_k + \tfrac{1-\nu}{2}\,\mathbf{f}_k - \tfrac{1-\nu}{2}\,\mathbf{f}_{k-1} \\
&\text{Else} && \Rightarrow \quad F_{i+\frac{1}{2}} = \mathbf{f}_i + \tfrac{1-\nu}{2}\,\mathbf{f}_{k+1} - \tfrac{1-\nu}{2}\,\mathbf{f}_k \ .
\end{aligned} \tag{3.16}$$

The results of a forward and adjoint model run are illustrated in Figure 3.8. The forward and adjoint solutions are virtually identical. The discrete solutions are stable and positive for all $0 \leq t \leq T$. This is due to the fact that the branch most frequently taken in the adjoint scheme gives a positive solution. The trade-off is that the wave profile is not so sharp: there is more numerical diffusion, since the Lax–Wendroff method is only second-order accurate. By examining the adjoint code in (3.17), we notice that the source of the instabilities has disappeared:

$$\begin{aligned}
&\text{If } \theta \leq 0 && \Rightarrow \quad \mathrm{adf}_k \leftarrow \mathrm{adf}_k + \mathrm{adflux}_k \\
&\text{Else if } \theta \leq 1 && \Rightarrow \quad \mathrm{adf}_{k-1} \leftarrow \mathrm{adf}_{k-1} - \tfrac{1-\nu}{2}\,\mathrm{adflux}_k \\
& && \qquad\quad \mathrm{adf}_i \leftarrow \mathrm{adf}_k + \tfrac{3-\nu}{2}\,\mathrm{adflux}_k \\
&\text{Else} && \Rightarrow \quad \mathrm{adf}_{k+1} \leftarrow \mathrm{adf}_{k+1} + \tfrac{1-\nu}{2}\,\mathrm{adflux}_k \\
& && \qquad\quad \mathrm{adf}_k \leftarrow \mathrm{adf}_k + \tfrac{1+\nu}{2}\mathrm{adflux}_k \ .
\end{aligned} \tag{3.17}$$

We now extend the technique shown above to flux limiter functions that are not piecewise linear. This requires the construction of piecewise linear approximations to ϕ, that are then

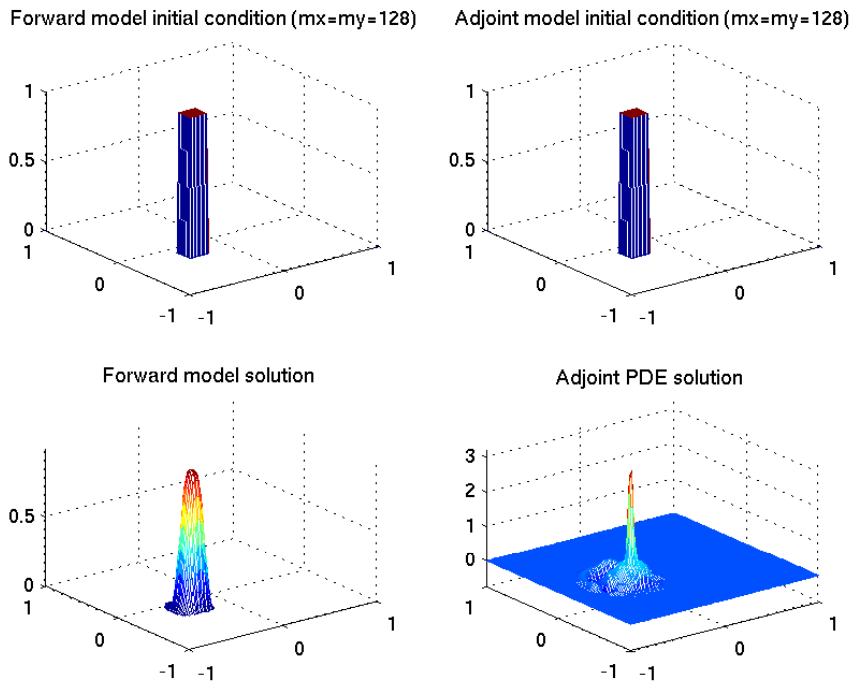


Figure 3.7: Stable discrete forward and adjoint solutions to (3.6). After a full rotation of the initial profile, we notice spurious modes in the numerical adjoint. They are caused by the point-wise inconsistency of the discrete adjoint formula with the continuous formulation at the mesh nodes where $\vec{\boldsymbol{\beta}}$ changes direction. However, these wiggles remain bounded as $t \to 0$.

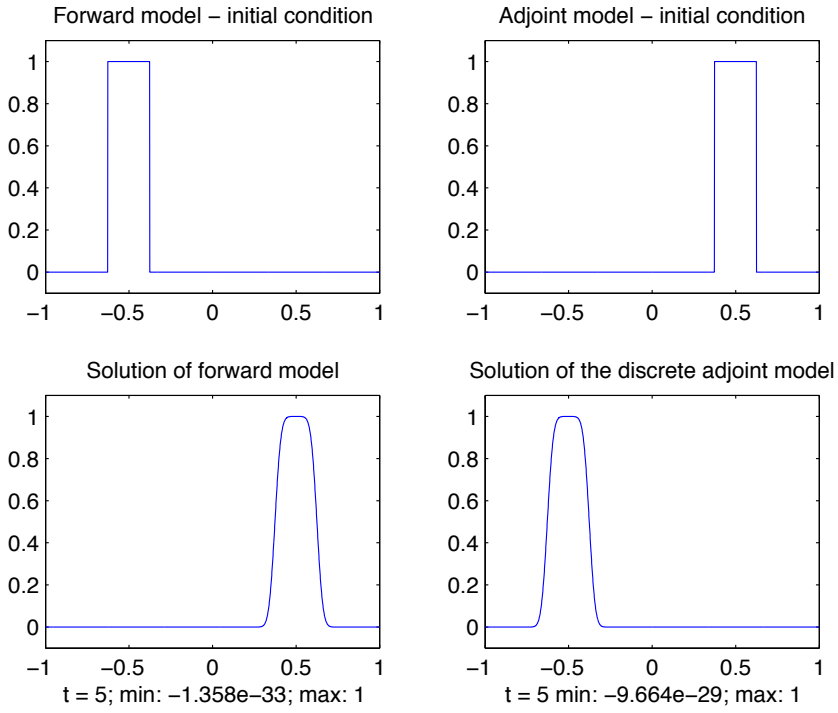


Figure 3.8: Forward model and adjoint model solutions of Lax–Wendroff scheme with the *minmod* limiter. $K_n = 2048$. We have more numerical diffusion than in Figure 3.4, but here the adjoint solution is both stable and positive, with negligible undershoots.

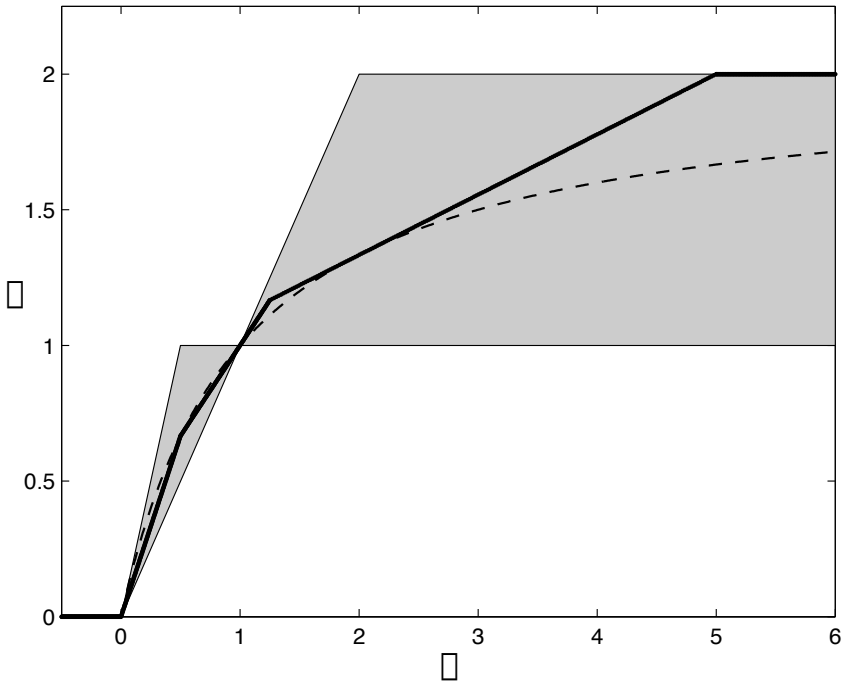


Figure 3.9: van Leer's limiter: exact formulation (dashed line) and a piecewise linear approximation (solid line). Note that the approximation lies in the TVD region and is smooth in the neighborhood of $\boldsymbol{\theta} = 1$.

used in the flux computations. The TVD properties and smoothness properties around $\theta = 1$ need to be preserved. To illustrate this approach, consider the well-known example of van Leer [120]:

$$\boldsymbol{\phi}(\boldsymbol{\theta}) = \frac{\boldsymbol{\theta} + |\boldsymbol{\theta}|}{1 + \boldsymbol{\theta}} \tag{3.18}$$

Figure 9 shows a piecewise linear approximation of the van Leer limiter function that is confined to Sweby's second order TVD region and is also smooth around $\boldsymbol{\theta} = 1$. The slope of the linear approximation around $\boldsymbol{\theta} = 1$ is equal to that of the third order upwind method slope, so the scheme will be fully accurate around that point. This is desirable since $\boldsymbol{\theta} \approx 1$ indicates that we are in a uniform flow region.

The discrete adjoint solution behaves similarly to the limited third-order upwind scheme from Figure 3.4. We do have undershoots and overshoots (approximately 9% in our test case). This is due to the fact that the branch most frequently taken by the adjoint code corresponds to $\boldsymbol{\theta} \approx 1$, i.e., the third-order upwind local approximation, which is an oscillatory scheme. However, $\boldsymbol{\lambda}^{h,n}$ remains stable and fairly accurate. Hence, useful sensitivity information can still be inferred from the discrete dual solution.

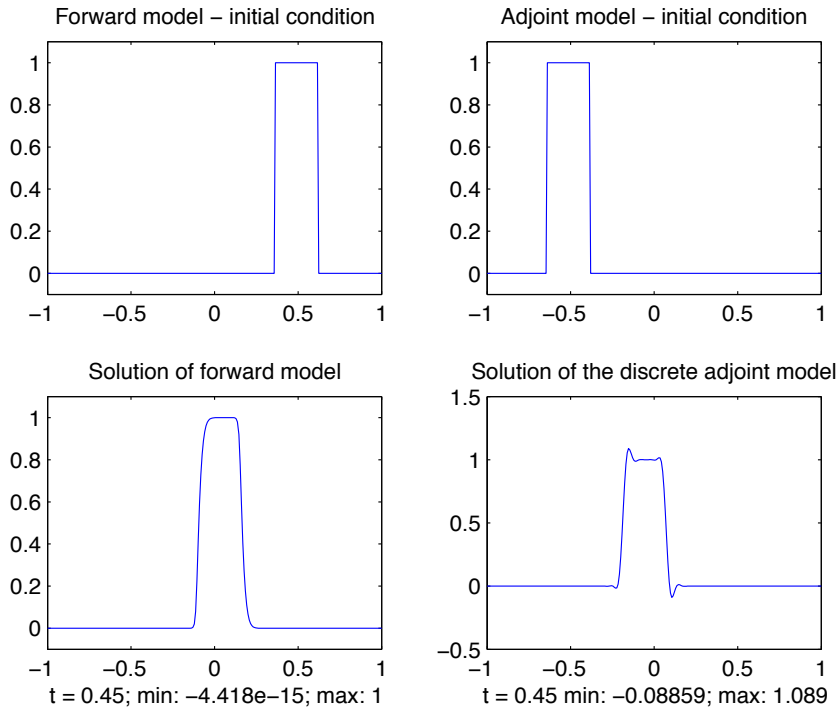


Figure 3.10: Discrete solutions $\mathbf{u}^{h,n}$ and $\boldsymbol{\lambda}^{h,n}$ obtained with a piecewise linear approximation of van Leer's flux limiter function. The solution is similar to the one in Figure 3.3. $K_n = 256$, and $\Delta t = 0.002$.

3.4 Conclusions

We have investigated the problem of constructing a stable discrete adjoint model of a flux-limited numerical scheme used to solve the one- and two-dimensional advection equation. Explicit dependencies of the boundary fluxes $\mathbf{F}_k$ on the flux gradient ratios $\boldsymbol{\theta}$ lead to numerical instabilities in the adjoint model solution. These are due to the fact that the differentiated value of $\boldsymbol{\theta}$ leads to nonphysical growth in the differentiated flux function. The piecewise linearity of the flux limiter functions $\boldsymbol{\phi}(\theta)$ allows a re-factoring of the discrete forward model code, such that these dependencies are removed, and stability is preserved. This is readily extended to nonlinear limiter functions (e.g. van Leer's limiter) by constructing a piecewise linear approximation of $\boldsymbol{\phi}$ and using it in our flux computations. Of course, for this procedure to be effective, the piecewise linear limiter must preserve the TVD property and preferably also the smoothness at $\boldsymbol{\theta} = 1$ of the nonlinear flux function $\boldsymbol{\phi}$ that it approximates.

Chapter 4

Inverse problems with time adaptivity

Introduction This chapter discusses adaptive time integration algorithms for inverse problems. A majority of inverse problems encountered in geophysics or the atmospheric sciences are time dependent, that is, they require the inversion of a dynamical model that depends on time as well as space. Thus, some temporal integration using numerical quadrature is a required part of any solution algorithm. This raises several complications that were not present with space-only discretizations. An essential component of solution procedure is the adjoint of the forward time stepping procedure. This reversal of the time integration will produce the gradient of the target functional under consideration with respect to the given control variables.

Again, there are two inversion approaches. The differentiate – then – discretize method, explored in the next section, implies the analytical derivation of the time-dependent adjoint equation, together with appropriate final conditions. This is a procedure that cannot be done automatically. Once the adjoint and quadrature equations have been derived, adaptive time integration together with dense-output based polynomial interpolation may be used for a highly efficient reversal of nonlinear models. The DENSERKS software [121] implements several well-known Runge–Kutta embedded pairs with adaptive step control. The implementations are shown to be competitive with established software for sensitivity analysis, such as DASPKADJOINT [31, 3], and Sundials CVODES [2].

The second section in this chapter explores the alternative route: discretize – then – differentiate. This approach is very attractive in practice due to the use of automatic differentiation, which allows the derivation of adjoint models for time dependent problems with relatively low effort (compared to the hand-coded approach above). Moreover, it relies on the dual consistency results for Runge–Kutta discretizations [122]: the discrete adjoints of Runge-Kutta schemes are known to have the full accuracy of the forward method.

However, there are several caveats with this approach. The black-box application of AD on adaptive time integration routines may generate erroneous values for the adjoint gradients,

if care is not taken. For discrete tangent linear models, the AD mechanism may pick up the dependency between the solution and the time step (through the error estimation and control mechanism), and generate spurious gradients that corrupt the accuracy of the sensitivity solutions [4]. We explore the effects of reverse mode AD on time-adaptive Runge–Kutta methods in the second section of this chapter. Similar to the forward approach, unphysical derivatives appear due to nonlinear dependencies between the forward numerical solution y^n, the time step h^n, and time points t^n. When passivation of the error mechanism is not an option, post-processing of the adjoint code is required to restore full accuracy in the adjoint solution.

4.1 The *differentiate – then – discretize* approach

This section focuses on sensitivity analysis time-dependent models described by ordinary differential equations (ODEs). The sensitivity analysis framework under consideration comprises continuous forward, tangent linear, and adjoint models of Runge-Kutta methods. While the discrete adjoints of explicit Runge-Kutta methods are consistent with the continuous adjoint equations [33, 122, 95], and automatic differentiation (AD) tools such as `TAMC` [58], `TAF` [57], and `TAPENADE` [54] considerably speed up the adjoint code generation, the code obtained through AD is frequently sub-optimal. Moreover, hand-coded modifications of adaptive discrete adjoint codes are normally required in order to guarantee their correctness [4, 121].

Adjoints of nonlinear models depend on the original model trajectory. Since adjoint models are integrated backward in time, they will require an approximation of the forward model solution at time points that cannot be known a priori. Some form of interpolation is necessary to provide these approximations. Existing software uses Hermite polynomial interpolation of up to 5th order. This is insufficient if one needs a higher order of accuracy for the adjoint solution. However, continuous extensions of Runge-Kutta schemes [123, 124, 125] have built in interpolants that are as accurate as the numerical schemes themselves. We propose to use this dense output mechanism to interpolate the original model solution. This approach is found to lead to a highly-accurate adjoint solution, with only a small increase in the computational cost of the forward integrator.

Our Fortran library, `DENSERKS`, implements several well-known continuous Runge-Kutta pairs for forward and adjoint sensitivity analysis. A two-level file checkpointing mechanism is used to minimize the solver memory requirements. This makes long term adjoint simulations feasible. Also, unlike other publicly available software, `DENSERKS` can directly be used for second order adjoint sensitivity analysis. Second order adjoint variables provide more accurate sensitivity information. They are computed most efficiently through forward-over-adjoint differentiation of the original model code [69]. A numerical experiment in section 4.1.4 discusses the computational and numerical advantages of the second order adjoint approach over the finite difference method.

4.1.1 Related work

A significant effort has been invested in the development of efficient sensitivity solvers, and currently there are several high quality, publicly available implementations. `SUNDIALS` [126] is a suite of ODE solvers with forward and first order adjoint sensitivity analysis capabilities. The `CVODES` package [30], part of `SUNDIALS`, is a collection of sensitivity-enabled ODE solvers. `CVODES` users can choose between backward differentiation schemes or Adams–Moulton methods for forward, tangent linear and adjoint model integrations. Either cubic Hermite interpolation or variable-order polynomial interpolation is used for approximating the forward model solution during the adjoint problem run [2].

Cao, Li and Petzold designed and implemented software for both the forward and adjoint sensitivity analysis of differential-algebraic equations (DAEs) with index up to 2 [46, 31]. Both their `DASPK` and `DASPKADJOINT` [31, 3] packages use variable-order backward differentiation formulas to solve the DAE sensitivity systems. Sandu et. al. [94] discuss the implementation of implicit Runge-Kutta and Rosenbrock methods and their discrete and continuous adjoints. Third order Hermite interpolation is used to approximate the original model solutions at the points required in the continuous adjoint model solvers. Their code is integrated into the Kinetic PreProcessor (`KPP`) software for solving chemical kinetics [127, 128, 129].

4.1.2 Theoretical background

Runge-Kutta schemes

Consider a system modeled by the following initial-value problem, henceforth referred to as the *forward model*:

$$\begin{aligned} \dot{\mathbf{y}} &= F\big(t, \mathbf{y}\left(t, \mathbf{q}\right), \mathbf{q}\big), \quad t^0 \leq t \leq t^F\,, \\ \mathbf{y}(t^0) &= \mathbf{y}^0\left(\mathbf{q}\right)\,, \end{aligned} \tag{4.1}$$

where $\mathbf{y}(t, \mathbf{q}) \in \mathbb{R}^{n_y}$ is the state vector, $\mathbf{q} \in \mathbb{R}^{n_q}$ denotes a vector of system parameters, and $F : \mathbb{R}^{1+n_y+n_q} \to \mathbb{R}^{n_y}$ is a prescribed function. We assume that (4.1) has a unique solution $\mathbf{y} = \mathbf{y}(t)$, and that F is twice continuously differentiable with respect to both $\mathbf{y}$ and $\mathbf{q}$, and for all $t^0 \leq t \leq t^F$.

Runge-Kutta schemes are a well-known class of numerical methods for solving initial value problems (IVPs) like (4.1). We focus on embedded explicit Runge-Kutta pairs. Let $\mathbf{y}^n \approx \mathbf{y}(t^n)$ be the numerical solution at time t^n. An s-stage embedded explicit Runge-Kutta pair

computes two approximations to $\mathbf{y}(t^{n+1})$ using the formulas:

$$\begin{aligned} \mathbf{y}^{n+1} &= \mathbf{y}^n + h^n \sum_{j=1}^{s} b_j \mathbf{k}_j^n \\ \widehat{\mathbf{y}}^{n+1} &= \mathbf{y}^n + h^n \sum_{j=1}^{s} \widehat{b}_j \mathbf{k}_j^n \ , \end{aligned} \tag{4.2}$$

where

$$\mathbf{k}_j^n = F\left(t^n + h^n c_j,\, \mathbf{y}^n + h^n \sum_{i=1}^{j-1} a_{ij} \mathbf{k}_i^n,\, \mathbf{q} \right) , \tag{4.3}$$

and $t^{n+1} = t^n + h^n$.

Let p be the order of accuracy of the Runge-Kutta pair (4.2). For a sufficiently smooth solution $\mathbf{y}(t, \mathbf{q}) \in \mathrm{C}^p$, we expect that

$$\mathbf{y}^n - \mathbf{y}(t^n) = \mathcal{O}(h^p) \ , \tag{4.4}$$

where $h = \max_{1 \le i \le n} h^i$. Both solution approximations $\mathbf{y}^{n+1}$ and $\widehat{\mathbf{y}}^{n+1}$ use the same function values (stages) $\mathbf{k}_j^n$. However, they have different orders of accuracy, depending on the particular choice of method coefficients a_{ij}, b_j, $\widehat{b}_j$ and c_j. Here y^{n+1} is accurate of order p and is used to continue the time integration. The second approximation, $\widehat{\mathbf{y}}^{n+1}$, has an order of accuracy less than p, and helps to estimate the local truncation error of the scheme as

$$\mathbf{e}^{n+1} = \mathbf{y}^{n+1} - \widehat{\mathbf{y}}^{n+1} \, . \tag{4.5}$$

This error estimate is used to automatically control the integration time step size (see Hairer *et al.* [124, Section II.4] for further details on the topic).

Dense Output for Runge-Kutta pairs

In many applications one needs the approximate solution at certain prescribed time points. It is highly inefficient to force the Runge-Kutta routine to compute those approximations, since this would impose unnecessary constraints on the size of the integration time steps. This situation motivated the construction of *dense output* formulas [124]. Given a numerical solution $\mathbf{y}^n$, the dense output formulas yield cheap numerical approximations to the exact solution $\mathbf{y}(t^n + \theta h^n)$, where θ usually satisfies $0 \le \theta \le 1$. Extrapolations with $\theta > 1$ are possible. However, in this case the approximations are often less accurate, and will not be considered further.

For most s-stage high-order Runge-Kutta schemes, one needs to append $s^* - s$ extra stages in order to accommodate the dense output interpolant. Thus, the performance penalty incurred

by the use of dense output formulas is equal to the cost of a few extra function evaluations per time step.

A p^*-th order dense output formula for (4.2) has the form

$$\mathbf{u}(\theta) = \mathbf{y}^n + h^n \sum_{j=1}^{s^*} \tilde{b}_j(\theta)\mathbf{k}_j^n \,, \tag{4.6}$$

where the $\mathbf{k}_j^n$ are defined in (4.3), and $\tilde{b}_j(\theta)$ are polynomials in θ determined such that

$$\mathbf{u}(\theta) - \mathbf{y}(t^n + h^n\theta) = \mathcal{O}\left((h^n)^{p^*+1}\right) \,. \tag{4.7}$$

Following Hairer et. al. [124], we consider the time interval $[t^n, t^{n+1}]$ (with t^n assumed to be sufficiently far from the initial time t^0), and denote by $\mathbf{z}(x)$ the solution of the local IVP

$$\begin{aligned} \dot{\mathbf{z}} &= F(t, \mathbf{z}, \mathbf{q}), \quad t^n \le t \le t^{n+1} \,, \\ \mathbf{z}(t^n) &= \mathbf{y}^n \,. \end{aligned} \tag{4.8}$$

Then the error of the dense output formula can be written as:

$$\mathbf{u}(\theta) - \mathbf{y}\left(t^n + h^n\theta\right) = \underbrace{\left[\mathbf{u}(\theta) - \mathbf{z}(t^n + h^n\theta)\right]}_{\mathcal{O}\left((h^n)^{p^*+1}\right)} + \underbrace{\left[\mathbf{z}(t^n + h^n\theta) - \mathbf{y}(t^n + h^n\theta)\right]}_{\mathcal{O}((h^n)^p)} \,. \tag{4.9}$$

The first term $(\mathbf{u} - \mathbf{z})$ in the right hand side of (4.9) is the interpolation error, and has magnitude $\mathcal{O}\left((h^n)^{p^*+1}\right)$. The second term $(\mathbf{z} - \mathbf{y})$ denotes the global error of the method, therefore is of order $\mathcal{O}\left((h^n)^p\right)$. Thus, to guarantee an order-p dense output approximation to $\mathbf{y}(t^n + \theta h^n)$, it suffices to require that

$$p^* \ge p - 1 \,. \tag{4.10}$$

For $p \le 4$, cubic Hermite interpolation is sufficiently accurate. However, for larger values of p, performing polynomial interpolation while preserving the number of stages becomes an increasingly cumbersome process, and the quality of the interpolated solution depends on the choice of interpolation points. Continuous Runge-Kutta schemes allow for an efficient interpolation, with only a modest increase in the computational cost coming from the $s^* - s$ additional stages incorporated in (4.2). Since the accuracy constraints on the dense output coefficients usually allow for one or more degrees of freedom, one selects the coefficients of the polynomials $\tilde{b}_j(\theta)$ such that a certain error norm or cost function is minimized (see, e.g., [123]). Various dense output methods for constructing high-order interpolants have been proposed, notably by Sharp and Verner [125], based on the bootstrapping scheme of Verner [130].

Finally, we note that the formulation of the tangent linear and adjoint ODEs described in the next section is evidently independent of the numerical method used to solve those equations. All that is needed is that the numerical method be fitted with a continuous extension, to allow an efficient on the fly interpolation of the forward trajectory during the adjoint model integration. Rosenbrock methods, extrapolation methods, as well as implicit Runge-Kutta methods, support continuous extensions [131, 132, 133]. Thus, the sensitivity analysis framework described herein can be seamlessly extended to such methods.

First order sensitivity analysis

We are interested in solving the following optimal control problem:

$$\text{Find } \mathbf{q}_* = \min_{\mathbf{q}} \mathcal{J}(\mathbf{y}, \mathbf{q}) \text{ , subject to (4.1) ,} \tag{4.11}$$

where the cost functional $\mathcal{J}$ is defined as

$$\mathcal{J}(\mathbf{y}, \mathbf{q}) \;=\; \int_{t^0}^{t^F} \mathbf{g}\big(t, \mathbf{y}(t, \mathbf{q}), \mathbf{q}\big)\, dt\,. \tag{4.12}$$

Here $\mathbf{g} : \mathbb{R}^{n_y+1} \times \mathbb{R}^{n_q} \to \mathbb{R}$ is a given real-valued function. Most optimization algorithms require some type of derivative information to build successive approximations to the optimal solution. If first order derivatives are required, we compute the reduced gradient:

$$\nabla_{\mathbf{q}} \mathcal{J} = \frac{\partial \mathcal{J}}{\partial \mathbf{q}}\,. \tag{4.13}$$

It has been shown [30, 46, 129] that (4.13) can be computed from the solutions $\boldsymbol{\tau}_i(t) = \partial \mathbf{y}/\partial \mathbf{q}_i\,(t)$ to the *tangent linear* model (TLM), i.e., the linearized primal:

$$\begin{aligned} \dot{\boldsymbol{\tau}}_i &= F_{\mathbf{y}}(t, \mathbf{y})\boldsymbol{\tau}_i + F_{\mathbf{q}_i}\,, \;\; t^0 \le t \le t^F\,, \\ \boldsymbol{\tau}_i(t^0) &= \frac{\partial \mathbf{y}^0}{\partial \mathbf{q}_i}\,, \quad i = 1 \ldots n_q\,. \end{aligned} \tag{4.14}$$

The gradient is computed component-wise by solving the following quadrature equations together with (4.14):

$$\begin{aligned} \dot{\boldsymbol{\zeta}}_i &= \mathbf{g}_{\mathbf{y}}^T\, \boldsymbol{\tau}_i + \mathbf{g}_{\mathbf{q}_i}\,, \\ \boldsymbol{\zeta}_i(t^0) &= 0 \;\;, \;\; i = 1 \ldots n_q\,. \end{aligned} \tag{4.15}$$

The i-th quadrature solution equals the i-th component of the gradient:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{q}_i} = \boldsymbol{\zeta}_i(t^F)\,. \tag{4.16}$$

Note that we have to solve (4.14)–(4.15) once per gradient component. The two models are usually integrated as a coupled system of $n_q + 1$ equations. Since $(\partial \mathbf{J}/\partial \mathbf{q}) \in \mathbb{R}^{n_q}$, n_q TLM and quadrature solutions are needed to obtain the gradient vector. Hence, the forward sensitivity method quickly becomes impractical for large n_q.

The adjoint method [69, 126, 46] significantly reduces the cost of a reduced gradient computation. It can be formulated as follows. Let the adjoint variable be $\boldsymbol{\lambda} \in \mathbb{R}^{n_y}$, and define the extended cost functional

$$\widehat{\mathcal{J}}(\mathbf{q}) \;=\; \mathcal{J}(\mathbf{q}) - \int_{t^0}^{t^F} \boldsymbol{\lambda}^T \left(\dot{\mathbf{y}} - F(t, \mathbf{y}, \mathbf{q})\right) \, dt \; . \tag{4.17}$$

From (4.1), $\widehat{\mathcal{J}}(\mathbf{q}) = \mathcal{J}(\mathbf{q})$. It follows that the sensitivity of $\widehat{\mathcal{J}}$ with respect to the parameters $\mathbf{q}$ is

$$\left(\frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{q}}\right)^T = \left(\frac{\partial \mathcal{J}}{\partial \mathbf{q}}\right)^T = \int_{t^0}^{t^F} (\mathbf{g}_\mathbf{y}^T \mathbf{y}_\mathbf{q} + \mathbf{g}_\mathbf{q}^T) - \int_{t^0}^{t^F} \boldsymbol{\lambda}^T (-F_\mathbf{q} - F_\mathbf{y} \mathbf{y}_\mathbf{q} + \dot{\mathbf{y}}_\mathbf{q}) \, dt \, . \tag{4.18}$$

Integration by parts leads to:

$$\left(\frac{\partial \mathcal{J}}{\partial \mathbf{q}}\right)^T \;=\; \int_{t^0}^{t^F} (\mathbf{g}_\mathbf{q}^T + \boldsymbol{\lambda}^T F_\mathbf{q}) \, dt - \int_{t^0}^{t^F} (-\mathbf{g}_\mathbf{y}^T - \boldsymbol{\lambda}^T F_\mathbf{y} - \dot{\boldsymbol{\lambda}}^T) \mathbf{y}_\mathbf{q} \, dt - (\boldsymbol{\lambda}^T \mathbf{y}_\mathbf{q})|_{t^0}^{t^F} \, . \tag{4.19}$$

To avoid the computation of the forward sensitivities $\mathbf{y}_\mathbf{q}(t)$ at $t > t^0$, one defines $\boldsymbol{\lambda} \in \mathbb{R}^{n_y}$ as the solution of the *first order adjoint model*, described by the following final-value problem:

$$\begin{aligned} \dot{\boldsymbol{\lambda}} \;&=\; -F_\mathbf{y}^T(t, \mathbf{y}) \boldsymbol{\lambda} - \mathbf{g}_\mathbf{y} \, , \;\; t^F \geq t \geq t^0 \, , \\ \boldsymbol{\lambda}(t^F) \;&=\; 0 \, . \end{aligned} \tag{4.20}$$

Then

$$\left(\frac{\partial \mathcal{J}}{\partial \mathbf{q}}\right)^T = \int_{t^0}^{t^F} (\mathbf{g}_\mathbf{q}^T + \boldsymbol{\lambda}^T F_\mathbf{q}) \, dt + \left(\boldsymbol{\lambda}^T \mathbf{y}_\mathbf{q}\right)\big|_{t=t^0} \; , \tag{4.21}$$

where the integral can be evaluated concurrently with (4.20) using the solution of a quadrature ODE:

$$\begin{aligned} \dot{\mathbf{w}}^T \;&=\; -\mathbf{g}_\mathbf{q}^T - \boldsymbol{\lambda}^T F_\mathbf{q} \, , \;\; t^F \geq t \geq t^0 \, , \\ \mathbf{w}^T(t^F) \;&=\; 0 \, . \end{aligned} \tag{4.22}$$

Hence,

$$\left(\frac{\partial \mathcal{J}}{\partial \mathbf{q}}\right)^T = \mathbf{w}^T(t^0) + \boldsymbol{\lambda}^T(t^0) \, \mathbf{y}_\mathbf{q}(t^0) \, . \tag{4.23}$$

Equations (4.20) and (4.22) are attractive since they give the entire gradient vector, at the expense of only a single first order adjoint and quadrature solve. However, the adjoint equation depends on the forward model trajectory through the Jacobian $F_{\mathbf{y}}(t, \mathbf{y})$. Moreover, (4.20) is integrated backward in time from t^F to t^0. Thus, solving for $\boldsymbol{\lambda}(t^0)$ and $\mathbf{w}(t^0)$ requires solving the forward model first, with a continuous Runge-Kutta method. The dense output interpolant is given by (4.6). The polynomials $\tilde{b}_j(\theta)$ are given by the continuous extension. To build $\mathbf{u}(\theta)$ on the fly during the backward run, the time steps h^n, the corresponding time points t^n, and the interpolation weights $\mathbf{k}_j^n$ all need to be stored. Then, in the backward time integration, (4.20) and (4.22) are solved simultaneously as a system of $n_y + n_q$ differential equations, using the interpolated forward solution $\mathbf{u}(\theta) \approx \mathbf{y}(t^n + h^n\theta)$ when needed.

Second order adjoint sensitivity analysis

In this section we obtain second-order derivative information for the cost function (4.12). For large-scale models, where n_y and n_q can reach into the millions (or more), computing the full Hessian is not computationally feasible. Instead, one calculates Hessian-vector products:

$$\left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{q}^2}\right) \cdot \boldsymbol{\delta}\mathbf{q} \,, \tag{4.24}$$

where $\delta\mathbf{q} \in \mathbb{R}^{n_q}$ is some perturbation in the time-independent parameters $\mathbf{q}$. A forward differentiation of (4.18) results in the *second order adjoint* final value problem [69, 72]:

$$\begin{aligned} \dot{\boldsymbol{\sigma}} &= -F_{\mathbf{y}}^T \boldsymbol{\sigma} - \left(F_{\mathbf{yy}} \odot (\mathbf{y_q}\boldsymbol{\delta}\mathbf{q})\right)^T \boldsymbol{\lambda} - \left(F_{\mathbf{yq}} \odot \boldsymbol{\delta}\mathbf{q}\right)^T \boldsymbol{\lambda} - \mathbf{g_{yy}}\left(\mathbf{y_q}\boldsymbol{\delta}\mathbf{q}\right) - \mathbf{g_{yq}}\boldsymbol{\delta}\mathbf{q} \;, \quad t^F \geq t \geq t^0 \,, \\ \boldsymbol{\sigma}(t^F) &= 0 \,, \end{aligned} \tag{4.25}$$

where $\boldsymbol{\sigma} \in \mathbb{R}^{n_y}$ is the *second order adjoint variable*,

$$F_{\mathbf{yy}} = \left(\frac{\partial^2 F_i}{\partial \mathbf{y}_j \partial \mathbf{y}_k}\right)_{i,j,k=1\ldots n_y} \,, \tag{4.26}$$

and the $\odot$ symbol denotes the tensor product

$$F_{\mathbf{yy}} \odot (\mathbf{y_q}\boldsymbol{\delta}\mathbf{q}) = \left(\sum_{k=1}^{n_y} \left(F_{\mathbf{yy}}\right)_{i,j,k} \left(\mathbf{y_p}\boldsymbol{\delta}\mathbf{q}\right)_k\right)_{i,j=1\ldots n_y} \,. \tag{4.27}$$

The tensors $F_{\mathbf{yp}}$, $F_{\mathbf{pp}}$, $F_{\mathbf{py}}$ are defined by similar formulas. Using the second order adjoint model trajectory $\sigma(t)$, we can compute the Hessian-vector product for any given vector $\boldsymbol{\delta}\mathbf{q} \in \mathbb{R}^{n_q}$ as:

$$\begin{aligned} \left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{q}^2}\right) \cdot \boldsymbol{\delta}\mathbf{q} &= \int_{t^0}^{t^F} \left(\mathbf{g_{qq}}\boldsymbol{\delta}\mathbf{q} + \mathbf{g_{qy}}(\mathbf{y_q}\boldsymbol{\delta}\mathbf{q}) + F_{\mathbf{q}}^T \boldsymbol{\sigma} + \left(F_{\mathbf{qq}} \odot \boldsymbol{\delta}\mathbf{q}\right)^T \boldsymbol{\lambda} + \left(F_{\mathbf{qy}} \odot (\mathbf{y_q}\boldsymbol{\delta}\mathbf{q})\right)^T \boldsymbol{\lambda}\right) \mathrm{d}t \\ &+ \left.\left(\left(\mathbf{y_{qq}} \odot \boldsymbol{\delta}\mathbf{q}\right)^T \boldsymbol{\lambda} + {\mathbf{y_q}}^T \boldsymbol{\sigma}\right)\right|_{t=t^0} \,. \end{aligned} \tag{4.28}$$

The integral term in (4.28) can be evaluated concurrently with (4.25) through a set of n_q quadrature equations:

$$\begin{aligned} \dot{\mathbf{v}} &= -F_{\mathbf{q}}^T\sigma - \left(F_{\mathbf{qq}} \odot \boldsymbol{\delta}\mathbf{q}\right)^T \boldsymbol{\lambda} - \left(F_{\mathbf{qy}} \odot (\mathbf{y_q}\boldsymbol{\delta}\mathbf{q})\right)^T \boldsymbol{\lambda} - \mathbf{g_q}\boldsymbol{\delta}\mathbf{q} - \mathbf{g_{py}}(\mathbf{y_q}\boldsymbol{\delta}\mathbf{q}) \ , \quad t^F \geq t \geq t^0 \ , \\ \mathbf{v}(t^F) &= 0 \ . \end{aligned} \tag{4.29}$$

It follows that

$$\left(\frac{\partial^2 G}{\partial q^2}\right) \cdot \boldsymbol{\delta}\mathbf{q} = \mathbf{v}(t^0) + \left(\left(\mathbf{y_{qq}} \odot \boldsymbol{\delta}\mathbf{q}\right)^T \boldsymbol{\lambda} + \mathbf{y_q}^T\sigma\right)\Big|_{t=t^0} \ . \tag{4.30}$$

We note that a TLM integration starting from $\boldsymbol{\tau}(t^0) = \mathbf{y_q}(t^0) \cdot \boldsymbol{\delta}\mathbf{q}$ is necessary, to obtain the $\mathbf{y_q}(t)\boldsymbol{\delta}\mathbf{q}$ term in (4.25) [46, 72]. One can generalize this technique to derive the tangent linear and adjoint models for point-wise functionals [30, 46]:

$$\mathcal{J} = \mathbf{g}\left(t^F, \mathbf{y}(t^F), \mathbf{q}\right) \ . \tag{4.31}$$

Similarly, a simple change of variables [72] in the forward model leads to the differential equations used for calculating first and second order sensitivities with respect to the initial conditions $\mathbf{y}^0$.

Using dense output in adjoint sensitivity analysis

For general nonlinear models (4.1), the first order adjoint variable $\boldsymbol{\lambda}(t)$, and the second order adjoint solution $\boldsymbol{\sigma}(t)$ depend on the forward and tangent linear variables $\mathbf{y}(t)$ and $\boldsymbol{\tau}(t)$. Moreover, adjoint models are final value problems integrated backward in time, and the time steps in the adjoint simulation are adapted independently from the forward integration. Thus it is necessary to approximate the forward solution $\mathbf{y}(t)$ at a set of time points that cannot be known *a priori*. To do this, we use the continuous extensions of the Runge-Kutta pairs. Dense output allows for a cost-effective and accurate interpolation. This is confirmed by the numerical experiments described in section 4.1.4. The computational overhead of dense output is equal to a small number of right-hand-side forward or tangent linear model evaluations per time step.

4.1.3 Implementation details

Runge-Kutta implementations

`DENSERKS` implements several explicit Runge-Kutta pairs, listed in Table 4.1. It is important to note that the user needs to employ the same Runge-Kutta process for both the forward and adjoint mode integrations when solving (4.20) and (4.25). This is required by the fact that the adjoint code uses the interpolation weights $\mathbf{k}_j^n$ computed by forward or tangent

linear code when it builds the interpolant $u(\theta)$ (4.6). Thus, order q accuracy in the adjoint solution is possible only if the the stages $\mathbf{k}_j^n$ corresponding to the forward Runge-Kutta scheme are stored during the forward integration. The Butcher tableaus and the dense output coefficients corresponding to the Runge-Kutta schemes in Table 4.1 can be found in [123, 124, 134, 135, 136]. The `DENSERKS` adjoint model integrators can use either cubic/quintic Hermite polynomial interpolation or dense output for forward trajectory recomputations. The interpolation options are dependent on the particular choice of Runge-Kutta pair.

Checkpointing

For nonlinear models, the adjoint equations (4.20) and (4.25) are dependent on the forward and tangent linear model solutions $\mathbf{y}(t)$ and $\boldsymbol{\tau}(t) = \mathbf{y_q}\boldsymbol{\delta}\mathbf{q}$. Moreover, since both adjoint equations are final value problems, they cannot be integrated alongside (4.1) and (4.14). The error estimators and time step controllers for the adjoint problems are independent from those used in the forward integrations. Thus, (4.20) and (4.25) will require an approximation to $\mathbf{y}(t)$ and $\boldsymbol{\tau}(t)$ at time points that can not be determined a priori. Continuous Runge-Kutta schemes can, as explained above, provide the interpolated values. However, to construct the forward solution interpolants on the fly the adjoint model run, all the interpolation weights need to be stored in memory during the forward model simulation. This can be prohibitive, especially for long time integrations.

To mitigate the memory storage problems that may arise in an adjoint model integration, `DENSERKS` employs a two-level checkpointing strategy [137, 126, 47]. The checkpointing algorithm can be briefly explained as follows. We perform one forward model simulation from t^0 to t^F. During this integration, `DENSERKS` writes the integrator state to a disk file every `Nd` time steps, and discards all other solution information. The library user can control the value of `Nd`. The forward model run yields `Nc` state snapshots (also called file checkpoints) that will be reused in the adjoint model integration. The information contained in a checkpoint must be sufficient to allow a hot restart of the forward integration from the time point at which the checkpoint was written. This means that the forward model trajectory computed after a restart is identical to the one calculated during the initial simulation, when all the checkpoints were written.

During the adjoint model run, `DENSERKS` divides the forward problem into Nc smaller problems, each solvable in at most Nd time steps. To do so, the software automatically reads the file checkpoints in reverse order. Consider any two consecutive file checkpoints corresponding to time points t^a and t^b ($t^a < t^b$). First, `DENSERKS` recomputes the forward trajectory from t^a to t^b, using the checkpoint written at t^a as the initial data. The interpolation weights is stored in the working memory. Since the forward subproblem can be solved in at most Nd steps, the memory requirements are lowered significantly. The user can set Nd according to the amount of available memory. Then, `DENSERKS` integrates the adjoint model from t^b to t^a, making use of the interpolated forward trajectory when needed.

Table 4.1: Explicit Runge-Kutta schemes implemented by `DENSERKS`: RK2 - Fehlberg-type Runge-Kutta method of order 2(3); RK3 - 3rd order Runge-Kutta with second order error control; RK4 - 4th order Runge-Kutta method with 3rd order error control built on the classic "3/8 rule"; RK5 - DOPRI5(4); RK6 - Runge-Kutta pair built on top of RK6(5)9FM; RK8 - DOPRI8(6).

Runge-Kutta method	**Order of accuracy**	**Error control order**	**Interpolation method**
RK2 (Fehlberg)	2	3	Hermite (3rd/5th order)
RK3	3	2	Hermite (3rd/5th order)
RK4 ("3/8-rule")	4	3	Hermite (3rd/5th order)
RK5	5	4	Hermite (3rd/5th order), dense output (4th order)
RK6 (DOPRI5)	6	5	Hermite (3rd/5th order), dense output (6th order)
RK8 (DOPRI8)	8	6	Hermite (3rd/5th order), dense output (7th order)

The same strategy is used for the TLM. The checkpointing mechanism lowers the memory storage requirements considerably, at the expense of additional recomputations of the forward model or TLM trajectories. This makes long-time adjoint integrations computationally feasible.

Let us denote the computational cost of integrating (4.1) from t^0 to t^F (a complete forward model integration) by $\mathcal{C}_{\text{FWD}}$. Let $\mathcal{C}_{\text{ADJ}}$ be the cost of solving (4.20) backward in time from t^F to t^0, assuming the forward model state is available at all required time points (i.e., all the necessary interpolation weights are stored in memory). Finally, let $\mathcal{C}_{\text{QAD}}$ be the computational cost of solving (4.22) for $w(t^0)$. Then we have the following bounds for $\mathcal{C}_{\text{BKWD}}$, the cost of computing $\boldsymbol{\lambda}^0 \approx \boldsymbol{\lambda}(t^0)$ given only $\mathbf{y}(t^0)$ (i.e., a backward problem integration with two-level checkpointing):

$$\mathcal{C}_{\text{FWD}} + \mathcal{C}_{\text{ADJ}} + \mathcal{C}_{\text{QAD}} \leq \mathcal{C}_{\text{BKWD}} \leq 2 \times \mathcal{C}_{\text{FWD}} + \mathcal{C}_{\text{ADJ}} + \mathcal{C}_{\text{QAD}} \,. \tag{4.32}$$

The upper bound in (4.32) follows from the observation that we need to integrate (4.1) at most two times: once when writing the file checkpoints, and a second time during the integration of (4.20), to obtain the interpolated forward solution at the time points needed by the adjoint integrator. The second inequality in (4.32) becomes an equality when the last step in the forward model integration is written as a file checkpoint.

The computational cost $\mathcal{C}_{\text{BKWD2}}$ of a second order adjoint solve ((4.1), (4.14), (4.25), and (4.29)), can be similarly bounded as

$$\mathcal{C}_{\text{L}} \leq \mathcal{C}_{\text{BKWD2}} \leq \mathcal{C}_{\text{H}} \,, \tag{4.33}$$

where

$$\begin{aligned} \mathcal{C}_{\text{L}} &= \mathcal{C}_{\text{FWD}} + \mathcal{C}_{\text{TLM}} + \mathcal{C}_{\text{ADJ}} + \mathcal{C}_{\text{SOA}} + \mathcal{C}_{\text{QSOA}} \\ \mathcal{C}_{\text{H}} &= 2 \times \mathcal{C}_{\text{FWD}} + 2 \times \mathcal{C}_{\text{TLM}} + \mathcal{C}_{\text{ADJ}} + \mathcal{C}_{\text{SOA}} + \mathcal{C}_{\text{QSOA}} \,. \end{aligned} \tag{4.34}$$

Here $\mathcal{C}_{\text{TLM}}$, $\mathcal{C}_{\text{SOA}}$, and $\mathcal{C}_{\text{QSOA}}$ denote the computational cost incurred when integrating (4.14), (4.25), and (4.29), respectively.

Code organization and usage

Table 4.2 lists the integrator subroutines in `DENSERKS`, together with short descriptions of their purpose and usage. We provide the code for the test problems described in section 4.1.4 in the `DENSERKS` package. Note that backward time integration in the forward mode is not supported, nor is forward time integration in the adjoint mode. Thus, all integrators in Table 4.2 require that $t^0 \leq t^F$.

We now describe a step by step procedure describing how to enhance an existing embedded Runge-Kutta code with sensitivity analysis capabilities. As before, we let q be the order of accuracy of the Runge-Kutta pair.

Table 4.2: `DENSERKS` integrator subroutines.

Integrator	**Description**
`RKINT`	Forward model integrator. Writes forward model trajectory data to the memory buffers and file checkpoints, if the memory buffering and file checkpointing mechanisms are enabled. This routine is user-callable.
`RKINT_TLM`	Tangent linear model integrator. Simultaneously integrates the forward model (the resulting system has size $2 \times n_y$). Writes forward and tangent linear model trajectory data to the memory buffers and file checkpoints, if the memory buffering and file checkpointing mechanisms are enabled. This routine is user-callable.
`RKINT_ADJ`	First order adjoint model integrator. The routine is used to integrate the first order adjoint model and the associated quadrature equations (for a total backward problem size of $n_y + n_q$) between two given consecutive checkpoints. `RKINT_ADJ` reads forward model trajectory data from the memory buffers, implicitly assuming they hold valid forward integration information. The user should *not* call this subroutine directly. Instead, all first order adjoint integrations should be done via calls to the `RKINT_ADJDR` wrapper subroutine.
`RKINT_SOA`	Second order adjoint model integrator. Simultaneously integrates the first and second order adjoint model equations and the quadrature equations (the resulting system has size $2 \times n_y + n_q$). Reads data from the memory buffers only. The user should *not* call this subroutine directly. Instead, all first order adjoint integrations should be done via calls to the `RKINT_SOADR` wrapper subroutine.
`RKINT_ADJDR`	Wrapper for `RKINT_ADJ`. Handles all checkpoint reads and calls `RKINT` to calculate the forward model trajectory, should such recomputations be required. Calls `RKINT_ADJ` for first order adjoint model integration between consecutive checkpoints. This routine is user-callable.
`RKINT_SOADR`	Wrapper for `RKINT_SOA`. Handles all checkpoint reads and calls `RKINT_TLM` to calculate the tangent linear model trajectory, should such recomputations be required. Calls `RKINT_SOA` for second order adjoint model integration between consecutive checkpoints. This routine is user-callable.

1. Fit the Runge-Kutta pair with a dense output interpolant of order p^*. This may require an increase in the number of stages over the original scheme. For smooth $\mathbf{y}$ and $\boldsymbol{\tau}$, the interpolated forward and TLM trajectories will be q-th order accurate if $p \leq p^* + 1$, as shown in section 4.1.2. If $p > p^* + 1$, one should expect a loss of accuracy in the adjoint model solution, due to less accurate forward interpolation data.

2. Implement a checkpointing mechanism for the Runge-Kutta pair to reduce the memory requirements of an adjoint integration.

3. Write a Runge-Kutta integrator for the first and second order adjoint model based on the original Runge-Kutta pair. This solver should be able to use the checkpoint information to interpolate the forward or tangent linear model trajectories when needed.

4. Build the right hand sides of the forward and adjoint sensitivity equations (4.1), (4.14), (4.20), (4.25). These subroutines can be either hand coded, or generated using automatic differentiation (see the appendix for more details). When computing derivatives of nontrivial cost functions, we also need to code the corresponding right hand sides of the quadrature equations (4.22) and (4.29).

5. Set initial or final values for all the state variables: $\mathbf{y}(t^0)$, $\boldsymbol{\tau}(t^0)$, $\boldsymbol{\zeta}(t^0)$, $\boldsymbol{\lambda}(t^F)$, $\mathbf{w}(t^F)$, $\boldsymbol{\sigma}(t^F)$, $\mathbf{v}(t^F)$.

6. Solve the equations (4.1) - (4.29) for the desired sensitivity information using the appropriate Runge-Kutta solvers.

This procedure can be extended to, e.g, Rosenbrock, implicit Runge-Kutta, and extrapolation methods, since they all support continuous extensions [131, 132, 133]. Only the actual construction of the dense output interpolant is different; steps 2 to 6 are identical to the ones described here.

To use the `DENSERKS` subroutines in Table 4.2, skip directly to step 4 in the procedure listed above. A first order adjoint problem can be solved by the following steps:

```
!! define the size of the forward and adjoint model state vectors
   integer ny = ...
!! define the size of the quadrature system state (number of parameters)
   integer nq = ...
!! set the initial and final integration time
   double precision t0 = ..., tF = ...
!! define the forward and adjoint model state vectors
   double precision y(ny), ady(nx)
!! w is the quadrature system state vector
   double precision w(nq)
!! write a file checkpoint every Nd time steps
```

```
   integer Nd = ...
!! allocate the memory buffers and initialize the checkpoint files, if needed
   call rk_AllocateTapes(...)
!! initialize y, and other integrator parameters as needed
   y(t0) = ...
!! integrate the forward model
!! FWD_RHS = subroutine computing the right hand side of (4.1) at given (t,y(t))
!! if we disable file checkpointing, the values of Nd and Nc are ignored by the algorithm
   call RKINT(nx,y,RHS,t0,tF,Nd,Nc,...)
!! upon return from RKINT, Nc (integer) = the total number of file checkpoints written to disk
!! set final values for ady and w
   ady(tF) = ...
   w(tF) = ...
!! integrate the first order adjoint model
!! ADJ_RHS = subroutine computing the right hand side of (4.20) at given (t,y(t),ady(t))
!! QUAD_RHS = subroutine computing the right hand side of (4.22) at given (t,y(t),ady(t))
   call RKINT_ADJDR(nx,ady,ADJ_RHS,...,nq,w,QUAD_RHS,...,RHS,...,Nc,...)
!! upon exit from RKINT_ADJDR:
!!    ady ≈ λ(t^0) in (4.20)
!!    w ≈ w(t^0) in (4.22)
!! close the checkpoint files and deallocate the memory buffers
   call rk_DeallocateTapes
```

Second order adjoint models are handled similarly. In the case of multiple integrations of the same first or second order adjoint system (with different final conditions), it is not recommended to call `RKINT_ADJDR` or `RKINT_SOADR` multiple times. Instead, `RKINT_ADJDR_M` and `RKINT_SOADR_M` allow for significant performance gains, due to the reuse of forward and tangent linear model trajectory data over all adjoint model integrations. This lowers the cost of M first order adjoint model integrations, $\mathcal{C}_{\text{M\_BKWD}}$, from $M \times \mathcal{C}_{\text{BKWD}}$ in (4.32) to

$$\mathcal{C}_{\text{FWD}} + M \times \mathcal{C}_{\text{ADJ}} \leq \mathcal{C}_{\text{M\_BKWD}} \leq 2 \times \mathcal{C}_{\text{FWD}} + M \times \mathcal{C}_{\text{ADJ}} \,. \tag{4.35}$$

A similar reduction in computational cost is noticed in the case of M second order adjoint model integrations with `RKINT_SOADR_M`. The corresponding cost bounds $\mathcal{C}_{\text{L}}$ and $\mathcal{C}_{\text{H}}$ in (4.34) become

$$\begin{aligned} \mathcal{C}_{\text{LM}} &= \mathcal{C}_{\text{FWD}} + \mathcal{C}_{\text{TLM}} + M \times \mathcal{C}_{\text{ADJ}} + M \times \mathcal{C}_{\text{SOA}} \\ \mathcal{C}_{\text{HM}} &= 2 \times \mathcal{C}_{\text{FWD}} + 2 \times \mathcal{C}_{\text{TLM}} + M \times \mathcal{C}_{\text{ADJ}} + M \times \mathcal{C}_{\text{SOA}} \,. \end{aligned} \tag{4.36}$$

Thus

$$\mathcal{C}_{\text{LM}} \leq \mathcal{C}_{\text{M\_BKWD2}} \leq \mathcal{C}_{\text{HM}} \,. \tag{4.37}$$

Comparison with other related software

Table 4.3 provides a quick overview and comparison between the main features of `DENSERKS`, `CVODES`, and `DASPKADJOINT`.

Code availability

The `DENSERKS` source code is released under a GNU open source license and is available for download at:
`http://people.cs.vt.edu/~asandu/Software/DENSERKS`.

4.1.4 Numerical experiments

All numerical experiments were performed with double precision Fortran on a 3 GHz Intel Pentium 4 workstation running Fedora Core 6 Linux.

The Arenstorf orbit

The Arenstorf orbit IVP is usually given as a set of two second order ODEs, which can readily be transformed into a first order initial-value problem [124]:

$$\begin{aligned}
\dot{\mathbf{y}}_1 &= \mathbf{y}_3 \\
\dot{\mathbf{y}}_2 &= \mathbf{y}_4 \\
\dot{\mathbf{y}}_3 &= \mathbf{y}_1 + 2\mathbf{y}_4 - \widehat{\mu}\frac{\mathbf{y}_1+\mu}{\left[(\mathbf{y}_1+\mu)^2+\mathbf{y}_2^2\right]^{3/2}} - \mu\frac{\mathbf{y}_1-\widehat{\mu}}{\left[(\mathbf{y}_1-\widehat{\mu})^2+\mathbf{y}_2^2\right]^{3/2}} \\
\dot{\mathbf{y}}_4 &= \mathbf{y}_2 - 2\mathbf{y}_3 - \widehat{\mu}\frac{\mathbf{y}_2}{\left[(\mathbf{y}_1+\mu)^2+\mathbf{y}_2^2\right]^{3/2}} - \mu\frac{\mathbf{y}_2}{\left[(\mathbf{y}_1-\widehat{\mu})^2+\mathbf{y}_2^2\right]^{3/2}} \ ,
\end{aligned} \tag{4.38}$$

where $\mu = 1 - \widehat{\mu} = 0.012277471$. The initial conditions are:

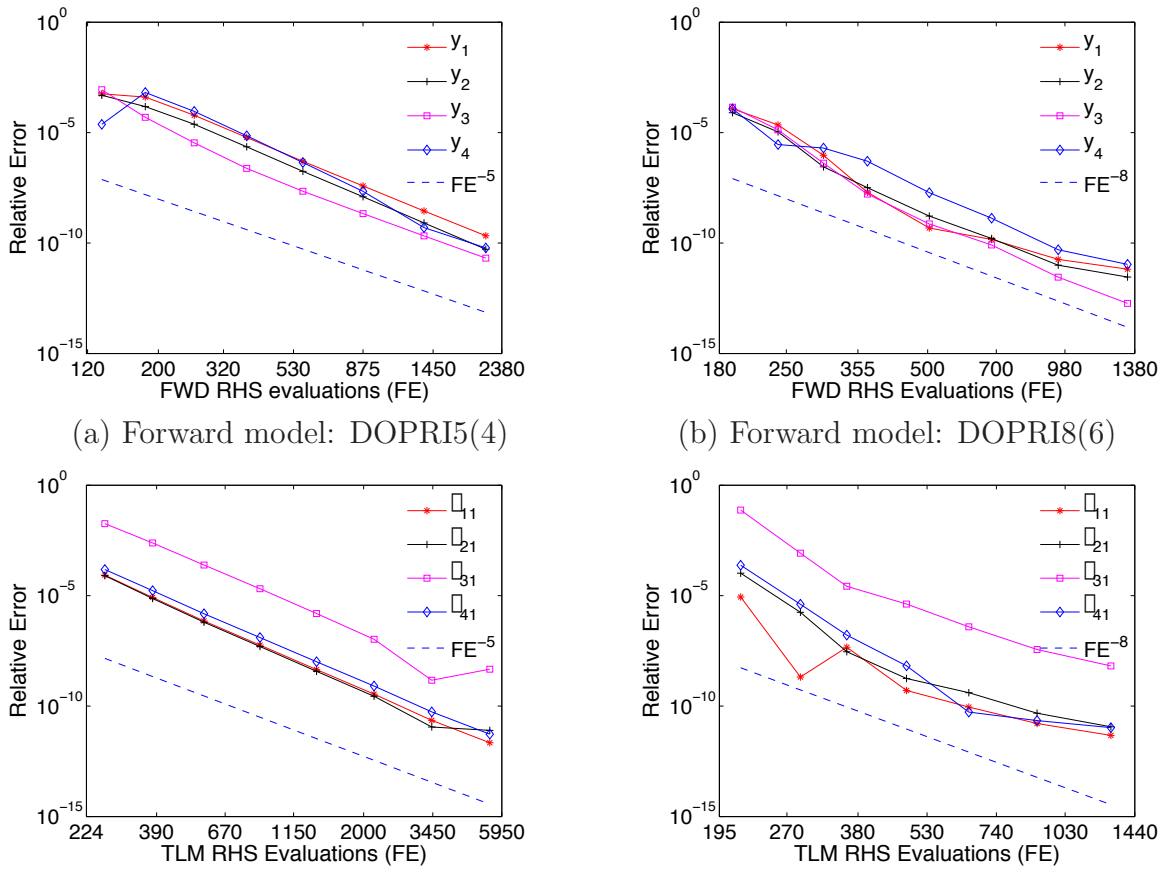
$$\begin{aligned}
\mathbf{y}_1(t^0) &= 0.994 \\
\mathbf{y}_2(t^0) &= 0 \\
\mathbf{y}_3(t^0) &= 0 \\
\mathbf{y}_4(t^0) &\approx -2.0016 \ .
\end{aligned} \tag{4.39}$$

We compute the sensitivities of the model trajectory with respect to variations in the initial value of the first solution component $y_1(t^0)$. Thus, the cost functionals chosen for this example are:

$$\mathcal{J}_i(\mathbf{y}_1(t^0)) = \mathbf{y}_i(t^F) \ \ , \ \ i = 1 \ldots 4 \ . \tag{4.40}$$

Table 4.3: High-level comparison of `DENSERKS` against `CVODES` v2.5.0 [2] and `DASPKADJOINT` [3]. `DASPKADJOINT` implements sensitivity analysis for DAEs (not available in `DENSERKS` or `SUNDIALS`). Note that `DENSERKS` has built in support for second order sensitivity analysis, a feature not directly present in the other two sensitivity solvers.

	DENSERKS	**CVODES**	**DASPKADJOINT**
Numerical methods implemented	Explicit Runge-Kutta	Adams-Moulton and BDF	BDF
Orders of accuracy	2 – 8	1 – 12 (Adams-Moulton) 1 – 5 (BDF) (variable order)	1 – 5
Interpolation method	Dense output (built-in)	Hermite, variable order polynomial method	Hermite
Interpolation order	3 – 8	3 / 5	3
Built-in sensitivity analysis	1st order forward / adjoint 2nd order (forward-over-adjoint)	1st order forward / adjoint	1st order forward / adjoint



(a) Forward model: DOPRI5(4)

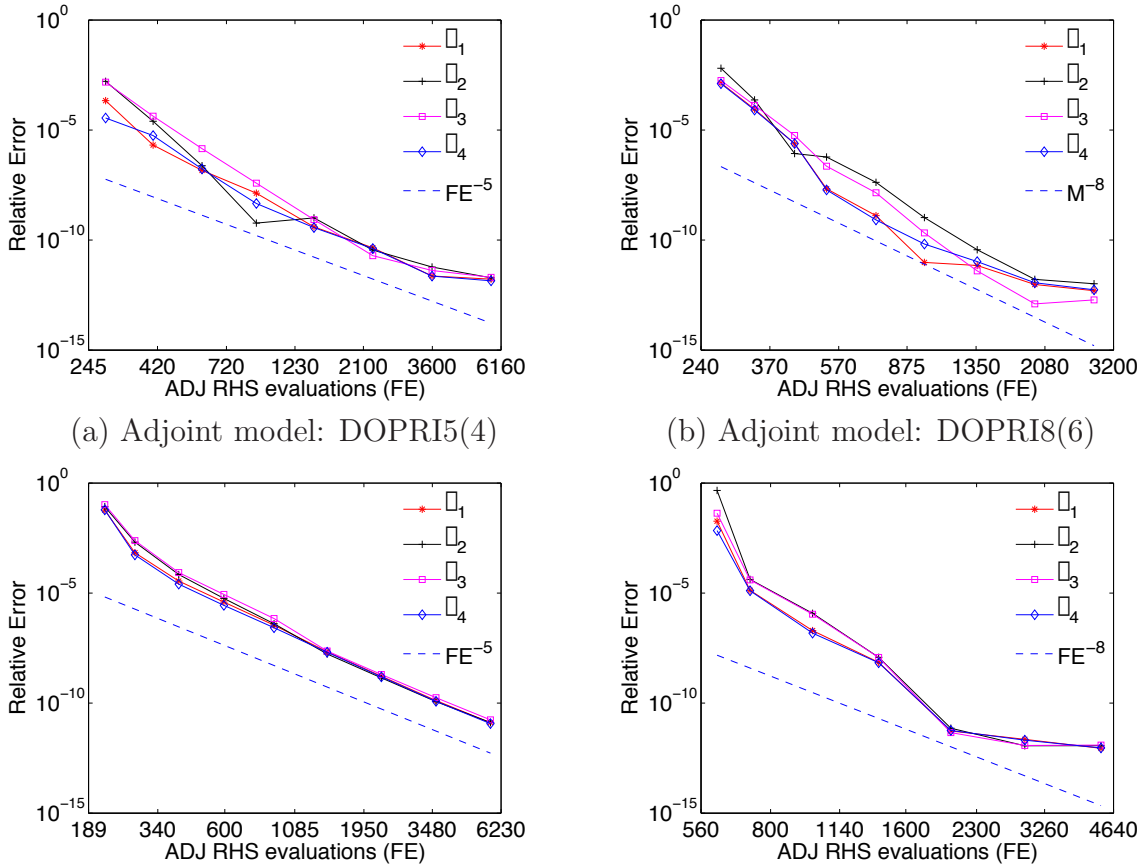
(b) Forward model: DOPRI8(6)

(c) TLM: DOPRI5(4)

(d) TLM: DOPRI8(6)

Figure 4.1: The Arenstorf orbit: Relative errors in $\mathbf{y}(t)$ and $\boldsymbol{\tau}_{i1} = (\partial \mathbf{y}_i / \partial \mathbf{y}_1^0)\,(t^F)$. $t^0 = 0$, $t^F \approx 1.707$ (1/10 of an orbit period).

We consider one-tenth of a full orbit period as our forward model integration window: $t^0 = 0$ and $t^F \approx 1.7065$. Figures 4.1 and 4.2 illustrate the absolute errors of the forward, tangent linear and first order adjoint model solvers using the DOPRI5(4) and DOPRI8(6) Runge-Kutta numerical schemes with 4th order ($p^* = 4$) and 7th order ($p^* = 4$) dense output. Since the necessary condition $p \leq p^* + 1$ is satisfied for both Runge-Kutta pairs, the corresponding tangent linear and adjoint model solutions are 5th and 8th order accurate, respectively. This shows that the dense output interpolants make it possible to compute a high-quality adjoint solution that has the same order of accuracy as the underlying Runge-Kutta method. The reference solution was obtained by a TLM integration with very tight absolute and relative tolerances ($\texttt{ATOL} = \texttt{RTOL} = 10^{-14}$), using the DOPRI8(53) code in [124].



(a) Adjoint model: DOPRI5(4)

(b) Adjoint model: DOPRI8(6)

(c) Adjoint model: DOPRI5(4) with variable forward tolerances

(d) Adjoint model: DOPRI8(6) with variable forward tolerances

Figure 4.2: The Arenstorf orbit: Work-precision diagrams for the DOPRI5(4) pair ((a),(c)) with 4th order dense output, and the DOPRI8(6) scheme ((b),(d)) with 7th order dense output. $t^0 = 0$, $t^F \approx 1.707$. We ran two series of tests. In the first, we integrated the adjoint model with variable tolerances, while keeping the forward integration tolerances constant ((a) and (c) illustrate this case, with $\text{ATOL}^{\text{fwd}} = \text{RTOL}^{\text{fwd}} = 10^{-12}$). Second, we varied both the forward and adjoint model tolerances during several adjoint model test runs ((b) and (d) show the results of several adjoint model integrations where the prescribed absolute and relative forward and adjoint integrator tolerances are equal and vary between 10^{-4} and 10^{-12}).

The Van der Pol oscillator

First order sensitivity analysis We now consider the Van der Pol oscillator [131]. The first-order ODE system reads:

$$\begin{aligned} \dot{\mathbf{y}}_1 &= \mathbf{y}_2 \\ \dot{\mathbf{y}}_2 &= \frac{1}{\epsilon}\left(\mathbf{y}_2 - \mathbf{y}_1 - \mathbf{y}_1^2\mathbf{y}_2\right) , \end{aligned} \tag{4.41}$$

with $\epsilon = 10^{-2}$ and initial conditions

$$\begin{aligned} \mathbf{y}_1(t^0) &= 2 \\ \mathbf{y}_2(t^0) &= 0 . \end{aligned} \tag{4.42}$$

Consider the cost function

$$\mathcal{J}(\mathbf{y}(t^0)) = \mathbf{y}_1(t^F) . \tag{4.43}$$

The corresponding adjoint model can be computed using the Lagrangian method described in (4.17) – (4.21), where $\mathcal{J}$ can informally be written as

$$\mathcal{J}(\mathbf{y}(t^0)) = \int_{t^0}^{t^F} \delta(t - t^F)\, \mathbf{y}_1(t)\, dt . \tag{4.44}$$

where $\delta(\cdot)$ is the Dirichlet delta function:

$$\delta(z) = \begin{cases} 1 & , \quad z = 0 \\ 0 & , \quad z \neq 0 \end{cases} . \tag{4.45}$$

Hence the adjoint final value problem for (4.41) – (4.43) has the following form:

$$\begin{aligned} \dot{\boldsymbol{\lambda}}_1 &= \frac{1}{\epsilon}(2\mathbf{y}_1\,\mathbf{y}_2 + 1)\boldsymbol{\lambda}_2 \\ \dot{\boldsymbol{\lambda}}_2 &= -\lambda_1 + \frac{1}{\epsilon}(\mathbf{y}_1^2 - 1)\boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_1(t^F) &= 1 \\ \boldsymbol{\lambda}_2(t^F) &= 0 . \end{aligned} \tag{4.46}$$

The time integration interval spans from $t^0 = 0$ to $t^F = 2$. Note from figure 4.3(c–d) that both forward and adjoint integrators use adaptive time stepping, and step size control is performed independently during the forward and the reverse integrations.

Figure 4.3(e–f) shows how the forward integration errors affect the accuracy of the adjoint solution. It can be seen that for this particular problem, we obtain an 8th order decrease in

the error of the adjoint solution even when interpolating data from a less accurate forward integration. This error behavior reflects the order of the Runge-Kutta method. Thus, using the adaptive forward model integrator with looser tolerances still results in a fully accurate adjoint model solution. As a general rule of thumb, we recommend that the forward tolerances be set equal to or slightly lower than those used in the adjoint simulation, to avoid losing accuracy in the adjoint solution.

Figure 4.4 compares the performance of `DENSERKS` and `CVODES` for the Van der Pol problem. The results indicate that the high-order Runge-Kutta pairs implemented in `DENSERKS` outperform the multistep methods implemented in `SUNDIALS` when solving the forward and tangent linear models, requiring considerably less right-hand-side evaluations to reach a prescribed solution accuracy. When integrating the adjoint of (4.41), `CVODES` fares better than DOPRI5(4), but is still surpassed in performance by the DOPRI8(6) implementation. These results indicate that the Runge-Kutta methods and dense output-based interpolants implemented in `DENSERKS` can be competitive with other high quality implementations, yielding very accurate solutions with relatively low computational efforts.

Second order sensitivity analysis To illustrate the second order sensitivity analysis capabilities of `DENSERKS`, we look at a variant of the Van der Pol system with a larger number of parameters and more nonlinear dependencies between the variables (adapted from [69]):

$$\begin{aligned} \dot{\mathbf{y}}_1 &= \left(1-\mathbf{y}_2^2\right)\mathbf{y}_1 - \mathbf{y}_2 + \mathbf{v}(t,p) \\ \dot{\mathbf{y}}_2 &= \mathbf{y}_1 \\ \dot{\mathbf{y}}_3 &= \mathbf{y}_1^2 + \mathbf{y}_2^2 + \mathbf{v}^2(t,p)\,, \end{aligned} \tag{4.47}$$

with $t^0 = 0$, $t^F = 5$,

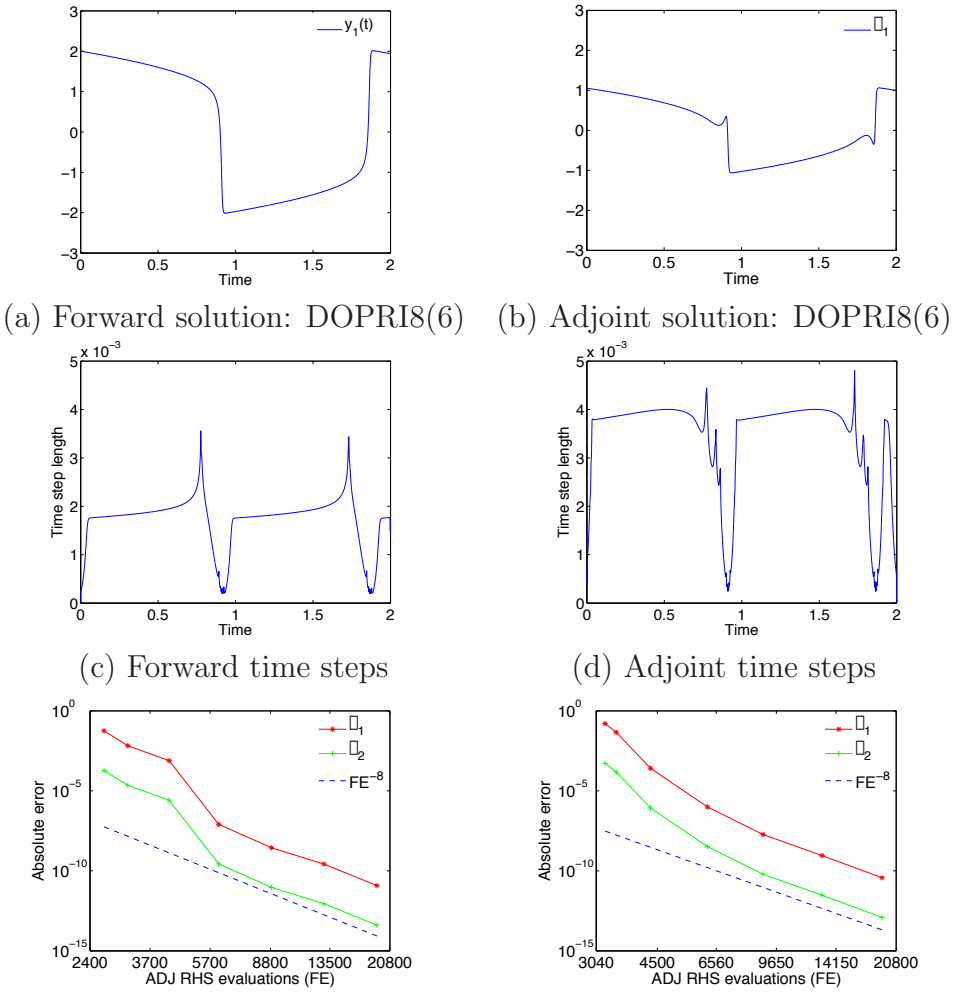
$$\mathbf{v}(t,p) = \sum_{i=1}^{n_q-1} t\,\mathbf{q}_i\,\mathbf{q}_{i+1}\,,$$

and initial conditions $\mathbf{y}_1(t^0) = 0$, $\mathbf{y}_2(t^0) = 1$, $\mathbf{y}_3(t^0) = 0$. The objective functional is chosen as

$$\mathcal{J}(p) = \mathbf{y}_3(t^F, p)\,.$$

Here

$$\begin{aligned} \mathbf{q} &= \left(\frac{1}{n_q}, \frac{1}{n_q}, \ldots, \frac{1}{n_q}\right)\,, \\ \boldsymbol{\delta}\mathbf{q} &= \left(1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n_q}\right)\,. \end{aligned}$$



(a) Forward solution: DOPRI8(6)

(b) Adjoint solution: DOPRI8(6)

(c) Forward time steps

(d) Adjoint time steps

(e) Relative errors: DOPRI8(6) with fixed forward tolerances

(f) Relative errors: DOPRI8(6) with variable forward tolerances

Figure 4.3: The Van der Pol oscillator (4.41): $t^0 = 0$, $t^F = 2$, $\epsilon = 10^{-2}$. ATOL $= 10^{-12}$ and RTOL $= 10^{-10}$. The adaptive time steps taken during the two model integrations are plotted in (c) and (d). The bottom plots illustrate the absolute errors of the adjoint solutions $\boldsymbol{\lambda}_i$ obtained using a fixed tolerance for the forward model run (ATOL$^{\text{fwd}}$ = RTOL$^{\text{fwd}}$ $= 10^{-12}$ in (e)) and, alternatively, variable tolerance values for both the forward and adjoint integrators (in (f)). The order of accuracy of the adjoint numerical solution is the same as that of the Runge-Kutta pair.

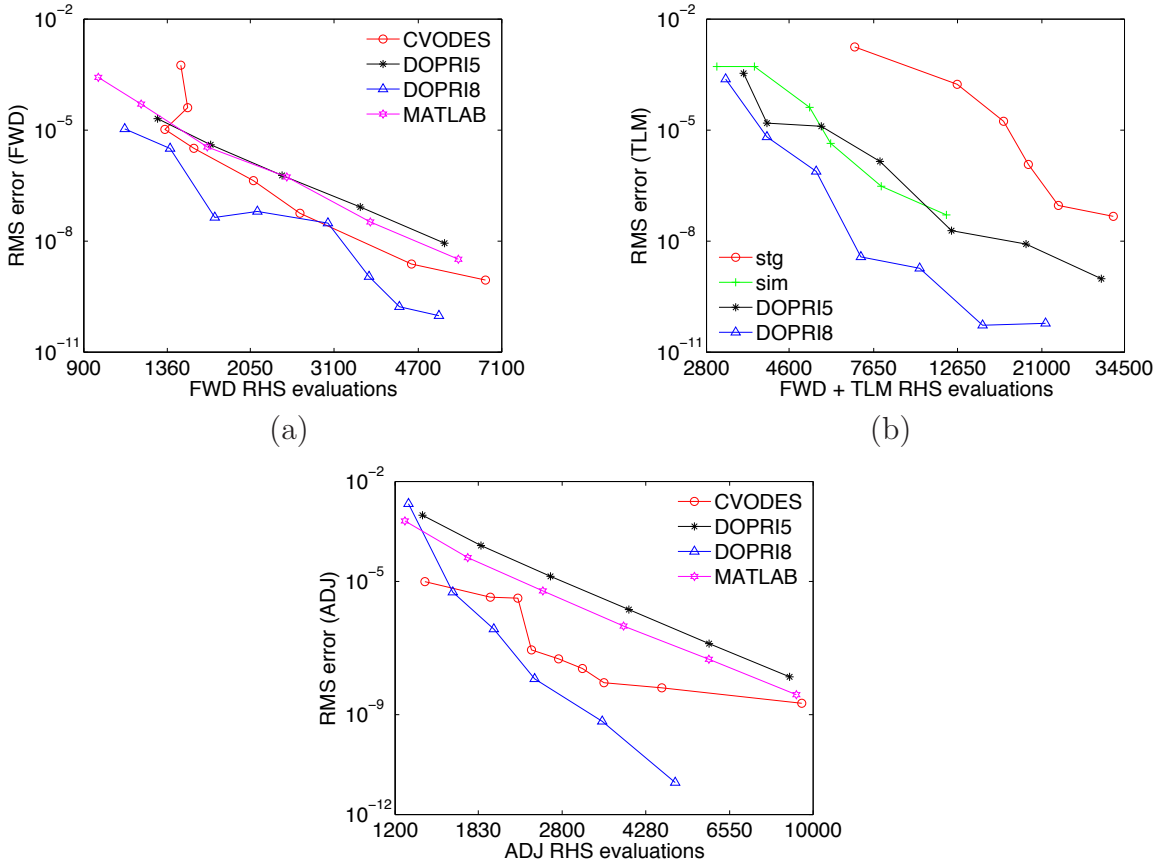


Figure 4.4: `DENSERKS` and `CVODES` work-precision diagrams for the Van der Pol system (4.41): total number of right-hand side function evaluations versus solution accuracies, for the (a) forward model, (b) TLM model, and (c) first order adjoint model. We tested `CVODES` with the staggered direct (`stg`) and simultaneous corrector methods (`sim`) (the staggered corrector method yielded results that are very similar to those obtained from its direct version). Also, we enabled full error control for the forward sensitivities. The time step controller implemented in `DENSERKS` includes tangent linear variables in the error estimator state. `DENSERKS`'s DOPRI8(6) implementation achieves best all-around work - accuracy ratios.

Table 4.4: Second order adjoint of the Van der Pol system (4.47): Runtime of the second order adjoint method (BKWD2) versus finite differences (FDIFF). DOPRI8(6) has been used for all model integrations.

n_q	FWD [ms]	BKWD2 [ms]	BKWD2/FDIFF
100	1	67	1.18
200	2	79	0.99
400	3	96	0.95
800	7	137	0.87
1600	11	245	0.82
3200	20	418	0.81
6400	41	794	0.77
12800	79	1659	0.78

The forward sensitivity system, the first and second order adjoint models and the quadrature equations are given in [69] for a general function $\mathbf{v}(t, p)$. We approximate the Hessian-vector product

$$\left(\frac{\partial^2 \mathcal{J}}{\partial \mathbf{q}^2}\right) \cdot \boldsymbol{\delta}\mathbf{q}\Big|_{t^F, \mathbf{q}} \tag{4.48}$$

using two distinct approaches. We first compute a finite difference approximation to (4.48):

$$\frac{\partial^2 \mathcal{J}}{\partial \mathbf{q}^2} \cdot \boldsymbol{\delta}\mathbf{q}\Big|_{t^F, \mathbf{q}} \approx \frac{\nabla_{\mathbf{q}}\, \mathbf{g}\left(t^F, \mathbf{q} + \epsilon\, \boldsymbol{\delta}\mathbf{q}\right) - \nabla_{\mathbf{q}}\, \mathbf{g}\left(t^F, \mathbf{q}\right)}{\epsilon}. \tag{4.49}$$

Next, we compare the finite difference approach with second order adjoint method (4.30). The ratios of the runtimes of these two schemes are given in table 4.4, for increasing values of n_q. We note that the advantages of the second order adjoint method are twofold. First, its computational cost (measured as runtime to convergence) is lower than that of the finite difference method in almost all of the test runs (as shown in the last column of table 4.4). Second, the accuracy of the Hessian-vector product approximation can easily be controlled via the user-chosen integration tolerances, whereas in the finite difference method (4.49) the accuracy depends on the particular choice of ϵ (in the best case, it is approximately equal to the square root of the chosen error tolerance). Optimizers which exploit second order information can thus use second order adjoint gradients to speed up convergence to (locally) optimal solutions.

The shallow-water equations

Our next numerical experiment concerns the two dimensional Saint-Venant approximation to fluid flow inside a shallow basin:

$$\begin{aligned}
\frac{\partial}{\partial t}\mathbf{h} + \frac{\partial}{\partial x}(\mathbf{uh}) + \frac{\partial}{\partial y}(\mathbf{vh}) &= 0 \\
\frac{\partial}{\partial t}(\mathbf{uh}) + \frac{\partial}{\partial x}(\mathbf{u}^2\mathbf{h} + \frac{1}{2}g\mathbf{h}^2) + \frac{\partial}{\partial y}(\mathbf{uvh}) &= 0 \\
\frac{\partial}{\partial t}(\mathbf{vh}) + \frac{\partial}{\partial x}(\mathbf{uvh}) + \frac{\partial}{\partial y}(\mathbf{v}^2\mathbf{h} + \frac{1}{2}g\mathbf{h}^2) &= 0\,, \quad (x,y) \in \Omega = [-3,\,3]^2\,, \quad 0 \le t \le t^F = 10
\end{aligned} \tag{4.50}$$

Here $\mathbf{h}(t,x,y)$ denotes the fluid layer thickness, and $\mathbf{u}(t,x,y)$, $\mathbf{v}(t,x,y)$ are the components of the velocity field. g is the standard value of the gravitational acceleration. The cost function quantifies the mismatch between a reference thickness $\mathbf{h}^{\text{ref}}$ and some perturbed solution $\mathbf{h}$ at the final simulation time t^F:

$$\mathcal{J} = \frac{1}{2}\sum_i \sum_j \left(\mathbf{h}(t^F, x_i, y_j) - \mathbf{h}^{\text{ref}}(t^F, x_i, y_j)\right)^2\,, \tag{4.51}$$

for all grid points $(x_i,\, y_j)$ inside a verification area $\Omega_v \subset \Omega$. The adjoint final condition reads:

$$\boldsymbol{\lambda}(t^F) = \begin{cases} \mathbf{h}(t^F, x_i, y_j) - \mathbf{h}^{\text{ref}}(t^F, x_i, y_j) & ,\quad (x_i,\, y_j) \in \Omega_v \\ 0 & ,\quad (x_i,\, y_j) \notin \Omega_v\,. \end{cases}$$

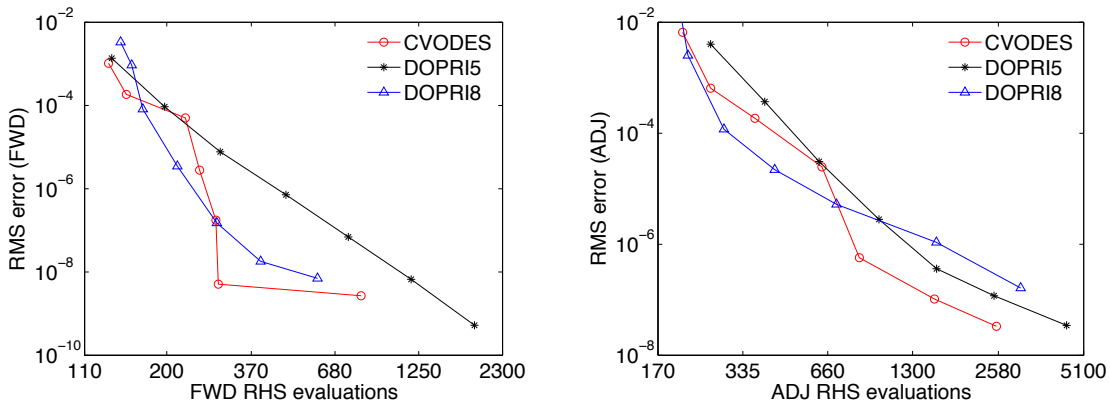
The shallow water PDEs (4.50) are converted to a semi-discrete form using third order upwind finite differences. The boundary conditions are periodic. File checkpoints are written every 50 time steps. The work-accuracy plots for forward and adjoint simulations with both `SUNDIALS` and `DENSERKS` are shown in figure 4.5. Again, the high-order continuous Runge-Kutta pairs yield performance comparable to that of the Adams-Moulton implementation found in `SUNDIALS`.

Numerical optimization for a convection-diffusion problem We now use `DENSERKS` to solve a parameter estimation problem. We consider the following IVP for $\mathbf{y}(t,x)$ (adapted from [138]):

$$\begin{aligned}
&\frac{\partial \mathbf{y}}{\partial t} = \mathbf{q}_1 \frac{\partial^2 \mathbf{y}}{\partial x^2} + \mathbf{q}_2 \frac{\partial \mathbf{y}}{\partial x} \\
&t^0 = 0 \le t \le t^F = 1 \\
&x_0 = 0 \le x \le x_1 = 2\,,
\end{aligned} \tag{4.52}$$

with initial and boundary conditions

$$\begin{aligned}
&\mathbf{y}(t, x_0) = \mathbf{y}(t, x_1) = 0\,, \;\; \forall\, t \in [t^0, t^F]\,, \\
&\mathbf{y}(t^0, x) = \mathbf{y}^0(x) = x(2-x)\, e^{2x}\,, \;\forall\, x \in [x_0,\, x_1]\;.
\end{aligned} \tag{4.53}$$



(a) (b)

Figure 4.5: Performance comparison of the high-order Runge-Kutta pairs in `DENSERKS` and the Adams-Moulton implementations in `SUNDIALS`, for the forward and adjoint two-dimensional shallow water problem: (a) Root mean square (RMS) errors in the fluid layer thickness $\mathbf{h}(t^F)$ and (b) RMS errors in the gradient $\nabla_{\mathbf{h}^0}\mathcal{J}$, for the cost function (4.51).

Let the solution of (4.52) with $\mathbf{q}_1 = 1$ and $\mathbf{q}_2 = 0.5$ be denoted by $\mathbf{y}^{\text{ref}}(t, x)$. The objective function reads:

$$\mathcal{J}(t^F, \mathbf{q}) = \frac{1}{2}\int_{x_0}^{x_1} \left(\mathbf{y}(t, x) - \mathbf{y}^{\text{ref}}\right)^2 \mathrm{d}x\,. \tag{4.54}$$

The objective is to find the parameters $\mathbf{q}_*$ that minimize the cost function $\mathcal{J}$ subject to the constraints (4.52). To solve this minimization problem, the `L-BFGS-B` optimization routine described in [139] is used. The exact solution is $\mathbf{q}_* = \mathbf{q}^{\text{ref}}$, with $\mathcal{J}\left(t^F, \mathbf{q}_*\right) = 0$. The gradient (4.54) can be obtained as:

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \mathbf{q}_1}\left(t^F, \mathbf{q}\right) &= \int_{t^0}^{t^F}\int_{x_0}^{x_1} \boldsymbol{\lambda}\,\frac{\partial^2 \mathbf{y}}{\partial x^2}\,\mathrm{d}x\,\mathrm{d}t \\
\frac{\partial \mathcal{J}}{\partial \mathbf{q}_2}\left(t^F, \mathbf{q}\right) &= \int_{t^0}^{t^F}\int_{x_0}^{x_1} \boldsymbol{\lambda}\,\frac{\partial \mathbf{y}}{\partial x}\,\mathrm{d}x\,\mathrm{d}t\,,
\end{aligned} \tag{4.55}$$

where $\boldsymbol{\lambda}(t, x)$ is the solution of the adjoint PDE:

$$\begin{aligned}
\frac{\partial \boldsymbol{\lambda}}{\partial t} &= -\mathbf{q}_1\frac{\partial^2 \boldsymbol{\lambda}}{\partial x^2} + \mathbf{q}_2\frac{\partial \boldsymbol{\lambda}}{\partial x} \\
\boldsymbol{\lambda}(t^F, x) &= \mathbf{y}(t^F, x) - \mathbf{y}^{\text{ref}}(t^F, x) \\
\boldsymbol{\lambda}(t, x_0) &= \boldsymbol{\lambda}(t, x_1) = 0\,.
\end{aligned} \tag{4.56}$$

We discretize the spatial derivatives in (4.52) and (4.56) using centered finite difference formulas on a uniform grid. Eliminating the homogeneous Dirichlet boundary conditions,

it follows that (4.52) and (4.56) are each described by a system of n_y ODEs. The gradient (4.55) is calculated via two additional quadrature equations:

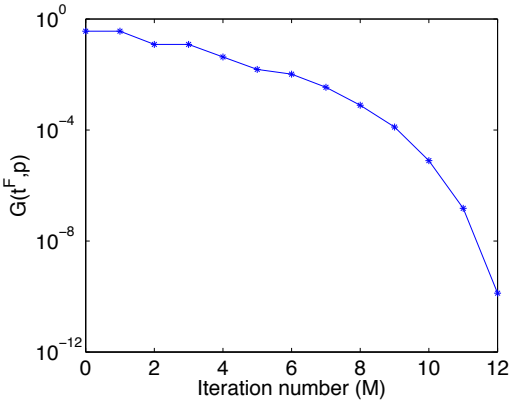
$$\begin{aligned} \dot{\mathbf{w}}_1 &= -\int_{x_0}^{x_1} \lambda \frac{\partial^2 \mathbf{y}}{\partial x^2}\, \mathrm{d}x \\ \dot{\mathbf{w}}_2 &= -\int_{x_0}^{x_1} \lambda \frac{\partial \mathbf{y}}{\partial x}\, \mathrm{d}x \\ \mathbf{w}_1(t^F) &= \mathbf{w}_2(t^F) = 0 \ , \quad t^F \geq t \geq t^0 \ . \end{aligned} \tag{4.57}$$

The spatial integrals in (4.57) are approximated using the trapezoidal rule, with the grid points chosen as quadrature nodes. Figure 4.6 illustrates the decrease in the cost function $\mathcal{J}(\mathbf{q})$ and its projected gradient norm during several `L-BFGS-B` iterations. The starting point is $(\mathbf{q}_1^0, \mathbf{q}_2^0) = (3, 3)$ and $n_y = 70$. After about 10 iterations, the parameters have converged to the reference values $(\mathbf{q}_1^{\mathrm{ref}}, \mathbf{q}_2^{\mathrm{ref}}) = (1, 0.5)$.

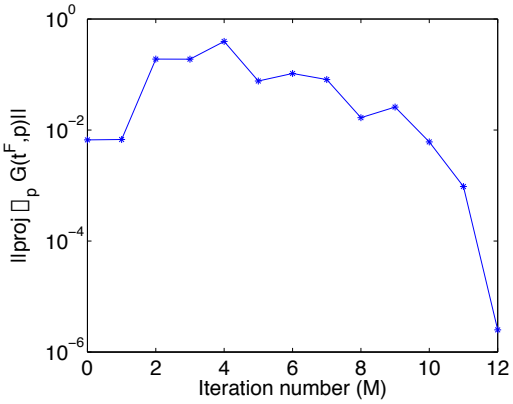
4.1.5 Summary

In this section we investigated the solution of forward and adjoint sensitivity ODEs using continuous Runge-Kutta schemes. Previous work in this area has focused on linear multistep methods. Adjoints of nonlinear models are dependent on the forward model solution. Since adjoint models need to be integrated backward in time, they require the original model solution at time points that cannot generally be known a priori. Hence some form of interpolation is needed. We propose using the dense output mechanism built in the continuous Runge-Kutta pairs as a forward solution interpolant. Since high-order Runge-Kutta pairs such as DOPRI5(4) or DOPRI8(6) support continuous extensions, dense output is shown to lead to the computation of very accurate adjoint model trajectories with only a small increase in the computational cost over the original Runge-Kutta pair. The overhead of dense output is approximately equal to the cost of the few extra stages introduced in the Runge-Kutta method to accommodate the interpolant.

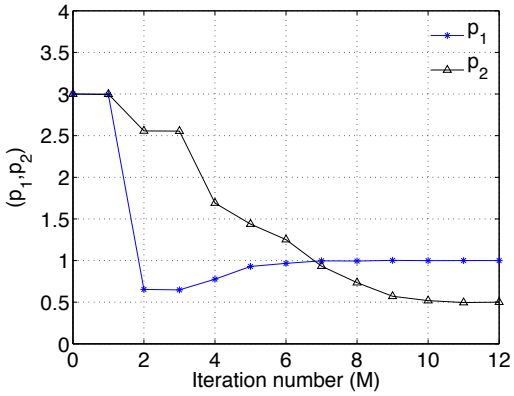
Our `DENSERKS` library implements several well-known continuous Runge-Kutta pairs. High order accuracy, as well as other features such as fully adaptive time stepping, easy coupling with AD-generated code, two-level file checkpointing, make `DENSERKS` an attractive option for sensitivity calculations. We tested the feasibility and efficiency of our approach on a range of numerical examples. Their results indicate that, for non-stiff or mildly stiff problems, the performance of our code is comparable to that of existing high-quality implementations based on linear multistep methods.



(a) $\mathcal{J}\left(t^F, \mathbf{q}\right)$



(b) $\left\|\nabla_{\mathbf{q}}\mathcal{J}\left(t^F, \surd\right)\right\|$



(c) Convergence of $\mathbf{q}$ to $\mathbf{q}^{\text{ref}} = (1, 0.5)$

Figure 4.6: The convection-diffusion equation: Optimization results using `L-BFGS-B`. (a) The decrease in the cost function $\mathcal{J}\left(t^F, \mathbf{q}\right)$ and (b) the decrease in the projected norm of the gradient of the cost function versus `L-BFGS-B` iteration count (M). (c) The convergence of the parameters towards the reference values $\left(\mathbf{q}_1^{\text{ref}}, \mathbf{q}_2^{\text{ref}}\right) = (1, 0.5)$, where $\mathcal{J}\left(t^F, \mathbf{q}^{\text{ref}}\right) = 0$. The starting point is $(\mathbf{q}_1^0, \mathbf{q}_2^0) = (3, 3)$ and the size of the spatially discretized forward ODE system is $n_y = 70$.

4.2 The *discretize - then - differentiate* approach

4.2.1 Introduction and motivation

The behavior of numerical integration algorithms under automatic differentiation was first studied in the broader context of general iterative solvers [140, 141]. It has been shown that derivatives of iterative solvers converge to the derivative of the original solution, under suitable assumptions [141]. Hager [33] investigated Runge–Kutta adjoints for distributed optimal control. He formulated the adjoint of a Runge–Kutta scheme of order $q \leq 4$, through a full discretization approach where both the forward and the adjoint systems are treated together. On the other hand, as explained in section 4.2.3, the reverse mode of AD employs a recursive discretization.

Discrete adjoints of numerical integration algorithms are attractive, because, unlike solvers for the continuous adjoint equations, they can be generated automatically using AD. Walther [95] proved that explicit Runge–Kutta methods retain their order of accuracy q under reverse mode AD, for $q \leq 4$. Sandu [122] extended the results to both explicit and implicit Runge–Kutta methods of arbitrary order q. Hager, Sandu and Walther all assume that any adaptive mechanisms, if present, are not differentiated. Hager and Walter also considered distributed controls in their work, which further complicates the analysis. The analysis in [122], similar to the present work, considers only initial conditions and time independent parameters as control variables.

Adaptive features induce additional complications when differentiating a numerical integration algorithm. This issue is of great importance, since virtually all modern ODE integrators make use of time step controllers, error estimators, or numerical extrapolation to maximize efficiency and speed. In some cases, one can isolate these parts of the code and prevent them from being differentiated, but the difficulty of this approach is highly problem dependent, and such decoupling is certainly far from trivial for more complicated large-scale integration codes [4].

Forward mode automatic differentiation of the adaptive mechanisms leads to spurious derivatives, as demonstrated by Eberhard et. al. [4]. These non-physical sensitivities introduce large errors in the tangent linear model trajectory, leading to an incorrect TLM trajectory. Analyzing explicit Runge–Kutta - like methods, Eberhard et. al. proposed a code correction that restores the accuracy of the discrete TLM solution. Also, they noted that forward mode AD will improve the accuracy of the final-time sensitivities when the last time step is trimmed such that the integrator stops at exactly the final integration time [4].

In this section, we investigate the behavior of adaptive ODE integration schemes under reverse-mode automatic differentiation. We show that the discrete adjoints of adaptive methods are inconsistent with the adjoint ODE due to the differentiation of the time step controller. Adjoining results in more complex code than forward mode differentiation, due to

the reversal of the original control flow. We build the discrete adjoint of a general adaptive, one-step explicit integration method, and quantify the perturbation introduced by the adjoint time step gradients in the adjoint solution. The discrete adjoint solutions are found to have a $\mathcal{O}(1)$ error. Hence, the naive invocation of reverse mode AD yields incorrect gradients. This is shown to hold for the particular case of a q-th order Runge–Kutta method. Thus, one has to eliminate the perturbations induced by AD before the discrete adjoint solution can be trusted to be accurate.

We propose two equivalent and easy to implement code corrections, that make the discrete adjoint consistent with its continuous counterpart. Moreover, we extend our analysis to second order adjoints of adaptive integrators, and show how to cancel the influence of the spurious derivatives.

We also investigate the effect of adjusting the time step of the integration to arrive exactly at a given final integration time. Eberhard et. al. [4] noted that a forward mode differentiation of this time step adjustment will act as a correction to the discrete tangent linear variables. For simplicity of exposition, we derive the tangent linear and adjoint models of a three-time step integration routine. We show that, while there is indeed an improvement in the accuracies of the tangent linear and adjoint model trajectories when the last time step is trimmed, not all spurious derivatives are eliminated. Thus, this strategy does not lead to sensitivities or gradients equal to those resulting from a full passivation of the time step controller (i.e., setting all spurious sensitivities to zero).

4.2.2 Forward and adjoint sensitivity analysis

Consider again the dynamical system modeled by the IVP (4.1). Through a simple change of variables, and without loss of generality [142], one can recast (4.11) into the following inverse problem:

$$\text{Find } \mathbf{y}^0_* = \arg\min_{\mathbf{y}^0} \mathcal{J} := \widehat{\mathcal{J}}(y^F) \,, \tag{4.58}$$

where the primal variable $\mathbf{y}(t)$ now satisfies the equation

$$\begin{aligned} \dot{\mathbf{y}} &= F(t, \mathbf{y}) \;,\; t^0 \le t \le t^F \,, \\ \mathbf{y}(t^0) &= \mathbf{y}^0 \,. \end{aligned} \tag{4.59}$$

Comparing this with equation (4.1), we note that (with a slight abuse of notation)

$$\mathbf{y} := \begin{bmatrix} \mathbf{y} \\ \mathbf{q} \\ \theta \end{bmatrix} \in \mathbb{R}^N \,, \quad F := \begin{bmatrix} F \\ 0 \\ \mathbf{g} \end{bmatrix} \,, \quad N := 1 + n_y + n_q \,. \quad \theta(t^0) = 0 \,. \tag{4.60}$$

The propagation of small perturbations $\boldsymbol{\delta}\mathbf{y}^0$ in the initial conditions of (4.59) is governed by the tangent linear model [46, 129, 30, 72]:

$$\begin{aligned} \boldsymbol{\delta}\dot{\mathbf{y}} &= \frac{\partial F}{\partial \mathbf{y}}(t,\mathbf{y})\,\boldsymbol{\delta}\mathbf{y}\ ,\ t^0 \le t \le t^F\ , \\ \boldsymbol{\delta}\mathbf{y}(t^0) &= \frac{\partial \mathbf{y}}{\partial \mathbf{y}^0}(t^0)\,. \end{aligned} \tag{4.61}$$

A small perturbation $\boldsymbol{\delta}\mathbf{y}^0$ in the initial condition of (4.59) will cause a small change in $\widehat{\mathcal{J}}$:

$$\boldsymbol{\delta}\widehat{\mathcal{J}} = \frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{y}^0}\,\boldsymbol{\delta}\mathbf{y}^0\,, \tag{4.62}$$

up to first order in $\boldsymbol{\delta}\mathbf{y}^0$. Thus, we will need N TLM integrations with N linearly independent perturbations to obtain the complete gradient vector $\nabla_{\mathbf{y}^0}\widehat{\mathcal{J}}$. On the other hand, the adjoint of (4.59) can yield the same gradient at the cost of a single backward-time integration:

$$\begin{aligned} \dot{\boldsymbol{\lambda}} &= -\left(\frac{\partial F}{\partial \mathbf{y}}(t,\mathbf{y})\right)^T \boldsymbol{\lambda}\ ,\quad t^F \ge t \ge t^0\ , \\ \boldsymbol{\lambda}(t^F) &= \left(\frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{y}}\right)^T \big(\mathbf{y}(t^F)\big)\ , \end{aligned} \tag{4.63}$$

Then:

$$\frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{y}^0} = \boldsymbol{\lambda}(t^0)\,. \tag{4.64}$$

Higher order derivatives [70] have proved to be useful in areas such as data assimilation and chemistry transport modeling [72, 71]. The second order adjoint framework provides sensitivity information in the form of Hessian - vector products. The second order derivatives can be obtained from the final value problem [69, 72]:

$$\begin{aligned} \dot{\boldsymbol{\sigma}} &= -\left(\frac{\partial F}{\partial \mathbf{y}}(t,\mathbf{y})\right)^T \sigma - \left(\frac{\partial^2 F}{\partial \mathbf{y}^2}(t,\mathbf{y}) \otimes \boldsymbol{\delta}\mathbf{y}(t)\right)^T \boldsymbol{\lambda} \\ \boldsymbol{\sigma}(t^F) &= \frac{\partial^2 \widehat{\mathcal{J}}}{\partial \mathbf{y}^2}(\mathbf{y}(t^F)) \cdot \boldsymbol{\delta}\mathbf{y}(t^F)\,, \end{aligned} \tag{4.65}$$

where $\boldsymbol{\delta}\mathbf{y}$ is the solution of the TLM (4.61). One can then show that

$$\boldsymbol{\sigma}(t^0) = \frac{\partial^2 \widehat{\mathcal{J}}}{\partial \mathbf{y}^2}(\mathbf{y}(t^0)) \cdot \boldsymbol{\delta}\mathbf{y}^0\,. \tag{4.66}$$

For cost functionals such as (4.58) and large N, equations (4.61) and (4.63) show that it is considerably more efficient to compute the gradient of the objective functional through the adjoint method, since only one adjoint model solve is required. One can compute adjoint sensitivities numerically through one of the following methods:

a. Use the *differentiate - then - discretize* approach: see the previous section of this chapter. Note that we can use AD to generate the right-hand side of the adjoint ODE, but the Runge–Kutta method itself is not differentiated. Assuming the numerical scheme is stable, the algorithm will yield a q-th order accurate approximation to the adjoint $\boldsymbol{\lambda}(t)$. One can successfully employ this method when the integration code is well separated from the model. This allows an easy division of labor where the developers can separately focus on their domain of expertise. However, this approach may entail a significant programming effort, if a suitable adjoint code is not readily available.

b. The alternative is to *discretize - then - differentiate* the IVP (4.59). First, one discretizes the forward model equations, and then integrates this discrete model using an adaptive time stepping method. The next step is to build the *discrete adjoint of the numerical integrator* using reverse mode AD. This approach is obviously attractive since it can be done in a fully automatic fashion. Moreover, the adjoint variables are the exact sensitivities of the discretized cost functional. It is the method of choice for practitioners when the integration and model codes are strongly coupled, which is often the case in large-scale legacy codes. However, there are no a priori guarantees that the black-box application of reverse mode AD on an adaptive numerical integration method will result in a correct adjoint model, for the reasons explained below.

We assume that the forward integration algorithm gives a p-th order accurate approximation to the exact solution $\mathbf{y}(t^F)$. We investigate numerical methods that *retain their accuracy under adjoining if the time step h is kept fixed*, i.e. the discrete adjoint solution $\boldsymbol{\lambda}^n$ satisfies

$$|\boldsymbol{\lambda}^n - \boldsymbol{\lambda}(t^n)| \sim \mathcal{O}\left(h^q\right) , \quad \forall\, n \ , \ \text{as } h \to 0 \ . \tag{4.67}$$

This is true for Runge–Kutta methods [122, 95], but not for multistep methods [96]. For adaptive algorithms, the time steps taken during the forward integration will depend on the forward solution $\mathbf{y}^n$. Reverse mode AD will pick up this dependence and generate non-physical adjoint time step gradients. We will show that these spurious derivatives lead to incorrect discrete adjoints. To recover an accurate adjoint solution, one has to eliminate the AD artifacts from the discrete adjoint update, as discussed in sections 4.2.3 and 4.2.4.

Another method to prevent spurious adjoint gradients would be to avoid the differentiation of the time step controllers or error estimators. The difficulty of this approach is problem dependent; parts of the code must be isolated manually, and that can be a cumbersome task for tightly coupled large-scale models.

4.2.3 A general framework for adaptive-step differentiation

One-step, explicit adaptive integration schemes for (4.59) can be written as

$$\begin{aligned} \mathbf{y}^{n+1} &= \mathbf{f}\left(\mathbf{y}^n, h^n, t^n\right) && \text{(4.68a)} \\ h^{n+1} &= \mathbf{g}\left(\mathbf{y}^n, h^n, t^n\right) && \text{(4.68b)} \\ t^{n+1} &= t^n + h^n \,, \quad 0 \le n \le M-1 \,. && \text{(4.68c)} \end{aligned}$$

Here M denotes the total number of integration steps, i.e. $t^M = t^F$. $\mathbf{f} : \mathbb{R}^{N+2} \to \mathbb{R}^N$ is chosen according to some prescribed accuracy criteria, and $\mathbf{g} : \mathbb{R}^{N+2} \to \mathbb{R}$ (not to be confused with (4.12)) is a step size controller, used to compute the new step size h^{n+1} based on an estimate of the local error at t^n. The initial time step h^0 is assumed to be equal to a sufficiently small constant. Superscripts are used to indicate discrete time moments, while subscripts denote partial derivatives, unless noted otherwise.

We consider methods with the following properties. For a fixed time step $h^n = h$, the method

$$\mathbf{y}^{n+1} = \mathbf{f}(\mathbf{y}^n, h, t^n) \ : \tag{4.69}$$

a. is asymptotically consistent with the adjoint ODE (4.63),

b. is accurate of order q, and

c. retains its order of accuracy under forward or reverse-mode differentiation, i.e., its tangent linear and adjoint formulations are q-th order accurate numerical schemes for the solution of the tangent linear and adjoint ODE, respectively.

Any q-th order Runge–Kutta method satisfies these assumptions [122].

The discrete adjoint of an integration scheme with above properties reads [72]:

$$\begin{aligned} \boldsymbol{\lambda}^n &= \left(\frac{\partial f}{\partial \mathbf{y}}\left(\mathbf{y}^n, h^n, t^n\right)\right)^T \boldsymbol{\lambda}^{n+1} \ , \ M-1 \ge n \ge 0 \,, \\ \boldsymbol{\lambda}^M &= \frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{y}^M} \,. \end{aligned} \tag{4.70}$$

We will show that the reverse mode of AD applied to (4.68a – 4.68c) does *not* yield (4.70). Instead, the AD engine introduces spurious derivatives of the time step controller into the discrete adjoint solution. Our correction eliminates the AD artifacts. Consider the integrator state vector for (4.68a – 4.68c):

$$\vec{\mathbf{v}}^n = \begin{bmatrix} \mathbf{y}^n \\ h^n \\ t^n \end{bmatrix} \,. \tag{4.71}$$

The derivation of the tangent linear model of (4.68a – 4.68c) amounts to constructing the local Jacobian matrix [143] for the state evolution

$$\mathbf{v}^n \quad \rightarrow \quad \mathbf{v}^{n+1}\,. \tag{4.72}$$

The TLM of (4.68a) – (4.68c) has the following form:

$$\begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^{n+1} \\ \boldsymbol{\delta}h^{n+1} \\ \boldsymbol{\delta}t^{n+1} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{y}} & \frac{\partial \mathbf{f}}{\partial h} & \frac{\partial \mathbf{f}}{\partial t} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{y}} & \frac{\partial \mathbf{g}}{\partial h} & \frac{\partial \mathbf{g}}{\partial t} \\ \mathbf{0}_{1\times N} & 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^{n} \\ \boldsymbol{\delta}h^{n} \\ \boldsymbol{\delta}t^{n} \end{bmatrix} \quad , \quad 0 \le n \le M-1\,. \tag{4.73}$$

All partial derivatives are evaluated at $(\mathbf{y}^n, h^n, t^n)$. The adjoint model is built by transposing the Jacobian of (4.72). Denote the adjoint state vector at step n by

$$\bar{\boldsymbol{\lambda}}^n = \begin{bmatrix} \boldsymbol{\lambda}^n \\ \mu^n \\ \nu^n \end{bmatrix}\,. \tag{4.74}$$

The n-th step in the discrete adjoint of (4.68a – 4.68c) can be written as

$$\begin{bmatrix} \boldsymbol{\lambda}^n \\ \mu^n \\ \nu^n \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^T & \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^T & \mathbf{0}_{N\times 1} \\ \left(\frac{\partial \mathbf{f}}{\partial h}\right)^T & \frac{\partial \mathbf{g}}{\partial h} & 1 \\ \left(\frac{\partial \mathbf{f}}{\partial t}\right)^T & \frac{\partial \mathbf{g}}{\partial t} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}^{n+1} \\ \mu^{n+1} \\ \nu^{n+1} \end{bmatrix}\,. \tag{4.75}$$

This leads to:

$$\boldsymbol{\lambda}^n = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^T \boldsymbol{\lambda}^{n+1} + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^T \mu^{n+1} \tag{4.76a}$$

$$\mu^n = \left(\frac{\partial \mathbf{f}}{\partial h}\right)^T \lambda^{n+1} + \frac{\partial \mathbf{g}}{\partial h}\,\mu^{n+1} + \nu^{n+1} \tag{4.76b}$$

$$\nu^n = \left(\frac{\partial \mathbf{f}}{\partial t}\right)^T \boldsymbol{\lambda}^{n+1} + \frac{\partial \mathbf{g}}{\partial t}\,\mu^{n+1} + \nu^{n+1} \quad , \quad M-1 \ge n \ge 0\,. \tag{4.76c}$$

The last term in the adjoint update (4.76a) is a side-effect of AD, generated by the differentiation of the time step controller mechanism (4.68b). μ is a spurious adjoint derivative of the time step h. These perturbations can generate $\mathcal{O}(1)$ errors in the discrete adjoint solution at t^0:

$$\boldsymbol{\lambda}^0 = \boldsymbol{\lambda}(t^0) + \mathcal{O}(1)\,.$$

In section 4.2.4 we prove that this happens with Runge–Kutta methods. The numerical experiments in section 4.2.7 confirm this conclusion (see Figure 4.7 and Figure 4.8).

We present two strategies to eliminate the spurious gradients. One can:

1. Zero out the non-physical adjoint time step derivatives. This "passivation" effectively cancels the influence of the time step controller on the solution adjoints. In the adjoint code, we set:

$$\mu^{n+1} \quad \leftarrow \quad 0\,, \tag{4.77}$$

before the update (4.76a). This modification requires both the identification of the time step adjoint variable (straightforward once the user is familiar with the forward model code), and knowledge regarding the location of (4.77). Depending on the code complexity, (4.77) may be easier or harder to apply than the following alternative strategy.

2. Alternatively, implement a correction of the form:

$$\boldsymbol{\lambda}^n \quad \leftarrow \quad \boldsymbol{\lambda}^n - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^T \mu^{n+1}\,. \tag{4.78}$$

Suppose that (4.68b) is implemented in FORTRAN as

```
subroutine g (t,y,h,hNew)
```

with `h` $= h^n$ and `hNew` $= h^{n+1}$. When `t` $= t^n$, `y` $= \mathbf{y}^n$, and `h` $= h^n$ are chosen as the independent variables, and `hNew` $= h^{n+1}$ is the dependent variable, an AD tool such as TAMC [58], or TAPENADE [54], will generate:

```
subroutine adg (t,y,h,adh,ady,adt,adhNew) ,
```

with `adh` $= \mu^n$ and `adhNew` $= \mu^{n+1}$. Since the value of `adhNew` is lost after the call to `adg()`, we need to save it into a temporary variable and reuse it in (4.78):

```
tmp ← adhNew .
```

We now post-process the adjoint trajectory $\boldsymbol{\lambda}^n$ right after the call to `adg()` inserted by AD by calling

```
call adg (t,y,h,0,ady,0,-tmp) .
```

This call implements (4.78). Since an evaluation of $\mathbf{g}$ is computationally inexpensive relative to a call to $\mathbf{f}$, and the additional cost of adjoining is not greater than that of a (small) constant number of forward model evaluations [23], the overhead of (4.78) is small compared to the overall cost of an adjoint Runge–Kutta step.

4.2.4 Explicit adaptive Runge–Kutta methods

We now focus on explicit Runge–Kutta methods. Let $\mathbf{k}_1, \mathbf{k}_2 \ldots \mathbf{k}_r \in \mathbb{R}^N$ be the Runge–Kutta stages, which are now part of the discrete integrator state. Also, $\mathbf{k} \equiv [\mathbf{k}_1 \;\; \mathbf{k}_2 \;\; \ldots \;\; \mathbf{k}_r] \in \mathbb{R}^{N\times r}$. As in the previous section, we will identify the integrator state variables at t^n by the superscript n.

The Runge–Kutta method can be written as follows:

$$\begin{aligned}
\mathbf{k} &= \mathbf{k}\left(\mathbf{y}^n, h^n, t^n\right) && (4.79a)\\
\mathbf{y}^{n+1} &= \mathbf{y}^n + h^n \sum_{m=1}^{r} b_m \, \mathbf{k}_m && (4.79b)\\
h^{n+1} &= h^n \, \tilde{\mathbf{g}}\left(\mathbf{y}^n, t^n, \mathbf{k}\right) && (4.79c)\\
t^{n+1} &= t^n + h^n \;, \qquad 0 \le n \le M-1 \,. && (4.79d)
\end{aligned}$$

Here $b \in \mathbb{R}^r$ denote the Runge–Kutta weights. We assume that $\mathbf{k}^{n+1}$ are the stage values after the update (4.79d), and $\mathbf{k}^M = 0$. The time step controller (4.79c) has a form that is widely used in practice (see (4.120) for an example). Therefore, the forward integrator state is described by the following vector:

$$\mathbf{v}^n = \begin{bmatrix} \mathbf{k}_1^n \\ \vdots \\ \mathbf{k}_m^n \\ \mathbf{y}^n \\ h^n \\ t^n \end{bmatrix} . \tag{4.80}$$

The Runge–Kutta discrete adjoint consistent with the adjoint ODE (4.63) is computed using the chain rule of differentiation:

$$\begin{aligned}
\boldsymbol{\lambda}^n &= \left(\mathbf{I}_N + h^n \sum_{m=1}^{r} b_m \frac{\partial \mathbf{k}_m}{\partial \mathbf{y}} \left(t^n, h^n, \mathbf{y}^n\right) \right)^T \boldsymbol{\lambda}^{n+1} \,, \quad M-1 \ge n \ge 1 \,, \\
\boldsymbol{\lambda}^M &= \frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{y}^M} \,. && (4.81)
\end{aligned}$$

Next, we will show that the discrete adjoint of (4.79a) – (4.79d) given by AD is *different* from (4.81).

Since both forward-mode differentiation and adjoining can be performed line by line [23], we can first build the TLM for (4.79a) and then for (4.79b) – (4.79d). This mimics the behavior of AD, since it necessary to account for the dependency arising between (4.79a) and (4.79b)

through the stage values k. Hence, the n-th step of the TLM reads:

$$\begin{bmatrix} \boldsymbol{\delta}\mathbf{k}_1^{n+1/2} \\ \vdots \\ \boldsymbol{\delta}\mathbf{k}_r^{n+1/2} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \mathbf{k}_1}{\partial y} & \dfrac{\partial \mathbf{k}_1}{\partial h} & \dfrac{\partial \mathbf{k}_1}{\partial t} \\ \vdots & \vdots & \vdots \\ \dfrac{\partial \mathbf{k}_r}{\partial y} & \dfrac{\partial \mathbf{k}_r}{\partial h} & \dfrac{\partial \mathbf{k}_r}{\partial t} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^n \\ \boldsymbol{\delta}h^n \\ \boldsymbol{\delta}t^n \end{bmatrix} \tag{4.82}$$

$$\begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^{n+1} \\ \boldsymbol{\delta}h^{n+1} \\ \boldsymbol{\delta}t^{n+1} \end{bmatrix} = \begin{bmatrix} h^n b_1 \mathbf{I}_N & \dots & h^n b_r \mathbf{I}_N & \mathbf{I}_N & \displaystyle\sum_m b_m \mathbf{k}_m & \mathbf{0}_{N\times 1} \\ h^n \dfrac{\partial \tilde{\mathbf{g}}}{\partial k_1} & \dots & h^n \dfrac{\partial \tilde{\mathbf{g}}}{\partial k_r} & h^n \dfrac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{y}} & \tilde{\mathbf{g}} & h^n \dfrac{\partial \tilde{\mathbf{g}}}{\partial t} \\ \mathbf{0}_{1\times N} & \dots & \mathbf{0}_{1\times N} & \mathbf{0}_{1\times N} & 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}\mathbf{k}_1^{n+1/2} \\ \vdots \\ \boldsymbol{\delta}\mathbf{k}_r^{n+1/2} \\ \boldsymbol{\delta}\mathbf{y}^n \\ \boldsymbol{\delta}h^n \\ \boldsymbol{\delta}t^n \end{bmatrix} \tag{4.83}$$

Here $\mathbf{I}_N$ denotes the N-by-N identity matrix, $\mathbf{0}_N$ stands for the zero N-by-N matrix, and the index $n+1/2$ indicates an intermediate step. We note that all entries of the Jacobian matrices are evaluated at $(t^n, \mathbf{y}^n, h^n)$. The adjoint variables form the costate vector:

$$\bar{\boldsymbol{\lambda}}^n = \begin{bmatrix} \boldsymbol{\psi}_1^n \\ \vdots \\ \boldsymbol{\psi}_r^n \\ \boldsymbol{\lambda}^n \\ \mu^n \\ \nu^n \end{bmatrix} . \tag{4.84}$$

The discrete adjoint follows immediately by transposing the Jacobian matrices and reversing the direction of integration:

$$\begin{aligned} \boldsymbol{\psi}_m^{n+1/2} &= h^n b_m \boldsymbol{\lambda}^{n+1} + h^n \left(\frac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{k}_m} \right)^T \mu^{n+1} \, , \; m = 1 \dots r \\ \boldsymbol{\lambda}^{n+1/2} &= \boldsymbol{\lambda}^{n+1} + h^n \left(\frac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{y}} \right)^T \mu^{n+1} \\ \mu^{n+1/2} &= \mu^{n+1} + \tilde{\mathbf{g}} \mu^{n+1} + \sum_{m=1}^{r} b_m \mathbf{k}_m^T \boldsymbol{\lambda}^{n+1} \\ \nu^{n+1/2} &= h^n \frac{\partial \tilde{\mathbf{g}}}{\partial t} \mu^{n+1} + \nu^{n+1} \, , \end{aligned} \tag{4.85}$$

and

$$\begin{aligned}
\boldsymbol{\lambda}^n &= \sum_{m=1}^{r} \left(\frac{\partial \mathbf{k}_m}{\partial y} \right)^T \boldsymbol{\psi}_m^{n+1/2} + \boldsymbol{\lambda}^{n+1/2} \\
\mu^n &= \sum_{m=1}^{r} \left(\frac{\partial \mathbf{k}_m}{\partial h} \right)^T \boldsymbol{\psi}_m^{n+1/2} + \mu^{n+1/2} \\
\nu^n &= \sum_{m=1}^{r} \left(\frac{\partial \mathbf{k}_m}{\partial t} \right)^T \boldsymbol{\psi}_m^{n+1/2} + \nu^{n+1/2} \;, \qquad M-1 \geq n \geq 0 \;.
\end{aligned} \tag{4.86}$$

Thus AD computes the following discrete adjoint update:

$$\begin{aligned}
\boldsymbol{\lambda}^n &= \left(\mathbf{I}_N + h^n \sum_{m=1}^{r} b_m \left(\frac{\partial \mathbf{k}_m}{\partial \mathbf{y}} \right)^T \right) \boldsymbol{\lambda}^{n+1} \\
&\quad + h^n \mu^{n+1} \left(\sum_{m=1}^{r} b_m \left(\frac{\partial \mathbf{k}_m}{\partial \mathbf{y}} \right)^T \left(\frac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{k}_m} \right)^T + \left(\frac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{y}} \right)^T \right) \\
&= \left(\mathbf{I}_N + h^n \sum_{m=1}^{r} b_m \left(\frac{\partial \mathbf{k}_m}{\partial \mathbf{y}} \right)^T \right) \boldsymbol{\lambda}^{n+1} + \mathcal{O}\left(h^n\right) \;.
\end{aligned} \tag{4.87}$$

All partial derivatives are evaluated at $(\mathbf{y}^n, h^n, t^n)$. The update (4.87) is clearly different from (4.81). The $\mathcal{O}\left(h^n\right)$ perturbations introduced at each step in the discrete adjoint $\boldsymbol{\lambda}^n$ add up to a $\mathcal{O}\left(1\right)$ perturbation in the adjoint solution $\boldsymbol{\lambda}^0$:

$$\left| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}(t^0) \right| = \mathcal{O}(1) \,. \tag{4.88}$$

To eliminate all perturbations, one can zero out the adjoint derivative μ^{n+1}, as in (4.77). Alternatively, we can insert

$$\begin{aligned}
\boldsymbol{\psi}_m^{n+1/2} &\leftarrow \boldsymbol{\psi}_m^{n+1/2} - h^n \left(\frac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{k}_m} \right)^T \mu^{n+1} \;,\; m = 1 \ldots r \\
\boldsymbol{\lambda}^{n+1/2} &\leftarrow \boldsymbol{\lambda}^{n+1/2} - h^n \left(\frac{\partial \tilde{\mathbf{g}}}{\partial \mathbf{y}} \right)^T \mu^{n+1} \,.
\end{aligned} \tag{4.89}$$

after (4.85). The FORTRAN implementation of (4.89) is based on a second call of `adg` (the adjoint of (4.79c)), and is very similar to the one described in the previous section. Both correction strategies will result in a discrete adjoint solution that is a q-th order accurate approximation to the true adjoint $\boldsymbol{\lambda}(t^0)$. Section 4.2.7 shows the divergent AD adjoint, as well as the convergence of the corrected solution, for a 5-th order Runge–Kutta method.

4.2.5 Final time step adjustment

Like in the case of forward mode differentiation [4], adjusting the last time step before the final integration time does usually improve the accuracy of the adjoint model trajectory.

However, not all spurious gradients present in the uncorrected discrete adjoint (4.76a) are eliminated. We show this by considering a three step integration procedure. We let

$$t^F = t^3 = t^0 + h^0 + h^1 + h^2 \,. \tag{4.90}$$

where t^0 and h^0 are fixed, and h_1, h_2 are estimated using (4.68b). We investigate four different test cases:

a. The spurious sensitivities δh^i, δt^i, and gradients μ^i, ν^i are set to zero in the AD-generated code. This obviously ensures the correctness of both linearizations.

b. h^2 is artificially adjusted:
$$h^2 = t^3 - t^2 \tag{4.91}$$
This is not strictly needed in our algorithm (because of (4.90)). However, we will look at the impact of (4.91) on the discrete tangent linear and adjoint solution variables.

c. No changes are made to the AD-generated tangent linear and adjoint codes.

d. The TLM solution is post processed as described in [4]. The discrete adjoint code is corrected using (4.78).

We investigate the behavior of the tangent linear and adjoint trajectories in all of the above cases.

Tangent linear differentiation

The cases under discussion lead to different formulations of the tangent linear model:

a. If the TLM is modified to ensure to ensure $\delta h^1 = \delta h^2 = 0$ and $\delta t^1 = \delta t^2 = 0$, then we have:
$$\begin{bmatrix} \widetilde{\boldsymbol{\delta}\mathbf{y}^3} \\ \widetilde{\boldsymbol{\delta}h^3} \\ \widetilde{\boldsymbol{\delta}t^3} \end{bmatrix} = \mathcal{A}_2^3\,\mathcal{H}\,\mathcal{A}_1^2\,\mathcal{H}\,\mathcal{A}_0^1 \begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^0 \\ \boldsymbol{\delta}h^0 = 0 \\ \boldsymbol{\delta}t^0 = 0 \end{bmatrix} , \tag{4.92}$$
where
$$\mathcal{H} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0}_{N\times 1} & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 0 & 0 \\ \mathbf{0}_{1\times N} & 0 & 0 \end{bmatrix} , \tag{4.93}$$
and
$$\mathcal{A}_i^{i+1} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial \mathbf{y}} & \dfrac{\partial \mathbf{f}}{\partial h} & \dfrac{\partial \mathbf{f}}{\partial t} \\ \dfrac{\partial \mathbf{g}}{\partial \mathbf{y}} & \dfrac{\partial \mathbf{g}}{\partial h} & \dfrac{\partial \mathbf{g}}{\partial t} \\ \mathbf{0}_{1\times N} & 1 & 1 \end{bmatrix}_{(\mathbf{y}^i, h^i, t^i)} , \tag{4.94}$$
for $i = 1, 2, 3$.

b. The time step adjustment introduces an extra step in the tangent linear model:

$$\begin{bmatrix} \widehat{\boldsymbol{\delta}\mathbf{y}^3} \\ \widehat{\boldsymbol{\delta}h^3} \\ \widehat{\boldsymbol{\delta}t^3} \end{bmatrix} = \mathcal{A}_2^3 \, \mathcal{B} \, \mathcal{A}_1^2 \, \mathcal{A}_0^1 \begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^0 \\ \boldsymbol{\delta}h^0 = 0 \\ \boldsymbol{\delta}t^0 = 0 \end{bmatrix} . \tag{4.95}$$

Here $\mathcal{B}$ is the Jacobian of the time step adjustment (4.91):

$$\mathcal{B} = \begin{bmatrix} \mathbf{I}_N & 0_{N\times 1} & 0_{N\times 1} \\ 0_{1\times N} & 0 & -1 \\ \mathbf{0}_{1\times N} & 0 & 1 \end{bmatrix} . \tag{4.96}$$

c. The TLM is similar to (4.95), except $\mathcal{B} = \mathbf{I}_{N+2}$, since no time step adjustment is made:

$$\begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^3 \\ \boldsymbol{\delta}h^3 \\ \boldsymbol{\delta}t^3 \end{bmatrix} = \mathcal{A}_2^3 \, \mathcal{A}_1^2 \, \mathcal{A}_0^1 \begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^0 \\ \boldsymbol{\delta}h^0 = 0 \\ \boldsymbol{\delta}t^0 = 0 \end{bmatrix} . \tag{4.97}$$

d. We cannot easily fit the post-processing error correction formula given in [4] into our general framework without assuming a specific Runge–Kutta method. Hence, we will only report the numerical results obtained with this correction for a model problem in section 4.2.5.

We also note that the tangents of the first time step and time point are zero, since these quantities are assumed to be constant. Multiplying through in (4.92)–(4.97) gives:

$$\widetilde{\boldsymbol{\delta}y_3} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_2 \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_1 \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_0 \boldsymbol{\delta}\mathbf{y}^0 , \tag{4.98a}$$

$$\widehat{\boldsymbol{\delta}y^3} = \widetilde{\boldsymbol{\delta}y^3} + \left[\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_2 \left(\frac{\partial \mathbf{f}}{\partial \mathbf{h}}\right)_1 - \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2 + \left(\frac{\partial \mathbf{f}}{\partial t}\right)_2\right] \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_0 \boldsymbol{\delta}\mathbf{y}^0 , \tag{4.98b}$$

$$\begin{aligned} \boldsymbol{\delta}\mathbf{y}^3 = \widetilde{\boldsymbol{\delta}\mathbf{y}^3} &+ \left[\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_2 \left(\frac{\partial \mathbf{f}}{\partial h}\right)_1 + \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2 \left(\frac{\partial \mathbf{g}}{\partial h}\right)_1 + \left(\frac{\partial \mathbf{f}}{\partial t}\right)_2\right] \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_0 \boldsymbol{\delta}\mathbf{y}^0 \\ &+ \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2 \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_1 \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_0 \boldsymbol{\delta}\mathbf{y}^0 . \end{aligned} \tag{4.98c}$$

In the above equations, the integer subscript on the partial derivatives denotes the time level where the Jacobians are evaluated at, e.g.,

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_i = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)(\mathbf{y}^i, h^i, t^i) . \tag{4.99}$$

As shown in [4], $\widetilde{\boldsymbol{\delta}\mathbf{y}^3} \neq \delta\mathbf{y}^3$. Moreover, it is apparent that adjusting the final time step with (4.91) eliminates only part of the spurious terms in $\boldsymbol{\delta}\mathbf{y}^3$. Hence $\widetilde{\boldsymbol{\delta}\mathbf{y}^3} \neq \widehat{\boldsymbol{\delta}\mathbf{y}^3}$. For an arbitrary $\boldsymbol{\delta}\mathbf{y}^0$, we can expect the perturbation in $\widehat{\boldsymbol{\delta}\mathbf{y}^3}$ (equation (4.98b)) to be small if

$$\left[\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_2 \left(\frac{\partial \mathbf{f}}{\partial h}\right)_1 + \left(\frac{\partial \mathbf{f}}{\partial t}\right)_2 - \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2\right] \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_0 , \tag{4.100}$$

is small with respect to $\|\boldsymbol{\delta}\mathbf{y}^0\|$, but this may not always be the case. The spurious derivatives in $\widehat{\boldsymbol{\delta}\mathbf{y}^3}$ may accumulate over time and reduce the order of accuracy of the TLM solution.

The discrete adjoint

The discrete adjoints for the test cases a., b., and c., are obtained by transposing each sequence of TLM transformations (4.92), (4.95), and (4.97), respectively:

$$\begin{bmatrix} \widetilde{\boldsymbol{\lambda}^0} \\ \widetilde{\mu^0} \\ \widetilde{\nu^0} \end{bmatrix} = \left(\mathcal{A}_0^1\right)^T \mathcal{H}^T \left(\mathcal{A}_1^2\right)^T \mathcal{H}^T \left(\mathcal{A}_2^3\right)^T \begin{bmatrix} \boldsymbol{\lambda}^3 \\ \mu^3 = 0 \\ \nu^3 = 0 \end{bmatrix} , \tag{4.101a}$$

$$\begin{bmatrix} \widehat{\boldsymbol{\lambda}^0} \\ \widehat{\mu^0} \\ \widehat{\nu^0} \end{bmatrix} = \left(\mathcal{A}_0^1\right)^T \left(\mathcal{A}_1^2\right)^T \mathcal{B}^T \left(\mathcal{A}_2^3\right)^T \begin{bmatrix} \boldsymbol{\lambda}^3 \\ \mu^3 = 0 \\ \nu^3 = 0 \end{bmatrix} , \tag{4.101b}$$

$$\begin{bmatrix} \boldsymbol{\lambda}^0 \\ \mu^0 \\ \nu^0 \end{bmatrix} = \left(\mathcal{A}_0^1\right)^T \left(\mathcal{A}_1^2\right)^T \left(\mathcal{A}_2^3\right)^T \begin{bmatrix} \boldsymbol{\lambda}^3 \\ \mu^3 = 0 \\ \nu^3 = 0 \end{bmatrix} . \tag{4.101c}$$

Additionally, we look at the corrected adjoint, where

$$\begin{bmatrix} \boldsymbol{\lambda}_c^0 \\ \mu_c^0 \\ \nu_c^0 \end{bmatrix} = \mathcal{C}_0^1 \, \mathcal{C}_1^2 \, \mathcal{C}_2^3 \begin{bmatrix} \boldsymbol{\lambda}^3 \\ \mu^3 = 0 \\ \nu^3 = 0 \end{bmatrix} . \tag{4.102}$$

Here $\mathcal{C}_0^1$, $\mathcal{C}_1^2$, and $\mathcal{C}_2^3$ are the corrected (transposed) Jacobians:

$$\mathcal{C}_i^{i+1} = \begin{bmatrix} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^T & -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right) + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^T & \mathbf{0}_{N\times 1} \\ \left(\frac{\partial \mathbf{f}}{\partial h}\right)^T & \frac{\partial \mathbf{g}}{\partial h} & 1 \\ \left(\frac{\partial \mathbf{f}}{\partial t}\right)^T & \frac{\partial \mathbf{g}}{\partial t} & 1 \end{bmatrix}_{(\mathbf{y}^i, h^i, t^i)} = \begin{bmatrix} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^T & \mathbf{0}_{N\times 1} & \mathbf{0}_{N\times 1} \\ \left(\frac{\partial \mathbf{f}}{\partial h}\right)^T & \frac{\partial \mathbf{g}}{\partial h} & 1 \\ \left(\frac{\partial \mathbf{f}}{\partial t}\right)^T & \frac{\partial \mathbf{g}}{\partial t} & 1 \end{bmatrix}_{(\mathbf{y}^i, h^i, t^i)} \tag{4.103}$$

Hence:

$$\widetilde{\boldsymbol{\lambda}^0} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_0^T \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_1^T \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_2^T \boldsymbol{\lambda}^3 \,, \tag{4.104a}$$

$$\widehat{\boldsymbol{\lambda}^0} = \widetilde{\boldsymbol{\lambda}^0} + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_0^T \left[\left(\frac{\partial \mathbf{f}}{\partial h}\right)_1^T \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_2^T - \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2^T + \left(\frac{\partial \mathbf{f}}{\partial t}\right)_2^T\right] \boldsymbol{\lambda}^3 \,, \tag{4.104b}$$

$$\begin{aligned}\boldsymbol{\lambda}^0 = \widetilde{\boldsymbol{\lambda}^0} &+ \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_0^T \left[\left(\frac{\partial \mathbf{f}}{\partial h}\right)_1^T \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2^T + \left(\frac{\partial \mathbf{g}}{\partial h}\right)_1 \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2^T + \left(\frac{\partial \mathbf{f}}{\partial t}\right)_2^T\right] \boldsymbol{\lambda}^3 \\ &+ \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)_0^T \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)_1^T \left(\frac{\partial \mathbf{f}}{\partial h}\right)_2^T \boldsymbol{\lambda}^3 \,,\end{aligned} \tag{4.104c}$$

$$\boldsymbol{\lambda}_c^0 = \widetilde{\boldsymbol{\lambda}^0} \,. \tag{4.104d}$$

If the last time step is adjusted in the forward code, the discrete adjoint $\widehat{\boldsymbol{\lambda}^0} \neq \widetilde{\boldsymbol{\lambda}^0}$, therefore the correction (4.78) is still necessary.

A numerical example

We implement the three-step algorithms discussed above for a scalar IVP:

$$\begin{aligned}\dot{\mathbf{y}} &= \gamma\, t\, \mathbf{y} \,, \\ \mathbf{y}(0) &= 1 \;\;,\;\; t^0 = 0 \leq t \leq t^3 \approx 0.1031 \,,\end{aligned} \tag{4.105}$$

with $\gamma = -1$. The exact solution of (4.105) is

$$\mathbf{y}(t) = \exp\left(\gamma \frac{t^2}{2}\right) . \tag{4.106}$$

Since (4.105) is a linear IVP, its tangent linear model has the exact same form. We choose the same initial condition $\boldsymbol{\delta} y(0) = \mathbf{y}(0) = 1$, hence $\boldsymbol{\delta}\mathbf{y}(t) = \mathbf{y}(t)$ for all t.

The final value problem adjoint to (4.105) is:

$$\begin{aligned}\dot{\boldsymbol{\lambda}} &= -\gamma\, t\, \boldsymbol{\lambda} \,, \\ \lambda(t^3) &= \mathbf{y}(t^0) = 1 \;\;,\;\; t^3 \geq t \geq t^0 \,,\end{aligned} \tag{4.107}$$

with

$$\boldsymbol{\lambda}(t^0) = \mathbf{y}(t^3) = \exp\left(\gamma \frac{(t^3)^2}{2}\right) . \tag{4.108}$$

Table 4.5: Errors in the discrete tangent linear solutions for the IVP (4.105) induced by the spurious AD-generated sensitivities. Here, $\boldsymbol{\delta}\mathbf{y}_c^3$ is the post-processed TLM solution, using the error correction formula in [4]. $\widetilde{\boldsymbol{\delta}\mathbf{y}^3}$ and $\widehat{\boldsymbol{\delta}\mathbf{y}^3}$ are defined in (4.98a) and (4.98b), respectively. The time step adjustment (4.91), and the a posteriori error correction [4], lead to improvements over the uncorrected solution $\boldsymbol{\delta}\mathbf{y}^3$. However, these two approaches are not equivalent to a full passivation of the time step controller mechanism. The extra derivatives in (4.98b) and (4.98c) are found to exactly account for these errors.

	$\left\Vert \widetilde{\boldsymbol{\delta}\mathbf{y}^3} - \boldsymbol{\delta}\mathbf{y}^3 \right\Vert$		1.61063×10^{-4}
	$\left\Vert \widetilde{\boldsymbol{\delta}\mathbf{y}^3} - \widehat{\boldsymbol{\delta}\mathbf{y}^3} \right\Vert$		4.37644×10^{-6}
	$\left\Vert \widetilde{\boldsymbol{\delta}\mathbf{y}^3} - \boldsymbol{\delta}\mathbf{y}_c^3 \right\Vert$		4.52807×10^{-6}

One step of the forward integration algorithm looks as follows:

$$\begin{aligned} \mathbf{y}^{n+1} &= \mathbf{f}(\mathbf{y}^n, h^n, t^n) = \mathbf{y}^n + h^n \gamma\, t^n\, \mathbf{y}^n \ , \\ h^{n+1} &= \mathbf{g}(\mathbf{y}^n, h^n, t^n) = h^n \sqrt{\frac{\text{ATOL}}{\left| \exp\left(\frac{\gamma}{2}\left[(t^n + h^n)^2 - (t^n)^2\right]\right) \right| \mathbf{y}^n}} \ , \\ t^{n+1} &= t^n + h^n \ . \end{aligned} \tag{4.109}$$

We evolve the solution with the forward Euler method. The time step estimator controls the absolute local error based on the exact solution, with ATOL $= 10^{-3}$. We wrote the code using in double precision FORTRAN, and we generated the tangent linear and discrete adjoint models with the help of TAMC [58].

Tables 4.5 and 4.6 show the numerical errors generated by the spurious forward and adjoint derivatives. While small in absolute value for this example, they may grow in time and impact the overall accuracy of the numerical solutions. This may happen even when the final time step is adjusted, if (4.100) is $\mathcal{O}(1)$ for a given numerical method. The time step adjustment partially corrects for the spurious factors in (4.98c), and (4.104c). Also, the post processing technique derived in [4] improves the accuracy of the tangent linear and adjoint solutions. Still, both approaches do not eliminate all the AD artifacts, and are therefore not equivalent to a full controller "passivation": $\widetilde{\boldsymbol{\delta}\mathbf{y}^3} \neq \boldsymbol{\delta}\mathbf{y}_c^3$, $\widetilde{\boldsymbol{\delta}\mathbf{y}^3} \neq \widehat{\delta\mathbf{y}^3}$, and $\widetilde{\boldsymbol{\lambda}^0} \neq \widehat{\boldsymbol{\lambda}^0}$.

We note that $\widehat{\boldsymbol{\delta}\mathbf{y}^3} = \widehat{\lambda^0}$, $\boldsymbol{\delta}\mathbf{y}^3 = \boldsymbol{\lambda}^0$, and $\widetilde{\boldsymbol{\delta}\mathbf{y}^3} = \widetilde{\boldsymbol{\lambda}^0}$ (up to machine roundoff). This is expected from the set up for our problem, and is also due to the transpose relationship that holds between the discrete tangent linear and adjoint models (compare, for example (4.92) and (4.101a)).

Table 4.6: Errors in the discrete adjoint solutions for the final value problem (4.107) generated by the nonphysical time step adjoints. Here, $\boldsymbol{\lambda}_c^0$ is the post-processed discrete adjoint at t^0. $\widetilde{\boldsymbol{\lambda}^0}$ and $\widehat{\boldsymbol{\lambda}^0}$ are defined in (4.104a) and (4.104b), respectively. The adjoint correction (4.78) leads to an accurate discrete adjoint. Adjusting h^2 to hit t^3 exactly improves the quality of $\widehat{\boldsymbol{\lambda}^0}$ over the default solution $\boldsymbol{\lambda}^0$ for this problem, but not all spurious derivatives are eliminated, as shown in (4.104b). Hence, in general, $\widetilde{\boldsymbol{\lambda}^0} \neq \widehat{\boldsymbol{\lambda}^0}$.

	$\left\| \widetilde{\boldsymbol{\lambda}^0} - \boldsymbol{\lambda}^0 \right\|$		1.61063×10^{-4}
	$\left\| \widetilde{\boldsymbol{\lambda}^0} - \widehat{\boldsymbol{\lambda}^0} \right\|$		4.37644×10^{-6}
	$\left\| \widetilde{\boldsymbol{\lambda}^0} - \boldsymbol{\lambda}_c^0 \right\|$		0

4.2.6 Discrete second order adjoints

It is known that second order adjoints can be computed either by two reverse-mode differentiations (adjoint-over-adjoint) or, more efficiently, through a forward mode differentiation of the original model's adjoint (forward-over-adjoint) [23, 69]. Thus, it is natural to ask if we can couple the adjoint corrections (4.78) or (4.89) with the tangent linear code modifications described in [4]. We will show that this approach leads to accurate discrete second order adjoints of adaptive numerical integrators.

We will work within the general framework described in section 4.2.3. The second order discrete adjoint system allows one to obtain second order derivative information for the cost function (4.58), in the form of Hessian-vector products

$$\frac{\partial^2 \widehat{\mathcal{J}}}{(\partial \mathbf{y}^0)^2} \cdot \boldsymbol{\delta}\mathbf{y}^0 \,, \tag{4.110}$$

with $\boldsymbol{\delta}\mathbf{y}^0 \in \mathbb{R}^N$ an arbitrary vector. We henceforth mark all discrete second-order adjoint variables by an upper dot. Thus, the second order adjoint state vector at t^n has the following structure (compare with 4.74):

$$\dot{\boldsymbol{\lambda}} = \begin{bmatrix} \dot{\boldsymbol{\lambda}} \\ \dot{\mu} \\ \dot{\nu} \end{bmatrix} . \tag{4.111}$$

The discrete second order adjoint consistent with the ODE (4.65) reads:

$$\begin{aligned} \dot{\boldsymbol{\lambda}}^n &= \left(\left. \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \right|_{(\mathbf{y}^n, h^n, t^n)} \right)^T \dot{\boldsymbol{\lambda}}^{n+1} + \left(\left. \frac{\partial^2 \mathbf{f}}{\partial \mathbf{y}^2} \right|_{(\mathbf{y}^n, h^n, t^n)} \otimes \boldsymbol{\delta}\mathbf{y}^n \right)^T \boldsymbol{\lambda}^{n+1} \quad , \quad M-1 \geq n \geq 0 \\ \dot{\boldsymbol{\lambda}}^M &= \frac{\partial^2 \widehat{\mathcal{J}}}{(\partial \mathbf{y}^M)^2} \cdot \boldsymbol{\delta}\mathbf{y}^M \,. \end{aligned} \tag{4.112}$$

In (4.112) $\boldsymbol{\lambda}^n$ satisfies the first order adjoint equation (4.70), and $\boldsymbol{\delta}\mathbf{y}^n$ is solution of the discrete TLM model:

$$\boldsymbol{\delta}\mathbf{y}^n = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\,\boldsymbol{\delta}\mathbf{y}^{n-1}\,, \qquad 1 \le n \le M\,. \tag{4.113}$$

Note that the initial tangent linear model state is set to δy^0. Solving (4.112) yields [72]:

$$\dot{\boldsymbol{\lambda}}^0 = \frac{\partial^2 \widehat{\mathcal{J}}}{(\partial \mathbf{y}^0)^2} \cdot \boldsymbol{\delta}\mathbf{y}^0\,. \tag{4.114}$$

We now investigate the second order adjoint model of (4.68a – 4.68c) given by two successive invocations of AD. For efficiency [69], we take the forward-over-adjoint path, i.e., we differentiate (4.76a) – (4.76c) in the direction

$$\boldsymbol{\delta}\mathbf{v}^0 = \begin{bmatrix} \boldsymbol{\delta}\mathbf{y}^0 \\ \delta h^0 \\ \delta t^0 \end{bmatrix}^T\,. \tag{4.115}$$

Hence, the second order adjoint model generated by AD has the following structure:

$$\begin{aligned}
\dot{\boldsymbol{\lambda}}^n &= \left(\frac{\partial^2 \mathbf{f}}{\partial \mathbf{y}^2} \otimes \delta\mathbf{y}^n\right)^T \boldsymbol{\lambda}^{n+1} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^T \dot{\boldsymbol{\lambda}}^{n+1} \\
&\quad + \delta h^n \left(\frac{\partial^2 \mathbf{f}}{\partial \mathbf{y}\,\partial h}\right)^T \boldsymbol{\lambda}^{n+1} + \delta t^n \left(\frac{\partial^2 \mathbf{f}}{\partial \mathbf{y}\,\partial t}\right)^T \boldsymbol{\lambda}^{n+1} + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^T \dot{\mu}^{n+1} \\
&\quad + \mu^{n+1} \left(\frac{\partial^2 \mathbf{g}}{\partial \mathbf{y}^2}\right)^T \boldsymbol{\delta}\mathbf{y}^n + \mu^{n+1} \left(\frac{\partial^2 \mathbf{g}}{\partial \mathbf{y}\,\partial h}\right)^T \boldsymbol{\delta} h^n + \mu^{n+1} \left(\frac{\partial^2 \mathbf{g}}{\partial \mathbf{y}\,\partial t}\right)^T \boldsymbol{\delta} t^n
\end{aligned} \tag{4.116a}$$

$$\begin{aligned}
\dot{\mu}^n &= (\boldsymbol{\delta}\mathbf{y}^n)^T \left(\frac{\partial^2 \mathbf{f}}{\partial h\,\partial \mathbf{y}}\right)^T \boldsymbol{\lambda}^{n+1} + \delta h^n \left(\frac{\partial^2 \mathbf{f}}{\partial h^2}\right)^T \boldsymbol{\lambda}^{n+1} + \delta t^n \left(\frac{\partial^2 \mathbf{f}}{\partial h\,\partial t}\right)^T \boldsymbol{\lambda}^{n+1} \\
&\quad + \mu^{n+1} \frac{\partial^2 \mathbf{g}}{\partial h\,\partial \mathbf{y}}\,(\boldsymbol{\delta}\mathbf{y}^n)^T + \boldsymbol{\delta} h^n \frac{\partial^2 \mathbf{g}}{\partial h^2}\,\mu^{n+1} + \boldsymbol{\delta} t^n \frac{\partial^2 \mathbf{g}}{\partial h\,\partial t}\,\mu^{n+1} \\
&\quad + \left(\frac{\partial \mathbf{f}}{\partial h}\right)^T \dot{\boldsymbol{\lambda}}^{n+1} + \frac{\partial \mathbf{g}}{\partial h}\,\dot{\mu}^{n+1} + \dot{\nu}^{n+1}
\end{aligned} \tag{4.116b}$$

$$\begin{aligned}
\dot{\nu}^n &= (\boldsymbol{\delta}\mathbf{y}^n)^T \left(\frac{\partial^2 \mathbf{f}}{\partial t\,\partial \mathbf{y}}\right)^T \boldsymbol{\lambda}^{n+1} + \delta h^n \left(\frac{\partial^2 \mathbf{f}}{\partial t\,\partial h}\right)^T \boldsymbol{\lambda}^{n+1} + \delta t^n \left(\frac{\partial^2 \mathbf{f}}{\partial t^2}\right)^T \boldsymbol{\lambda}^{n+1} \\
&\quad + (\boldsymbol{\delta}\mathbf{y}^n)^T \frac{\partial^2 \mathbf{g}}{\partial t\,\partial \mathbf{y}}\,\mu^{n+1} + \boldsymbol{\delta} h^n \frac{\partial^2 \mathbf{g}}{\partial t\,\partial h}\,\mu^{n+1} + \boldsymbol{\delta} t^n \frac{\partial^2 \mathbf{g}}{\partial t^2}\,\mu^{n+1} \\
&\quad + \frac{\partial \mathbf{f}}{\partial t}\,\dot{\lambda}^{n+1} + \frac{\partial \mathbf{g}}{\partial t}\,\dot{\mu}^{n+1} + \dot{\nu}^{n+1}\,, \qquad M-1 \ge n \ge 0\,.
\end{aligned} \tag{4.116c}$$

The update (4.116) is not identical to (4.112): several spurious terms are added at each time step. Also, note that δy^n gets updated by the AD-generated TLM model (4.73)

$$\boldsymbol{\delta}\mathbf{y}^n = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\boldsymbol{\delta}\mathbf{y}^{n-1} + \frac{\partial \mathbf{f}}{\partial h}\boldsymbol{\delta} h^{n-1} + \frac{\partial \mathbf{f}}{\partial t}\delta t^{n-1}\,, \qquad 1 \le n \le M\,, \tag{4.117}$$

instead of (4.61). This mismatch is another source of errors in the second order discrete adjoint solution $\dot{\boldsymbol{\lambda}}^n$, since the second order adjoint computations need to make use of an accurate tangent linear model trajectory.

To cancel out all the AD-induced perturbations in the discrete second order adjoint, one should follow this two-step procedure:

1. restore the TLM solution $\boldsymbol{\delta}\mathbf{y}^n$ to the value given by (4.113). This can be done by zeroing out the spurious tangent derivatives in the TLM code:

$$\begin{aligned} \boldsymbol{\delta}h^n &\leftarrow 0 \\ \boldsymbol{\delta}t^n &\leftarrow 0 \end{aligned} \tag{4.118}$$

   before the update (4.116). Alternatively, one can apply the post-processing strategy described in [4] at each time step. Both of these approaches will result in a TLM solution accurate to order p (under suitable smoothness assumptions on the solution $\boldsymbol{\delta}\mathbf{y}(t)$). Note that (4.118) implies that the forward differentiation of (4.76a) – (4.76c) is now performed in the direction of $\boldsymbol{\delta}v^0 = \left[(\boldsymbol{\delta}\mathbf{y}^0)^T \quad 0 \quad 0\right]^T$.

2. Next, in the second order adjoint code, zero out the non-physical first and second order adjoint derivatives of the time step:

$$\begin{aligned} \mu^{n+1} &\leftarrow 0 \\ \dot{\mu}^{n+1} &\leftarrow 0\,. \end{aligned} \tag{4.119}$$

After (4.118) and (4.119), the second order adjoint trajectory is restored to the value given by (4.112). For completeness, we remark that an approach equivalent to (4.119) is to implement (4.78), and then post-process [4] the second order adjoint solution.

It is important to note that any correction to the TLM or first order adjoint variables, such as (4.118) or (4.119), should be made *after* the second (forward) differentiation. If (4.118) is introduced before the second order adjoint code is generated, then it may influence the behavior of the AD engine and result in unusable code.

4.2.7 Numerical experiments

All numerical tests were performed on an Intel Pentium 4 Workstation running Fedora Core Linux, with the Runge–Kutta routines coded in double precision Fortran 90.

We used the 5th order DOPRI5(4) Runge–Kutta method [134] with variable time stepping in all numerical experiments. DOPRI5(4) uses the stage values to compute two solution approximations of different accuracies at every time step. The first of these two numerical solutions is used to advance the integrator state, and the other (less accurate) solution serves

to control the local error and the time step size. This approach significantly lowers the cost of step rejections. The time step controller is based on the following formula [124]:

$$h^{n+1} = h^n \cdot \min\left\{5\,,\,\max\left\{0.2\,,\,0.9\,\|\mathbf{e}^n\|^{-1/5}\right\}\right\}. \tag{4.120}$$

Here $\|\mathbf{e}^n\|$ is an estimate for the weighted norm of the local error at step n. This estimate is computed based on the relative (RTOL) and absolute (ATOL) integration tolerances:

$$\|\mathbf{e}^n\| = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\mathbf{y}^n_{err}(i)}{\mathrm{ATOL(i)} + \mathrm{RTOL(i)}\cdot|\mathbf{y}^n_{\mathrm{i}}|}\right)^2}\,, \tag{4.121}$$

with $\mathbf{y}_{\mathrm{err}}$ approximating the local error of the Runge–Kutta method. We do not adjust the final integration step. The reference solutions were obtained with MATLAB's `ode45` [144]. Here $\mathrm{ATOL} = 10^{-13}, \mathrm{RTOL} = 10^{-12}$.

First order adjoint sensitivity analysis

We first investigate the discrete adjoint of the Prothero–Robinson IVP [145]:

$$\begin{aligned} \dot{\mathbf{y}} &= \gamma\left(\mathbf{y} - \phi(t)\right) + \dot{\phi}(t)\,, \\ t^0 &= 0 \le t \le t^F \approx 2\,, \\ \mathbf{y}(t^0) &= \begin{bmatrix}0.5 & 0.5\end{bmatrix}^T\,, \end{aligned} \tag{4.122}$$

with $\gamma = -5$, and $\phi(t) = [\sin t \quad \cos t]^T$.

We choose:

$$\mathcal{J}(\mathbf{y}) = \mathbf{y}_1(t^F)\,. \tag{4.123}$$

Hence, the continuous adjoint of (4.122) reads:

$$\begin{aligned} \dot{\boldsymbol{\lambda}} &= \gamma\boldsymbol{\lambda}\quad,\quad t^F \ge t \ge t^0\,, \\ \boldsymbol{\lambda}(t^F) &= \begin{bmatrix}1 & 0\end{bmatrix}^T\,. \end{aligned} \tag{4.124}$$

We differentiate our Runge–Kutta implementation using the reverse mode of TAMC. Figure 4.7 shows the two discrete adjoint solution components and their root-mean-square (RMS) errors, both before ($\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$) and after ($\boldsymbol{\lambda}^c_1$, $\boldsymbol{\lambda}^c_2$) the adjoint correction (4.78) is applied. The RMS errors are computed using the formula:

$$\epsilon_{\mathrm{RMS}} = \sqrt{\frac{1}{M}\sum_{n=0}^{M}\left(\frac{|\boldsymbol{\lambda}^{\mathrm{ref}}(t^n) - \boldsymbol{\lambda}^n|}{\max\left\{|\boldsymbol{\lambda}^{\mathrm{ref}}(t^n)|\,,|\boldsymbol{\lambda}^n|\,,\mathrm{TOL}\right\}}\right)^2}\,, \tag{4.125}$$

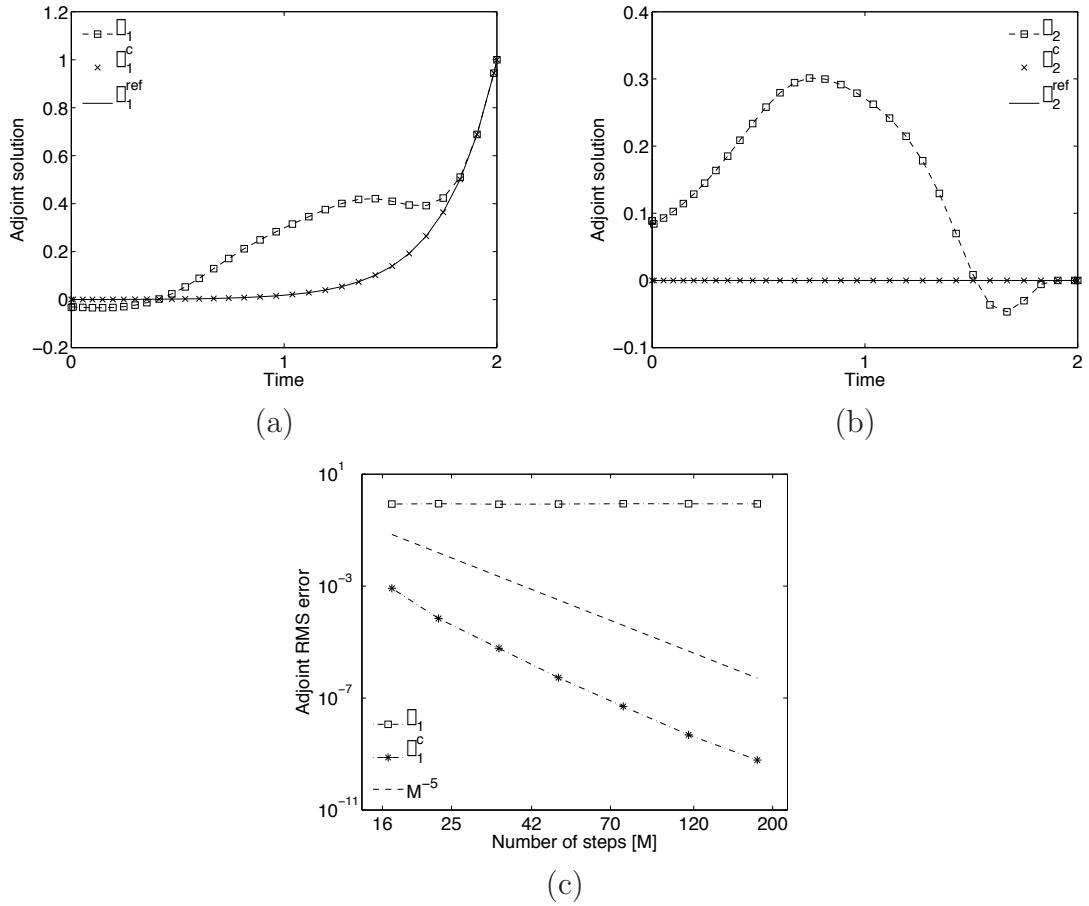


Figure 4.7: (a–b) Discrete adjoint trajectories ($ATOL = RTOL = 10^{-7}$), and (c) RMS errors for the system (4.122). The AD adjoints $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, and the corrected solutions $\boldsymbol{\lambda}_1^c$, $\boldsymbol{\lambda}_2^c$ were computed with DOPRI5(4). The reference solution $\boldsymbol{\lambda}^{\text{ref}}$ was obtained through a backward time integration of the continuous adjoint (4.124), using MATLAB's `ode45` function. It is clear from (c) that the AD discrete adjoint of the DOPRI method is inconsistent with the continuous system (4.124). After the correction, the adjoint solution and the reference adjoint trajectory are visually indistinguishable. As seen in (c), post-processing restores the discrete adjoint solution to full accuracy.

with TOL $\approx 10^{-12}$. One can see in Figure 4.7(a–b) that black-box invocation of AD results in very large errors in both components of the discrete adjoint trajectory. This is consistent with the behavior predicted by the mathematical derivations (4.87). The adjoint solution is completely wrong. After the correction is applied, the accuracy of the discrete adjoint solution matches that of the underlying Runge–Kutta method, as Figure 4.7(c) illustrates.

Note that the two adjoint corrections (4.77) and (4.78) yield the same adjoint solutions (up to machine roundoff). Figure 4.7 only shows the numerical results obtained with (4.77).

Second order adjoint sensitivity analysis

For second order sensitivity analysis, we introduce a nonlinear term in the right-hand side of (4.122):

$$\begin{aligned} \dot{\mathbf{y}}_1 &= \gamma\,(\mathbf{y}_1 - \sin t) + \mathbf{y}_2^3 \cos t\,, \\ \dot{\mathbf{y}}_2 &= \gamma\,(\mathbf{y}_2 - \cos t) - \mathbf{y}_1^3 \sin t\,, \\ t^0 &= 0 \le t \le t^F = 2\,, \\ \mathbf{y}(t^0) &= [0.5 \quad 0.5]^T\,. \end{aligned} \tag{4.126}$$

Let the cost function $\mathcal{J}$ be defined by (4.123), and

$$\delta\mathbf{y}^0 = [1 \quad 0]^T \tag{4.127}$$

in (4.110). Then, the first-order adjoint system for (4.126) has the form

$$\begin{aligned} \dot{\boldsymbol{\lambda}}_1 &= -\gamma\boldsymbol{\lambda}_1 + 3y_1^2\boldsymbol{\lambda}_2 \sin t \\ \dot{\boldsymbol{\lambda}}_2 &= -3y_2^2\boldsymbol{\lambda}_1 \cos t - \gamma\boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}(t^F) &= [1 \quad 0]^T\,, \end{aligned} \tag{4.128}$$

whereas the second order adjoint model reads:

$$\begin{aligned} \dot{\boldsymbol{\sigma}}_1 &= -\gamma\boldsymbol{\sigma}_1 + 3\mathbf{y}_1^2\boldsymbol{\sigma}_2 \sin t + 6\mathbf{y}_1\boldsymbol{\delta}\mathbf{y}_1\boldsymbol{\lambda}_2 \sin t \\ \dot{\boldsymbol{\sigma}}_2 &= -3\mathbf{y}_2^2\boldsymbol{\sigma}_1 \cos t - \gamma\boldsymbol{\sigma}_2 - 6\mathbf{y}_2\boldsymbol{\delta}\mathbf{y}_2\boldsymbol{\lambda}_1 \cos t \\ \boldsymbol{\sigma}_1(t^F) &= \boldsymbol{\sigma}_2(t^F) = 0\,. \end{aligned} \tag{4.129}$$

Here $\boldsymbol{\sigma}(t)$ denotes the second order adjoint variable, and $\boldsymbol{\delta}\mathbf{y}$ is the solution of the tangent linear model:

$$\begin{aligned} \boldsymbol{\delta}\dot{\mathbf{y}}_1 &= \gamma\boldsymbol{\delta}\mathbf{y}_1 + 3y_2^2\boldsymbol{\delta}\mathbf{y}_2 \cos t \\ \boldsymbol{\delta}\dot{\mathbf{y}}_2 &= -3\mathbf{y}_1^2\boldsymbol{\delta}\mathbf{y}_1 \sin t + \gamma\boldsymbol{\delta}\mathbf{y}_2 \\ \boldsymbol{\delta}\mathbf{y}(t^0) &= \boldsymbol{\delta}\mathbf{y}^0\,. \end{aligned} \tag{4.130}$$

We build the second order adjoint of our DOPRI5(4) implementation through forward over reverse differentiation. The results are shown in Figure 4.8. As expected from (4.116), the second order adjoint of the DOPRI method is inconsistent with its continuous counterpart (4.129). The reason for this is twofold: the errors in the second order adjoint variables are generated both by the perturbations present in the first order adjoint solution $\boldsymbol{\lambda}$ (see 4.87), and the spurious derivatives generated during the second (forward) differentiation [4]. Figure 4.8(a–b) contrasts the discrete solutions before – $\boldsymbol{\lambda}_1$,$\boldsymbol{\lambda}_2$ – and after – $\boldsymbol{\lambda}_1^c$, $\boldsymbol{\lambda}_2^c$ – the post-processing (4.118 – 4.119), for ATOL = RTOL = 10^{-7}. The corrected solution is visually indistinguishable from the reference $\boldsymbol{\lambda}^{\text{ref}}$, whereas the AD-computed adjoint is several orders of magnitude away from the true solution. Finally, Figure 4.8 shows the order of accuracy of the post-processed discrete adjoint, which matches that of the DOPRI pair used in the forward model integration.

Figure 4.8 only shows the solutions obtained after zeroing out all spurious forward and adjoint time step derivatives. Applying (4.89) gives virtually identical results.

4.2.8 Conclusions

In this section we investigate the behavior of adaptive numerical integration algorithms under the reverse mode of automatic differentiation. To maximize efficiency and reduce computation time, such algorithms rely on time step controllers and local error estimators to keep the solution accuracy within user-specified bounds. The discrete adjoints of such integrators can be automatically generated using automatic differentiation. If the time step controllers and error estimators are not differentiated, adjoints of explicit Runge–Kutta methods remain consistent with the corresponding continuous equations, as shown in [95, 122]. However, isolating parts of the integrator code, in order to hide them from the AD engine, can be far from trivial for legacy or industrial-scale numerical codes.

If the controllers or the error estimators are differentiated, the AD mechanism will pick up the dependencies between the time steps taken by the forward method and the model solution. This results in spurious adjoint time and time step gradients. These non-physical derivatives influence the discrete adjoint trajectory at every time step. We show that the perturbations add up and generate a $\mathcal{O}(1)$ error in the final solution. Thus, using the discrete adjoint code as-is will yield incorrect gradients. An analysis of the adjoint of a general one-step explicit adaptive integration scheme, reveals that simple code modifications lead to a discrete adjoint solution that has the same order of accuracy as the underlying numerical method.

As a special case, we discuss the case when the last time step is adjusted to hit a prescribed final integration time. This is a common feature of adaptive numerical integration codes. The time step adjustment leads to a significant improvement in accuracy of the discrete adjoint solution over the discrete adjoint computed with the original code (which has a $\mathcal{O}(1)$ error). This is due to the result in [4], and the transpose relationship between the discrete tangent linear and adjoint models [143]. However, we show that not all the spurious derivatives

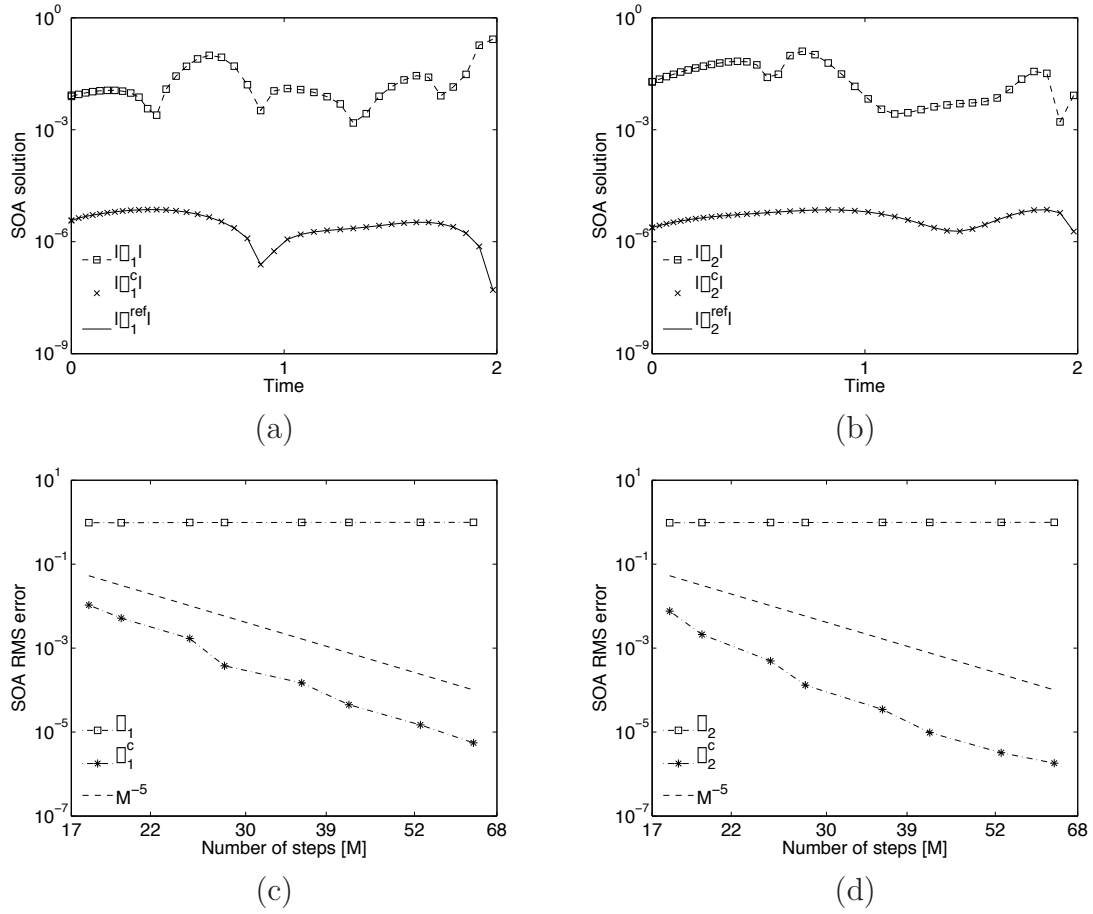


Figure 4.8: Discrete second order adjoints (a–b) and RMS errors (c–d) for the IVP (4.126). The AD discrete second order adjoint $\boldsymbol{\sigma}_{1,2}$ differs from the reference trajectory $\boldsymbol{\sigma}^{\text{ref}}$ by several orders of magnitude. However, the corrected solution $\boldsymbol{\sigma}^{c}_{1,2}$ is visually indistinguishable from $\boldsymbol{\sigma}^{\text{ref}}$ (ATOL = RTOL = 10^{-7}). Also, the decrease in the RMS errors of the corrected trajectory (c–d) confirms that canceling the spurious adjoint derivatives yields a 5th order accurate second order adjoint solution.

from the tangent linear or adjoint solutions are eliminated. The surviving perturbations may impact the accuracy of the numerical sensitivities, leading to an artificial drop in the order of accuracy of the numerical integration scheme. Hence, an adjoint correction is still necessary. The mathematical derivations are validated on a three-step numerical integration problem.

We also consider second-order adjoints of adaptive integrators, obtained through forward-over-adjoint differentiation. The analysis shows that it is possible to obtain accurate second order derivative information through straightforward post-processing of the second order adjoint code.

Two examples that use Dormand and Prince's DOPRI5(4) Runge–Kutta pair with adaptive error control are given. The numerical results fully support the mathematical derivations in both cases. The AD-generated Runge–Kutta adjoints are inconsistent with the continuous adjoint ODEs. However, once the adjoint corrections are applied, the first and second order discrete adjoint solutions have the same order of accuracy as the underlying Runge–Kutta algorithm.

Chapter 5

Space adaptivity

Introduction We demonstrate the feasibility and efficiency of the discrete adjoint method for adaptive time-dependent inverse problems. While the efforts of other authors have considered almost exclusively on steady-state problems (with some exceptions, see [91, 66]), we examine discrete adjoints for evolution problems, and highlight the benefits of both time and space adaptivity. Recent advances in adjoint computation strategies have made reversal of time-dependent codes computationally feasible (see [65], and references therein). The main computational advantage of the discrete adjoint approach is that the adjoint model code can be obtained through automatic differentiation, hence saving a significant amount of software development effort. Automatic differentiation tools are available for all the major programming languages used in scientific computing [58, 54, 57, 56, 60]. Specialized finite element software such as DEAL.II allow the mesh adaptation to be performed in a transparent fashion, and independent from the numerical core of the solution algorithm. Thus, automatic differentiation can be used in a targeted fashion, such that we obtain the adjoint of only the time marching procedure, and that of the (linearized) right hand side of the discrete model equations. This simplifies both the derivation and verification of the discrete adjoint model.

We first present the general form of the discrete adjoint method, and then discuss the issues that arise with adaptive mesh and time refinement. For spectral-type numerical methods such as discontinuous Galerkin (a method particularly amenable to space-time adaptivity), the discrete intergrid operators are implemented through orthogonal projections. Thus, there are no inconsistencies introduced through the use of the adjoint interpolation and restriction operators obtained by automatic differentiation. The analysis is then extended to general meshes, where the same operator properties are verified. However, this orthogonality property is not recovered for all numerical methods. The adjoint analysis of finite volume mesh transfer operators shows the adjoint of high-order interpolation (through solution averaging) to be only first order accurate in the general case.

We then focus on the concept of adjoint consistency for time-dependent discontinuous Galerkin discretizations. The concept of adjoint consistency, defined, e.g., in [28, 32] for

elliptic problems, plays an important role in the analysis of the dual (adjoint) problem solution, in the convergence of the primal approximation, as well as in the accuracy of the target functional under consideration. Building on previous duality results for time [33, 122, 95] and space discretizations [28, 146], we develop an unified framework for investigating dual consistency of discretizations for a general type of time-dependent PDEs.

This chapter is structured as follows. Section 5.1 discusses the adaptive inverse problem framework and the difficulties associated with adaptivity and the discrete adjoint method. In section 5.2 we review the mathematical foundation of the discontinuous Galerkin (DG) method. Section 5.3 concerns intergrid transfer operators for Galerkin-type discretizations. We remark that both h- and p-refinement with structured AMR are performed through orthogonal projections, hence the accuracy of the discrete adjoint solution will not be affected by intergrid solution transfer operators (beyond the intrinsic loss in accuracy associated with mesh coarsening). Section 5.3.4 discusses the interpolation and restriction operators for the h-adaptive finite volume method. Subsequently, section 5.4 discusses in detail the derivation of formal adjoint systems for general differential problems that obey a given set of compatibility conditions. Section 5.5 considers the dual consistency of time discretizations in Runge-Kutta DG methods. The accuracy and computational advantages of the discrete adjoint approach are demonstrated on a two-dimensional numerical test problem in section 5.5.3.

5.1 The adaptive inverse problem framework

5.1.1 Model problems

We use the strong formulation of the model problems given in the introduction, i.e., equations (1.13), and (1.13). For convenience, we restate them here.

The continuous inverse problem

$$\text{Find } \mathbf{q} = \arg \min_{\mathbf{u}\in\mathcal{U},\mathbf{q}\in\mathcal{Q}} \mathcal{J}(\mathbf{u}, \mathbf{q}) \text{ , subject to (1.13) .} \tag{5.1}$$

The discrete inverse problem

$$\text{Find } \mathbf{q}^h = \arg \min_{\mathbf{u}^h\in\mathcal{U}_h,\mathbf{q}^h\in\mathcal{Q}_h} \mathcal{J}_h(\mathbf{u}^{h,0:N}, \mathbf{q}^h) \text{ , subject to (1.13) .} \tag{5.2}$$

5.1.2 Derivation of the discrete adjoint equations

Consider the steepest descent method [20] applied to minimize the cost functional $\mathcal{J}^h$. The solution update has the following form:

$$\mathbf{q}^h_{\text{new}} = \mathbf{q}^h_{\text{old}} - \alpha \left(\frac{d\mathcal{J}^h}{d\mathbf{q}^h} \right)^T ,$$

where α is a suitably chosen step length, and the reduced gradient reads:

$$\frac{d\mathcal{J}^h}{d\mathbf{q}^h} = \frac{\partial \mathcal{J}_h}{\partial \mathbf{q}^h} + \sum_{n=1}^{N} \frac{\partial \mathcal{J}_h}{\partial \mathbf{u}^{h,n}} \frac{\partial \mathbf{u}^{h,n}}{\partial \mathbf{q}^h} . \tag{5.3}$$

Since equation (1.13) is discretized in residual form, the implicit function theorem gives the following equation, also called the *tangent linear model* (henceforth referred to as the TLM):

$$\frac{\partial \mathcal{F}^{h,n}}{\partial \mathbf{u}^{h,n}} \frac{\partial \mathbf{u}^{h,n}}{\partial \mathbf{q}^h} + \frac{\partial \mathcal{F}^{h,n}}{\partial \mathbf{u}^{h,n-1}} \frac{\partial \mathbf{u}^{h,n-1}}{\partial \mathbf{q}^h} = -\frac{\partial \mathcal{F}^{h,n}}{\partial \mathbf{q}^h} , \;\; n = 1 \ldots N . \tag{5.4}$$

Hence, the space-time matrix formulation of the sensitivity equations reads as follows:

$$M \begin{bmatrix} \dfrac{\partial \mathbf{u}^{h,N}}{\partial \mathbf{q}^h} \\ \dfrac{\partial \mathbf{u}^{h,N-1}}{\partial \mathbf{q}^h} \\ \vdots \\ \dfrac{\partial \mathbf{u}^{h,1}}{\partial \mathbf{q}^h} \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial \mathcal{F}^{h,N}}{\partial \mathbf{q}^h} \\ -\dfrac{\partial \mathcal{F}^{h,N-1}}{\partial \mathbf{q}^h} \\ \vdots \\ -\dfrac{\partial \mathcal{F}^{h,0}}{\partial \mathbf{q}^h} \end{bmatrix} , \tag{5.5}$$

where

$$M := \begin{bmatrix} \dfrac{\partial \mathcal{F}^{h,N}}{\partial \mathbf{u}^{h,N}} & \dfrac{\partial \mathcal{F}^{h,N}}{\partial \mathbf{u}^{h,N-1}} & 0 & \ldots & 0 \\ 0 & \dfrac{\partial \mathcal{F}^{h,N-1}}{\partial \mathbf{u}^{h,N-1}} & \dfrac{\partial \mathcal{F}^{h,N-1}}{\partial \mathbf{u}^{h,N-2}} & 0 & \ldots \\ 0 & 0 & \ddots & \ddots & \ldots \\ 0 & \ldots & \ldots & 0 & \dfrac{\partial \mathcal{F}^{h,0}}{\partial \mathbf{u}^{h,1}} \end{bmatrix} .$$

The sensitivity matrices $\partial \mathbf{u}^{h,n} / \partial \mathbf{q}^h$ are very expensive to compute, since they scale with both the number of states and that of the control variables. When a new control variable is added, (5.4) needs to be solved anew. The *discrete adjoint method* [27] calculates the gradient (5.3) at a significantly lower cost than finite differences or forward sensitivities when the number

of parameters is large compared to the number of outputs of interest. By defining the N discrete adjoint variables $\boldsymbol{\lambda}^{h,n}$ as the solution components of the *discrete adjoint equation*, which reads:

$$M^T \begin{bmatrix} \boldsymbol{\lambda}^{h,N} \\ \boldsymbol{\lambda}^{h,N-1} \\ \vdots \\ \boldsymbol{\lambda}^{h,1} \end{bmatrix} = \begin{bmatrix} \left(\dfrac{\partial \mathcal{J}_h}{\partial \mathbf{u}^{h,N}}\right)^T \\ \left(\dfrac{\partial \mathcal{J}_h}{\partial \mathbf{u}^{h,N-1}}\right)^T \\ \vdots \\ \left(\dfrac{\partial \mathcal{J}_h}{\partial \mathbf{u}^{h,1}}\right)^T \end{bmatrix}, \tag{5.6}$$

we obtain

$$\frac{d\mathcal{J}^h}{d\mathbf{q}^h} = \frac{\partial \mathcal{J}^h}{\partial \mathbf{q}^h} - \sum_{n=1}^{N} \left(\boldsymbol{\lambda}^{h,n}\right)^T \frac{\partial \mathcal{F}^n}{\partial \mathbf{q}^h} \,. \tag{5.7}$$

Note that both the adjoint matrix M^T, and the right hand side of (5.6), are independent of the number of inversion variables. If the size of $\mathbf{q}^h$ does not scale directly with the model state size $\mathbf{u}^h$, the cost of the discrete adjoint approach does not depend on the number of inversion variables. From the block lower bidiagonal structure of (5.6), we remark that the adjoint equations are solved backwards in time, i.e. from $n = N$ to $n = 0$. The size of the blocks may vary with n because of the mesh adaptation mechanism. This change in local solution size is easily accommodated by standard single-step Runge-Kutta-type ODE solvers such as the ones employed in this chapter. Should a s-stage Runge-Kutta be used, the forward and adjoint systems (5.5)–(5.6) will have s non-zero block diagonals, and the computational cost scales accordingly. We do not consider linear multistep methods [131] in this chapter, since they are not generally adjoint consistent [96].

5.1.3 Computational advantages of the discrete adjoint method

One frequently raised objection to the discrete adjoint method (besides consistency issues with the numerical discretization) is that the forward mesh is frequently sub-optimal for the adjoint problem. While independent mesh refinement for the adjoint problem would enhance both the accuracy and the efficiency of the discrete adjoint solver, the additional complexity may increase the overall cost of the inversion process. The development effort required for the discretization of the continuous adjoint equations becomes significantly larger (there is no possibility of automating the computation). Features in the newly computed adjoint solutions may also be degraded when interpolating the gradient (5.7) to a reference optimization grid [15].

If we consider the Lagrangian interpretation of the adjoint variables, $\boldsymbol{\lambda}^{h,n}$ tracks how well the forward solution $\mathbf{u}^{h,n}$ satisfies the state equation [147]. Hence, it is a reasonable choice to

define both the $\mathbf{u}^{h,n}$ and $\boldsymbol{\lambda}^{h,n}$ on the same mesh. Note from equation (5.7) that the discrete adjoint method can easily accommodate a different mesh for the inversion variables. While the discrete partial derivatives $\dfrac{\partial \mathcal{J}^h}{\partial \mathbf{q}^h}$ and $\dfrac{\partial \mathcal{F}^{h,n}}{\partial \mathbf{q}^h}$ are now defined on a separate "parameter mesh", the adjoint system (5.6) and its solutions $\boldsymbol{\lambda}^{h,n}$ do not change. This approach was proven to be very beneficial in practice [147, 148].

5.1.4 Mesh adaptivity and the discrete adjoint method

The mesh adaptivity raises several complications for the discrete adjoint approach. The issues that arise with adaptive temporal discretizations and automatic differentiation have been discussed in [4, 97]. We focus our analysis on the mesh refinement process. The mesh at a given nonlinear iteration and/or time step is refined or coarsened based on some *a posteriori* (residual-based) error estimation for the current solution approximation. Thus, the forward solver is required to transfer the solution between different meshes. We can write the forward solution process for (1.13) as:

$$\begin{aligned} \mathbf{u}^{h,0} &= \mathbf{u}^{h,0}(\mathbf{x}^h) \,, \\ \mathbf{u}^{h,n+1} &= \mathcal{I}_{n\to n+1}\left(\mathcal{S}_{n\to n+1}\,\mathbf{u}^{h,n}\right) \;\;,\;\; 0 \le n \le N-1 \,. \end{aligned}$$

Here $\mathcal{S}_{n\to n+1}(\cdot)$ is the nonlinear solution operator that advances $\mathbf{u}^{h,n}$ in time from t^n to t^{n+1} on Ω_n^h. The linear intergrid transfer operator is $\mathcal{I}_{n\to n+1} : \Omega_n^h \to \Omega_{n+1}^h$. Hence, the discrete adjoint procedure for solving (5.6), that may be generated through automatic differentiation, reads:

$$\begin{aligned} \boldsymbol{\lambda}^{h,N} &= \boldsymbol{\lambda}^{h,N}(\mathbf{x}^h) \\ \boldsymbol{\lambda}^{h,n} &= \mathcal{S}'^{*}_{n+1\to n}\left(\mathcal{I}^T_{n\to n+1}\,\boldsymbol{\lambda}^{h,n+1}\right) \;\;,\;\; N-1 \ge n \ge 0 \,, \end{aligned}$$

where the adjoint solution operator

$$\mathcal{S}'^{*}_{n+1\to n}(\cdot) = -\left(\frac{\partial \mathcal{F}^n}{\partial \mathbf{u}^{h,n}}\right)^{-T}\left[\left(\frac{\partial \mathcal{F}^{n+1}}{\partial \mathbf{u}^{h,n}}\right)^{T}(\cdot) + \left(\frac{\partial \mathcal{J}^h}{\partial \mathbf{u}^{h,n}}\right)^{T}\right]$$

is the discrete adjoint (i.e., transpose) of the linearization of $\mathcal{S}_{n\to n+1}$. The grid transfer operator is $\left(\mathcal{I}_{n\to n+1}\right)^T : \Omega_{n+1}^h \to \Omega_n^h$. Barring consistency issues in the discrete adjoint of the spatial discretization, or in the reversal of the time integration procedure, there remains the question of the impact of the intergrid operators on the accuracy of the discrete adjoint solution. If

$$\mathcal{I}_{n+1\to n} = \mathcal{C}\cdot\left(\mathcal{I}_{n\to n+1}\right)^T \;, \tag{5.8}$$

with a constant $\mathcal{C}$ independent of mesh and time step size (a valid assumption in most multigrid implementations, see e.g., [149]), the discrete adjoint accuracy is not compromised,

and the adjoint code generated by AD can be used as-is (with a simple scaling to take into account the constant factor).

Note that the solution transfer may also be done before $\mathcal{S}_{n\to n+1}(\cdot)$ is applied:

$$\begin{aligned}
\mathbf{u}^{h,0} &= \mathbf{u}^{h,0}(\mathbf{x}^h) \,, \\
\mathbf{u}^{h,n+1} &= \mathcal{S}_{n\to n+1}\left(\mathcal{I}_{n\to n+1}\,\mathbf{u}^{h,n}\right) \;\;,\;\; 0 \le n \le N-1 \,. \\
\boldsymbol{\lambda}^{h,N} &= \boldsymbol{\lambda}^{h,N}(\mathbf{x}^h) \\
\boldsymbol{\lambda}^{h,n} &= \mathcal{I}^T_{n\to n+1}\left(\mathcal{S}'^{*}_{n+1\to n}\,\boldsymbol{\lambda}^{h,n+1}\right) \;\;,\;\; N-1 \ge n \ge 0 \,.
\end{aligned}$$

With few modifications, the analysis above remains valid.

5.1.5 Multigrid optimization with the discrete adjoint method

When the inversion variables are spatially distributed, e.g., $\mathbf{q}^h = \mathbf{u}^{h,0}$, they are represented on a given mesh, e.g., Ω_0^h. As the optimization proceeds the shape of the field $\mathbf{q}^h$ changes, and the grid may require adjustments in order to accurately represent the new $\mathbf{q}^h$. One possible solution is to apply grid refinement operations at the end of each optimization cycle. This approach has the disadvantage that the dimensions of the parameter and gradient vectors change during optimization. We do not know of any optimization algorithm that can handle a variable optimization space. To overcome this problem an alternative strategy is to define a fixed optimization mesh Θ_0^h, to project the parameter and gradient vectors from the computational mesh to the optimization one before the optimizer step, and to project the results back for the next function and gradient evaluation [150]. This approach has the disadvantage that the optimization mesh does not adapt to the changing solution profile; moreover, accuracy can be lost in the repeated interpolation process.

We propose to use a multigrid optimization approach [26]. The optimization grid is the computational grid Ω_0^H and is fixed throughout the inversion process. The optimizer need not accommodate changes in the discrete solution space size, and the code complexity is reduced as no interpolation onto a reference mesh is required. The optimization on the (coarser) Ω_0^H converges to obtain the solution $\mathbf{q}^H$. Through grid refinement operations both $\mathbf{q}^H$ and $\mathbf{u}^{H,0}$ are projected onto a finer Ω_0^h grid that allows a more accurate representation of the fields of interest. The optimization process is then restarted on Ω_0^h, using the current best solution approximation $\mathbf{q}^h = \mathcal{I}_{\Omega_0^H\to\Omega_0^h}\,\mathbf{q}^H$ as the initial iterate.

5.1.6 Dual consistency for space-time discretizations

In the following we extend the dual consistency analysis in [32, 28] to space-time discretizations. Discretization of the time dimension by a Runge–Kutta quadrature [131] introduces several complications that preclude a simple extension of the spatial dual consistency concept, defined, e.g., in [32]. Consider the following framework for sensitivity analysis:

- The continuous primal problem is defined by (1.13)–(5.1).
- The tangent linear problem, i.e. the linearization of the continuous primal formulation:

$$\begin{aligned} &\mathcal{F}'[\mathbf{u},\mathbf{q}](\delta\mathbf{u},\delta\mathbf{q}) = 0 \\ &\delta\mathcal{J} := \mathcal{J}'[\mathbf{u},\mathbf{q}](\delta\mathbf{u},\delta\mathbf{q}) \; . \end{aligned} \tag{5.9}$$

  Here the $'$ symbol denotes the Fréchet derivative of $\mathcal{F}$, while the bracket notation indicates the state about which the linearization is performed. The direction of differentiation is $(\delta\mathbf{u},\delta\mathbf{q})$, i.e. the full state vector of the tangent linear model. Note that $\mathcal{J}' = 0$ at the exact solution $(\mathbf{u})$

- The continuous $\mathcal{L}^2$-dual problem:

$$\begin{aligned} &\mathcal{F}'^{*}[\mathbf{u},\mathbf{q}](\boldsymbol{\lambda}) = 0 \\ &\mathcal{J}^a := \mathcal{J}'[\mathbf{u},\mathbf{q}](\boldsymbol{\lambda},\mathbf{q}) \; . \end{aligned} \tag{5.10}$$

  The * superscript denotes an adjoint operator. Also, $\boldsymbol{\lambda}$ is the continuous dual variable, and $\mathcal{J}^a = \delta\mathcal{J}$ is the expression of Fréchet derivative of $\mathcal{J}$ in terms of the dual variable.

- The discrete primal problem (1.13)–(5.2).
- The linearization of the discrete primal, i.e., the discrete tangent linear model

$$\begin{aligned} &\left(\mathcal{F}^{h,n}\right)'[\mathbf{u}^{h,n},\mathbf{q}^h](\delta\mathbf{u}^{h,n},\,\delta\mathbf{q}^h) = 0 \\ &\delta\mathcal{J}^h := \left(\mathcal{J}^h\right)'[\mathbf{u}^{h,n},\mathbf{q}^h](\delta\mathbf{u}^{h,n},\delta\mathbf{q}^h) \; . \end{aligned} \tag{5.11}$$

  This is obtained directly by Fréchet differentiation of the discrete primal formulation, in the direction $(\delta\mathbf{u}^h,\delta\mathbf{q}^h)$.

- The discrete dual problem, obtained, e.g., through automatic differentiation, directly from the discrete tangent linear model (5.11):

$$\begin{aligned} &\left(\mathcal{F}^{h,n}\right)'^{*}[\mathbf{u}^{h,n},\mathbf{q}^h](\boldsymbol{\lambda}^{h,n}) = 0 \\ &\mathcal{J}_a^h := \left(\mathcal{J}^h\right)'^{*}[\mathbf{u}^{h,n},\mathbf{q}^h](\boldsymbol{\lambda}^{h,0:N},\mathbf{q}^h) \; . \end{aligned} \tag{5.12}$$

  In the discrete space formulation (5.12), the adjoint operator is equivalent to a matrix transpose.

Consistency of the primal and tangent linear discretizations

The primal discretization is space-time consistent of order (α,β), if, in the limit of the spatial and temporal discretizations, it holds that:

$$\begin{aligned} &\left\| \mathcal{F}^{h,n}(\mathbf{u},\mathbf{q}) \right\| \sim \mathcal{O}(h^\alpha,\tau^\beta) \\ &\left| \mathcal{J}^h(\mathbf{u},\mathbf{q}) - \mathcal{J}(\mathbf{u},\mathbf{q}) \right| \sim \mathcal{O}(h^\alpha,\tau^\beta) \; . \end{aligned} \tag{5.13}$$

We assume that the discrete primal variables are *a priori* consistent with their continuous counterparts. We will also refer to (5.13) as the residual consistency condition. Here $\mathbf{u}$ and $\mathbf{q}$ are the exact solutions to the continuous primal problem (1.13)–(5.1), while h and τ denote the size of the time and space meshes. Residual consistency is defined in a very similar manner for the discrete TLM (5.11):

$$\begin{aligned} &\left\| \left(\mathcal{F}^{h,n}\right)' [\mathbf{u},\mathbf{q}](\delta\mathbf{u},\delta\mathbf{q}) \right\| \sim \mathcal{O}(h^\alpha,\tau^\beta) \\ &\left| \left(\mathcal{J}^{h}\right)' [\mathbf{u},\mathbf{q}](\delta\mathbf{u},\delta\mathbf{q}) - \mathcal{J}'[\mathbf{u},\mathbf{q}](\delta\mathbf{u},\delta\mathbf{q}) \right| \sim \mathcal{O}(h^\alpha,\tau^\beta) \ . \end{aligned} \tag{5.14}$$

Here $\delta\mathbf{u}$ and $\delta\mathbf{q}$ are the exact solution to (5.9).

Consider now the convergence of the linearized primal variables. Note that this is a stronger property than (5.14), for it automatically implies residual consistency. Since the tangent linear problem (5.11) is linear in $(\delta\mathbf{u}^{h,n},\delta\mathbf{q}^h)$, stability and residual consistency (if proven) imply convergence of the linearized variables, in the limit of both discretizations:

$$\left\|\delta\mathbf{u}^{h,n} - \delta\mathbf{u}(t^n)\right\| \sim \mathcal{O}(h^\alpha,\, \tau^\beta) \ , \text{ as } h\to 0\,, \tau\to 0 \ . \tag{5.15}$$

Note that neither stability nor consistency are automatically inherited by the tangent linear equations from the primal problem.

Consistency of the dual discretization

Space-time consistency definitions for the dual discretization follow those for the primal problem and its linearization. We say that the adjoint discretization (5.12) is space-time consistent of order (α,β) if, in the limit of h and τ, the following relations hold:

$$\begin{aligned} &\left\| \left(\mathcal{F}^{h,n}\right)'^{*} [\mathbf{u},\mathbf{q}](\boldsymbol{\lambda}) \right\| \sim \mathcal{O}(h^\alpha,\tau^\beta) \\ &\left| \mathcal{J}_a^h(\mathbf{u},\boldsymbol{\lambda},\mathbf{q}) - \mathcal{J}^a(\mathbf{u},\boldsymbol{\lambda},\mathbf{q}) \ \right| \sim \mathcal{O}(h^\alpha,\tau^\beta) \ . \end{aligned} \tag{5.16}$$

The asymptotic order of consistency in the cost functional may be higher than (α,β) due to super-convergence effects, or dual post-processing of $\mathcal{J}_a^h$ [108].

Equation (5.16) is equivalent to saying that the linearized primal discretization (5.11) is *dual consistent* of order (α,β). Note that the dual discretization (5.12) automatically inherits the stability properties of the discrete tangent linear formulation (5.11). A crucial point in the discrete adjoint analysis is the convergence of the discrete adjoint variables. This does not follow automatically from (5.16). Instead, we need both stability of the dual formulation (5.12), *and* residual consistency (5.16). In that case, convergence of the $\boldsymbol{\lambda}^{h,n}$ follows:

$$\left\|\boldsymbol{\lambda}^{h,n} - \boldsymbol{\lambda}(t^n)\right\| \sim \mathcal{O}(h^\alpha,\tau^\beta) \ , \text{ as } h\to 0\,, \tau\to 0 \ . \tag{5.17}$$

5.2 The discontinuous Galerkin method

We illustrate the general derivation of the discontinuous Galerkin (DG) method on the hyperbolic conservation law (see 3.1):

$$\begin{array}{lcll} \mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) & = & \mathbf{f}\,, & \mathbf{x} \in \Omega\,,\, t \in [0,\, T] \\ \mathbf{u}(t, \mathbf{x}) & = & \mathbf{g}(t, \mathbf{x})\,, & \mathbf{x} \in \partial\Omega_{\text{in}} \\ \mathbf{u}(t=0, \mathbf{x}) & = & \mathbf{u}^0(\mathbf{x})\,, & \end{array}$$

The exact solution to (5.18) is $\mathbf{u} \in \mathcal{L}^2\{[0,\, T]; \mathcal{U}\}$, where $\mathcal{U}$ is a function space that guarantees sufficient smoothness for $\mathbf{u}$. The discrete spatial mesh consists of polyhedral elements, i.e., $\Omega_n^h = \bigcup_{k=1}^{K_n} D_n^k$, and the inflow boundary $\partial\Omega_{\text{in}}$. In the *modal* DG formulation, the solution to (5.18) is approximated on a given element D_n^k by a truncated expansion of orthonormal polynomials (up to and including degree J):

$$\mathbf{u}^{h,n}\big|_{D_n^k} = \sum_{j=0}^{J} \widehat{\mathbf{u}}_j^{(k),n} \psi_j(\mathbf{x}^h)\,, \tag{5.18}$$

with $\int_{D_n^k} \psi_i(\mathbf{x})\, \psi_j(\mathbf{x})\, d\mathbf{x} = \delta_{ij}$. The unknowns are then the time-dependent expansion coefficients $\mathbf{u}_j^{(k),n}$, $j = 0 \ldots J$. In the *nodal* DG formulation, the solution is given by:

$$\mathbf{u}^{h,n}\big|_{D_n^k} = \mathbf{V} \left[\widehat{\mathbf{u}}_0^{(k),n}\ \widehat{\mathbf{u}}_1^{(k),n}\ \ldots\ \widehat{\mathbf{u}}_J^{(k),n}\right]^T := \mathbf{V}\, \widehat{\mathbf{u}}^{(k),n}\,. \tag{5.19}$$

Here $\mathbf{V}$ is a block-diagonal Vandermonde interpolation matrix [22]. Note that if either the modal or nodal form of DG for (5.18) is proved to be adjoint consistent, the consistency of the other formulation follows from (5.19). The global vector of unknown expansion coefficients at t^n is

$$\widehat{\mathbf{u}}^n := \begin{bmatrix} \widehat{\mathbf{u}}^{(1),n} \\ \widehat{\mathbf{u}}^{(2),n} \\ \vdots \\ \widehat{\mathbf{u}}^{(K),n} \end{bmatrix}\,.$$

We follow the DG notation in [28] throughout this derivation. Since we will be concerned mainly with the space discretization, we omit the time dependency for the rest of this section. Let $\mathcal{U}_h \subset \mathcal{U}$ denote the discrete solution space. The test functions $\mathbf{v}^h$ are assumed to be bounded in the $\mathcal{H}^1$ Sobolev norm on each mesh element. Given two neighboring elements D_-^k and D_+^k (with a common face or edge), we let $\mathbf{u}^\pm := \mathbf{u}|_{\partial D_\pm^k}$ denote the trace of $\mathbf{u}$ taken from the interior of $D_\pm^k$, respectively. The jump in the solution over an edge (or face) is given by

$$[\![\mathbf{u}^h]\!] := \mathbf{u}_+^h \vec{\mathbf{n}}_+ + \mathbf{u}_-^h \vec{\mathbf{n}}_-\,,$$

whereas the average at $\mathbf{x} \in D_-^k \cap D_+^k$ is $\{\mathbf{u}\} := (\mathbf{u}_-^h + \mathbf{u}_+^h)/2$. On a boundary edge, we have that $\{\mathbf{u}^h\} := \mathbf{u}_+^h$, and $[\![\mathbf{u}]\!] := \mathbf{u}_+^h \, \vec{\mathbf{n}}_+$.

Define the following two discrete volume and boundary inner products:

$$\left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_{D^h} := \int_{D^h} \left(\mathbf{u}^h\right)^T \, \mathbf{v}^h \, \mathrm{d}\mathbf{x}$$
$$\left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_{\partial D^h} := \int_{\partial D^h} \left(\mathbf{u}^h\right)^T \, \mathbf{v}^h \, \mathrm{d}\mathbf{s} \; ,$$

where ∂D^h is the boundary of the element D^k. A Galerkin projection onto the solution space $\mathcal{U}_h$, followed by an application of the divergence theorem, lead to the semi-discrete DG formulation: Find $\mathbf{u}^h \in \mathcal{U}_h$, such that for all $\mathbf{v}^h \in \mathcal{U}_h$ we have that

$$\sum_{D^k} \left(\left\langle \frac{d\mathbf{u}^h}{dt}, \, \mathbf{v}^h \right\rangle_{D^k} - \left\langle \mathbf{F}(\mathbf{u}^\mathbf{h}), \, \nabla \mathbf{v}^h \right\rangle_{D^k} + \left\langle \mathbf{F}^*(\mathbf{u}_-^h, \mathbf{u}_+^h, \vec{\mathbf{n}}), \, \mathbf{v}^h \right\rangle_{\partial D^k} \right)$$
$$- \sum_{D^k} \left\langle \mathbf{f}, \, \mathbf{v}^h \right\rangle_{D^h} = 0 \; .$$

The numerical flux $\mathbf{F}^{\mathrm{DG}}(\mathbf{u}_-^h, \mathbf{u}_+^h, \vec{\mathbf{n}})$ is obtained through the solution of a Riemann problem at the boundary between two adjacent elements [22]. For Runge–Kutta DG methods, the time derivative is discretized through a Runge-Kutta quadrature [131]. Alternatively, the time dimension may also be discretized by DG, leading to a space-time DG discretization [151, 152]. The consistency of Runge–Kutta time discretizations will be discussed in detail in section 5.5.

We focus on the discontinuous Galerkin method because of its multiple advantages over the finite volume, finite differences, and continuous Galerkin approaches. DG is particularly amenable to adaptive mesh refinement and parallelization, due to the weak coupling between elements (which are connected only through the boundary fluxes $\mathbf{F}^{\mathrm{DG}}$). The order of the numerical approximation can easily be varied inside each element, since the basis functions have only local support. This avoids the AMR complications introduced by global basis functions (necessary in the continuous Galerkin approach). Also, there is no increase in stencil size when higher order approximations are used, in contrast with the finite difference and finite volume methods, where p-refinement can only be implemented by adding extra points or cells to the computational stencil.

As remarked previously, consistency of discrete adjoints with the continuous problem is by no means guaranteed, even in the fixed mesh case. Hartmann [28] proposed a framework for investigating adjoint consistency of DG schemes for elliptic problems. Lack of this adjoint consistency property leads to non-smooth discrete adjoint solutions, as well as suboptimal rates of convergence for the primal problem [153, 154, 32]. This adjoint consistency concept can be extended to hyperbolic problems by considering a semi-discrete version of the primal problem. In this case, time derivatives can be implemented with a strong-stability preserving

Runge-Kutta method [155]. Care must be taken such that source terms are also discretized in a dual consistent manner [146].

We show below that the interpolation and restriction operators for $h-$ and $p-$refinement DG are exact transposes of each other (note that this transpose relationship holds if we have an embedding between the coarse/fine solution spaces, which may not be the case for curved domains).

5.3 Adjoint interpolation and restriction operators for h/p-adaptive DG

In this section we investigate the discrete adjoints of the projection and restriction operators for h- and p-adaptive DG. These are the grid transfer operators used in an adaptive adjoint code obtained through automatic differentiation. The analysis does not consider the differentiation of the mesh refinement logic that decides a new grid size based on truncation error estimates. It is expected that differentiation of the spatial mesh refinement logic leads to inconsistent discrete adjoints, as it does in the case of temporal mesh refinement [97].

The analysis in the three subsequent sections proves the following claim.

Proposition 5.3.1. *Consider the discontinuous Galerkin method with h/p-refinement. Assume the meshes Ω_n^h and Ω_{n+1}^h are comprised of polyhedral elements. Furthermore, let the mesh transfer operators $\mathcal{I}_{n\to n+1}$ and $\mathcal{I}_{n+1\to n}$ be defined as above. Then, it holds that*

$$\mathcal{I}_{n\to n+1} = \mathcal{I}_{n+1\to n}^T \,. \tag{5.20}$$

Proof. See sections 5.3.1, 5.3.2, and 5.3.3. □

5.3.1 Hierarchical h-refinement

We first analyze the refinement and coarsening operators in the context of hierarchical mesh refinement, i.e., an embedding of nested meshes with finer and finer spacing. Implementation of this refinement strategy is facilitated by data structures such as quad- or octrees [156]. Note that the shape and dimension of the elements is arbitrary; we only assume the existence of smooth bijective mappings from a canonical (reference) element D to the active element. A quick analysis shows that both $h-$ and $p-$refinement are done using orthogonal projections. Hence, we do not lose solution accuracy (beyond the aliasing introduced by coarsening itself) by using the code generated by AD for the adjoint intergrid solution transfers.

Consider the element D^k that is refined by the AMR mechanism into P smaller elements, i.e. $D^k = \bigcup_{p=1}^P D_p^k$. Let $\mathcal{M} : D \to D^k$ be a bijective mapping from the reference element D

to the active element D^k. Similarly, $\mathcal{M}_p$ is a one-to-one and onto map from D to $D_p^k \subset D^k$. We denote the orthonormal set of basis functions on D by $\{\Psi_j(x)\}_{j=0\ldots J}$.

The interpolation operator from D^k to its set of P children elements $\bigcup_p D_p^k$ reads:

$$\mathcal{P}_{H\to h} := \left[\begin{array}{c} \mathcal{P}^1 \\ \vdots \\ \mathcal{P}^P \end{array} \right] , \tag{5.21}$$

where

$$\mathcal{P}_{ij}^p := \int_{D_p^k} \Psi_j(\mathcal{M}^{-1}(x))\Psi_i(\mathcal{M}_p^{-1}(x))\, dx \ , \ i,j = 0\ldots J \ . \tag{5.22}$$

Mesh coarsening collapses the P child elements into their parent element. The restriction operator that performs this operation is:

$$\mathcal{R}_{h\to H} := \left[\begin{array}{ccc} \mathcal{R}^1 & \ldots & \mathcal{R}^P \end{array} \right] \ . \tag{5.23}$$

with

$$\mathcal{R}_{ij}^p := \int_{D_p^k} \Psi_i(\mathcal{M}^{-1}(x))\Psi_j(\mathcal{M}_p^{-1}(x))\, dx \ , \ i,j = 0\ldots J \ . \tag{5.24}$$

From (5.21–5.22) and (5.23–5.24):

$$\mathcal{R}_{h\to H} = \mathcal{P}_{H\to h}^T \ . \tag{5.25}$$

5.3.2 *p*-refinement

Adaptive order refinement (also called p-refinement) is useful for nonlinear problems, and implies the reduction or increase of the local order of the solution on a chosen subset of the grid elements. This is equivalent to adding or removing expansion coefficients (for the modal formulation), or interpolation points (in the nodal approach). Suppose for simplicity that the AMR mechanism flagged a set $\{D^{i_q}\}_{q=1\ldots Q}$ of Q out of the K mesh elements for p-refinement. Assume the order of the solution is increased by $\widehat{Q}$ on each of the Q mesh elements. Then, the p-refinement operator on element D^{i_q} has the following form:

$$\mathcal{A}_{J\to J+\widehat{Q}} := \left[\begin{array}{c} \mathbf{I}_{J\times J} \\ \mathbf{0}_{\widehat{Q}\times J} \end{array} \right] \ . \tag{5.26}$$

The reverse operation on D^{i_q} can be written in operator form as

$$\left[\begin{array}{c} \widehat{u}_1^{(i_q)} \\ \vdots \\ \widehat{u}_N^{(i_q)} \end{array} \right] = \mathcal{A}_{J\to J+\widehat{Q}}^T \left[\begin{array}{c} \widehat{u}_1^{(i_q)} \\ \vdots \\ \widehat{u}_N^{(i_q)} \\ \vdots \\ \widehat{u}_{N+\widehat{Q}}^{(i_q)} \end{array} \right] \ . \tag{5.27}$$

Since the solution coefficients on all elements outside the refinement set remain unchanged, the transpose relationship for the global p-refinement operator follows from (5.26)–(5.27). This result and equation (5.25) prove the transpose relationship (5.20) holds with $\mathcal{C} = 1$:

$$\mathcal{I}_{n+1\to n} \;=\; \mathcal{I}^T_{n\to n+1} \; . \tag{5.28}$$

This implies that the adjoint grid transfer operators generated via reverse mode automatic differentiation retain the accuracy of their forward model counterparts. Hence, h-adaptivity together with an adjoint consistent DG discretization [28] (see also the next section), lead to a stable and consistent discrete adjoint solution. Moreover, the adjoint DG code can be generated automatically from the forward problem discretization, without requiring any post-processing of the adjoint solution.

5.3.3 h-refinement with general meshes

We now extend our analysis of interpolation and coarsening to general triangulations. Consider two meshes that cover our domain Ω: $\Omega^h_A = \bigcup_k A^k$ and $\Omega^h_B = \bigcup_m B^m$. Consider also the elements generated by all intersections of elements of Ω^h_A and Ω^h_B: denote them by $C^{k,m} = A^k \bigcap B^m$. The corresponding mesh is $\Omega^h_C = \bigcup_{k,m} C^{k,m}$.

The solution on A^k is

$$\mathbf{u}_{A^k} = \sum_j a_{\{k,j\}} \phi_{\{k,j\}}(\mathbf{x}) \; , \quad \phi_{\{k,j\}} = \phi_j \left(\mathcal{M}^{-1}_{A,k}(\mathbf{x}) \right) \; ,$$

while the solution on element B^m reads:

$$\mathbf{u}_{B^m} = \sum_i b_{\{m,i\}} \psi_{\{m,i\}}(\mathbf{x}) \; , \quad \psi_{\{m,i\}} = \psi_i \left(\mathcal{M}^{-1}_{B,m}(\mathbf{x}) \right) \; .$$

We project the solution $\mathbf{u}_A$ defined on Ω^h_A on the basis ψ to obtain the solution $\mathbf{u}_B$ on Ω^h_B. Note that

$$B^m = \bigcup_k C^{k,m} \; .$$

Consequently,

$$\begin{aligned}
b_{\{m,i\}} &= \int_{B^m} \mathbf{u}_{B^m}(\mathbf{x}) \, \psi_{\{m,i\}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\
&= \sum_k \int_{C^{k,m}} \mathbf{u}_{B^m}(\mathbf{x}) \, \psi_{\{m,i\}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\
&= \sum_k \int_{C^{k,m}} \sum_j a_{\{k,j\}} \phi_{\{k,j\}}(\mathbf{x}) \, \psi_{\{m,i\}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\
&= \sum_{\{k,j\}} \left(\int_{C^{k,m}} \phi_{\{k,j\}}(\mathbf{x}) \, \psi_{\{m,i\}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \right) a_{\{k,j\}} \; .
\end{aligned}$$

The transfer matrix that maps the solution $\mathbf{u}_A$ (defined by the a coefficients) to the solution $\mathbf{u}_B$ (defined by the b coefficients) is:

$$\{b\} = \mathcal{T}^{A\to B} \cdot \{a\} \ , \quad \mathcal{T}^{A\to B}_{\{m,i\},\{k,j\}} = \int_{C^{k,m}} \phi_{\{k,j\}}(\mathbf{x})\, \psi_{\{m,i\}}(\mathbf{x})\, \mathrm{d}\mathbf{x} \ .$$

We do similar calculations for the solution transfer from B to A. In the above formulas we interchange the roles of a and b, and of ϕ and ψ to obtain:

$$a_{\{m,i\}} \ = \ \sum_{\{k,j\}} \left(\int_{C^{m,k}} \psi_{\{k,j\}}(\mathbf{x})\, \phi_{\{m,i\}}(\mathbf{x})\, \mathrm{d}\mathbf{x} \right) b_{\{k,j\}} \ .$$

The transfer matrix that maps the solution $\mathbf{u}_B$ (defined by the b coefficients) to the solution $\mathbf{u}_A$ (defined by the a coefficients) is:

$$\{a\} = \mathcal{T}^{B\to A} \cdot \{b\} \ , \quad \mathcal{T}^{B\to A}_{\{m,i\},\{k,j\}} = \int_{C^{m,k}} \psi_{\{k,j\}}(\mathbf{x})\, \phi_{\{m,i\}}(\mathbf{x})\, \mathrm{d}\mathbf{x} \ .$$

Clearly the two intergrid operators are the transpose of one another, which means (5.28) holds in this more general case:

$$\mathcal{T}^{B\to A}_{\{m,i\},\{k,j\}} = \mathcal{T}^{A\to B}_{\{k,j\},\{m,i\}} \ .$$

5.3.4 h-refinement via quadratic centered polynomial solution reconstruction

We use the finite volume notation from chapter 3, section 3.1.

Assume a smooth exact solution $\mathbf{u}(x,t)$ to (5.18). Consider three adjacent finite volume cells of size h: C^L, C^C, and C^R, centered at x^{i-1}, x^i, and x^{i+1}, respectively. Their corresponding exact averages are $\mathbf{U}_L$, $\mathbf{U}_C$, and $\mathbf{U}_R$. The cell C^C is split into two cells C^C_L and C^C_R, each with volume $h/2$. The solution inside $C^L \bigcup C^C \bigcup C^R$ is approximated by a quadratic polynomial $\mathbf{u}^h(x)$, with the unknown coefficients determined from (3.4). We then obtain the averages on the two finer cells using equation (3.1):

$$\begin{bmatrix} \mathbf{U}^C_L \\ \mathbf{U}^C_R \end{bmatrix} = \begin{bmatrix} \frac{1}{8} & 1 & -\frac{1}{8} \\ -\frac{1}{8} & 1 & \frac{1}{8} \end{bmatrix} \begin{bmatrix} \mathbf{U}_L \\ \mathbf{U}_C \\ \mathbf{U}_R \end{bmatrix} \ . \tag{5.29}$$

We are interested in the order of accuracy of these approximations, i.e. we want to estimate the errors of the two new cell averages

$$\begin{aligned} E_L \ &:= \ \left| \mathbf{U}^C_L - \frac{2}{h} \int_{x^{i-1/2}}^{x^i} \mathbf{u}(x) \mathrm{d}x \right| \\ E_R \ &:= \ \left| \mathbf{U}^C_R - \frac{2}{h} \int_{x^i}^{x^{i+1/2}} \mathbf{u}(x) \mathrm{d}x \right| \ . \end{aligned} \tag{5.30}$$

Using (3.1) (now assumed to hold exactly for the cells of size h), we get that:

$$\max\left(\left|E_L\right|,\left|E_R\right|\right) \;=\; \frac{3}{64}h^3\left|\frac{\mathrm{d}^3\mathbf{u}}{\mathrm{d}\mathbf{x}^3}(x^i)\right| + \mathcal{O}(h^5)\;,$$

hence the approximation (5.30) is third order accurate for our uniform centered stencil. We now consider the transposed operator that coarsens C_C^L and C_C^L into a single parent cell C_C. From (5.29) it is immediately apparent that the transposed coarsening operator is equivalent to first order (conservative) averaging. Moreover, using the adjoint (transpose) of the discrete interpolation operator in equation (5.29) has the undesired side-effect of perturbing the neighbor averages:

$$\begin{aligned}
\mathbf{U}_C &= \mathbf{U}_C^L + \mathbf{U}_C^R \\
\widehat{\mathbf{U}}_L &= \mathbf{U}_L + 1/8\left(\mathbf{U}_C^L - \mathbf{U}_C^R\right) := \mathbf{U}_L + \varepsilon_L \\
\widehat{\mathbf{U}}_R &= \mathbf{U}_R + 1/8\left(-\mathbf{U}_C^L + \mathbf{U}_C^L\right) := \mathbf{U}_R + \varepsilon_R\;.
\end{aligned} \tag{5.31}$$

Since by our assumptions $\mathbf{U}_L$ and $\mathbf{U}_R$ are exact averages, and $\mathbf{u}(\mathbf{x})$ is smooth over $C^L \bigcup C^C \bigcup C^R$, one can bound the perturbations ε_L and ε_R using Taylor approximations:

$$\max\left(\left|\varepsilon_L\right|,\,\left|\varepsilon_R\right|\right) = \frac{1}{16}h\left|\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}(x_i)\right| + \mathcal{O}(h^3)\;.$$

These numerical side-effects should be avoided in practice (preferably through post-processing of the discrete adjoint code). Nevertheless, this grid coarsening operation remains only first order. This can be also shown to hold for higher degree (centered or upwind) polynomial reconstructions, and for higher dimensional problems, as outlined in the next section.

5.3.5 General intergrid transfer operators in the finite volume method

We now consider a more general formulation of the intergrid transfer operators used in the finite volume method. In what follows $\widehat{J}$ is defined by equation (3.2). Consider the polynomial reconstruction formula (3.4) over $\widehat{J}$ cells C^1, …, $C^{\widehat{J}}$:

$$\mathbf{u}^h(\mathbf{x}) \;=\; \mathbf{a}^T \cdot \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_{\widehat{J}}(\mathbf{x}) \end{bmatrix}\;,$$

where $\mathbf{a} = \left[a_1,\,\ldots,\,a_{\widehat{J}}\right]^T$ are the polynomial coefficients, and ϕ_1, …, $\phi_{\widehat{J}}$ are a set of basis functions for the space of multivariate polynomials of degree J under consideration. Then, the average on cell C_j is

$$\frac{1}{\mathrm{Vol}(C^j)}\int_{C^j}\mathbf{u}^h(\mathbf{x})\,\mathrm{d}\mathbf{x} = \mathbf{w}_j^T\cdot\mathbf{a} = \mathbf{U}_j\;, \tag{5.32}$$

with $\mathbf{w}_j$ denoting the integration weights:

$$(\mathbf{w}_j)_i = \frac{1}{\mathrm{Vol}(C^j)} \int_{C^j} \phi_i(\mathbf{x})\, \mathrm{d}\mathbf{x} \ , \ \forall\, i = 1, \ \ldots, \ \widehat{J} \ . \tag{5.33}$$

Equation (5.32) leads to the matrix formulation over all $\widehat{J}$ cells:

$$\mathbf{W}\,\mathbf{a} = \mathbf{U} \ . \tag{5.34}$$

In equation (5.34), the $\mathbf{W} \in \mathbb{R}^{\widehat{J} \times \widehat{J}}$ is the weight matrix, and $\mathbf{U}$ denotes the column vector of $\widehat{J}$ cell averages. Suppose now that cell C^i is refined into K non-overlapping sub-cells $C_1^i, \ \ldots, \ C_K^i$. The averages inside the smaller K cells are given by:

$$\mathbf{U}_i^k \ = \ \mathbf{v}_k^T \mathbf{a} = \mathbf{v}_k^T \mathbf{W}^{-1} \mathbf{U} \ , \ \forall\, i = 1, \ \ldots, \ K \ ,$$

where the new integration weights $\mathbf{v}_k$ are defined by

$$(\mathbf{v}_k)_j := \frac{1}{\mathrm{Vol}(C_i^j)} \int_{C_i^j} \phi_j(\mathbf{x})\, \mathrm{d}\mathbf{x} \ , \ \forall\, i = 1, \ \ldots, \ \widehat{J} \ ,$$

and satisfy the equation

$$\sum_{k=1}^{K} \mathbf{v}_k^T = \mathbf{w}_i^T = \mathbf{e}_i^T \mathbf{W} \ . \tag{5.35}$$

Here $\mathbf{e}_i$ is the i-th unit basis vector of $\mathbb{R}^{\widehat{J}}$. For small values of J, and smooth $\mathbf{u}(\mathbf{x}, t)$, the averages in the finer sub-cells can be shown to be accurate of order h^{J+1}. Larger-sized stencils introduce oscillations in the approximating polynomials, hence some WENO-type stabilization is required [157]. The analysis of the transpose of the stabilization algorithm is beyond the scope of this chapter.

Proposition 5.3.2. *The transpose relationship (5.20) does not hold for refinement and coarsening operators used in finite volume mesh transfers. The transpose of the refinement operator, when used for coarsening, introduces first-order perturbations in the neighbor averages of the refined finite volume cell.*

Proof. We can write the refinement operation in matrix form as:

$$\begin{bmatrix} \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_{i-1} \\ \mathbf{U}_i^1 \\ \vdots \\ \mathbf{U}_i^K \\ \mathbf{U}_{i+1} \\ \vdots \\ \mathbf{U}_{\widehat{J}} \end{bmatrix} = \left[\begin{array}{ccc} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline & \mathcal{P}_K & \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right] \mathbf{U} \ , \tag{5.36}$$

with the prolongation sub-matrix

$$\mathcal{P}_K = \left[\begin{array}{c} \mathbf{v}_1^T \mathbf{W}^{-1} \\ \vdots \\ \mathbf{v}_K^T \mathbf{W}^{-1} \end{array} \right] \in \mathbb{R}^{K \times \widehat{J}} . \tag{5.37}$$

Use of the adjoint of (5.36) as a coarsening operator yields the following average for cell C_i:

$$\widetilde{\mathbf{U}}_i = \sum_{k=1}^{K} \left(\mathbf{W}^{-T} \mathbf{v}_k \right)_i \mathbf{U}_i^k . \tag{5.38}$$

From (5.35), we get a first order conservative reconstruction of the solution average inside cell C_i. However, there is one undesired side-effect of this operator. The average solution values inside all of the other $\widehat{J} - 1$ cells in the interpolation stencil are polluted by a first order perturbation stemming from the transposed restriction operator (5.38):

$$\widetilde{\mathbf{U}}_j \;\; = \;\; \mathbf{U}_j + \sum_{k=1}^{K} \left(\mathbf{W}^{-T} \mathbf{v}_k \right)_j \mathbf{U}_i^k \; , \; \forall \, j \neq i \; .$$

This perturbation is $\mathcal{O}(h)$ on general uniform and non-uniform grids. To establish this estimate, note that, using (5.35), we have

$$\sum_{k=1}^{K} \left(\mathbf{W}^{-T} \mathbf{v}_k \right)_j = 0 \; , \; \forall \, j \neq i \; .$$

In the general case this first order error term does not vanish. □

5.4 Space-time duality relations in function spaces

As mentioned previously, the concept of adjoint consistency, together with its implications in optimization, have been investigated for steady-state problems by Lu [32], Harriman, Gavaghan and Süli [154], Hartmann and Houston [158, 159], and Oliver and Darmofal [146]. Hartmann [28] proposed a general framework for establishing adjoint consistency for DG discretizations of stationary PDE models. We leverage previous results on dual consistency for temporal [122], and spatial discretizations [158] to give a unified framework for the analysis of adjoint consistency of space-time DG discretizations. This section discusses space-time duality relations for continuous model formulations. A general strategy to construct the adjoint system is given, applicable whenever the cost functional and the associated model differential operators satisfy a set of compatibility conditions. The next section will discuss dual consistency of the time quadratures for Runge–Kutta DG discretizations (assumed to be dual consistent in space).

Consider again equation (1.13). For simplicity of exposition, we rewrite (1.13) in the form:

$$\begin{aligned} \mathbf{u}_t &= N[\mathbf{u}] + \mathbf{f}\ ,\ \mathbf{x} \in \Omega\ ,\ t \in [0,T] \\ B[\mathbf{u}] &= \mathbf{g}\ ,\ \mathbf{x} \in \Gamma\ ,\ t \in [0,T] \\ \mathbf{u}(t=0,\mathbf{x}) &= \mathbf{u}^0\ ,\ \mathbf{x} \in \Omega\ . \end{aligned} \tag{5.39}$$

The PDE system admits solutions $\mathbf{u} : [0,T] \to \mathcal{U}$, such that $\mathbf{u} \in \mathcal{L}^2([0,T];\mathcal{U})$, and $\mathbf{u}_t \in \mathcal{L}^2([0,T];\mathcal{U})$, where $\mathcal{U}$ is an appropriate function space. Here N and B are Frechét differentiable, nonlinear differential operators, containing spatial and boundary derivative terms. We denote the Frèchet derivatives by:

$$\begin{aligned} L\,\mathbf{w} &= N'[\mathbf{u}]\,\mathbf{w} \\ B'\,\mathbf{w} &= B'[\mathbf{u}]\,\mathbf{w}\ . \end{aligned}$$

Consider a nonlinear cost functional of the form

$$\begin{aligned} \mathcal{J}(\mathbf{u}) &= \int_0^T \int_\Omega J_\Omega\big[C_\Omega\,\mathbf{u}\big]\,\mathrm{d}\mathbf{x}\,\mathrm{d}t + \int_0^T \int_\Gamma J_\Gamma\big[C_\Gamma\,\mathbf{u}\big]\,\mathrm{d}s\,\mathrm{d}t \\ &\quad + \int_\Omega K_\Omega\big[E_\Omega\,\mathbf{u}\big]_{t=T}\,\mathrm{d}\mathbf{x}\ . \end{aligned} \tag{5.40}$$

The differential operators C_Ω and E_Ω act on the domain Ω, while C_Γ is a boundary operator (all are assumed to be Frechét differentiable). Their Frèchet derivatives are denoted by C'_Ω, E'_Ω, and C'_Γ, respectively. Also, let

$$\begin{aligned} j_\Omega &= \big(J'_\Omega\big[C_\Omega\,\mathbf{u}\big]\big)^T \\ j_\Gamma &= \big(J'_\Gamma\big[C_\Gamma\,\mathbf{u}\big]\big)^T \\ k_\Omega &= \big(K'_\Omega\big[E_\Omega\,\mathbf{u}\big]\big)^T\ . \end{aligned}$$

5.4.1 The tangent linear PDE

Small variations $\delta\mathbf{u}$ in the solution $\mathbf{u}(\mathbf{x},t)$ of (5.39) satisfy (up to first order) the tangent linear model. These equations can be obtained from (5.39) by linearization in the direction $(\delta\mathbf{u},\,\delta\mathbf{f},\,\delta\mathbf{g})$:

$$\begin{aligned} \delta\mathbf{u}_t &= N'[\mathbf{u}]\ \delta\mathbf{u} + \delta\mathbf{f}\ ,\ \mathbf{x} \in \Omega\ ,\ t \in [0,T] \\ B'[\mathbf{u}]\,\delta\mathbf{u} &= \delta\mathbf{g}\ ,\ \mathbf{x} \in \Gamma\ ,\ t \in [0,T] \\ \delta\mathbf{u}(t=0,\mathbf{x}) &= \delta\mathbf{u}^0\ ,\ \mathbf{x} \in \Omega\ . \end{aligned} \tag{5.41}$$

We denote

$$\begin{aligned} \langle \mathbf{u},\,\mathbf{v}\rangle_{[0,T]\times\Gamma} &:= \int_0^T \int_\Gamma \mathbf{u}^T\,\mathbf{v}\,\mathrm{d}\mathbf{s}\,\mathrm{d}t \\ \langle \mathbf{u},\,\mathbf{v}\rangle_{[0,T]\times\Omega} &:= \int_0^T \int_\Omega \mathbf{u}^T\,\mathbf{v}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\ . \end{aligned} \tag{5.42}$$

The space-time weak form of (5.41) is written in terms of space-time inner products as:

$$\begin{aligned} \langle \mathbf{w}_t \,,\, \mathbf{v} \rangle_{[0,T]\times\Omega} &= \langle L\,[\mathbf{u}]\ \mathbf{w} \,,\, \mathbf{v} \rangle_{[0,T]\times\Omega} + \langle \delta\mathbf{f} \,,\, \mathbf{v} \rangle_{[0,T]\times\Omega} \\ \langle B'[\mathbf{u}]\,\mathbf{w} \,,\, \mathbf{v} \rangle_{[0,T]\times\Gamma} &= \langle \delta\mathbf{g} \,,\, \mathbf{v} \rangle_{[0,T]\times\Gamma} \\ \langle \mathbf{w},\, \mathbf{v} \rangle_{\Omega}\big|_{t=0} &= \left\langle \delta\mathbf{u}^0,\, \mathbf{v} \right\rangle_{\Omega} \ ,\ \forall\ \mathbf{v} \in \mathcal{L}^2\left([0,T];\mathcal{U}\right) \ . \end{aligned} \tag{5.43}$$

The space of all possible solutions of (5.43) is:

$$\mathcal{U}^{\rm tlm} = \{\mathbf{w} \in \mathcal{U} \mid \exists\ \delta\mathbf{f}\,,\ \delta\mathbf{g}\,,\ \delta\mathbf{u}_0 \ \ s.t.\ \ \mathbf{w} \text{ is a solution of (5.43)}\} \ . \tag{5.44}$$

Clearly $\mathcal{U}^{\rm tlm}$ is a vector subspace of $\mathcal{U}$. The variation of the cost functional (5.40) is

$$\delta\mathcal{J} = \mathcal{J}'[\mathbf{u}]\,\delta\mathbf{u} \ , \tag{5.45}$$

where

$$\begin{aligned} \mathcal{J}'[\mathbf{u}]\,\mathbf{w} &= \int_0^T \int_\Omega J'_\Omega\big[C_\Omega\,\mathbf{u}\big]\,C'_\Omega\,\mathbf{w}\,d\mathbf{x}\,dt + \int_0^T \int_\Gamma J'_\Gamma\big[C_\Gamma\,\mathbf{u}\big]\,C'_\Gamma\,\mathbf{w}\,d\mathbf{s}\,dt \\ &\quad + \left(\int_\Omega K'_\Omega\big[E_\Omega\,\mathbf{u}\big]\ E'_\Omega\,\mathbf{w}\ d\mathbf{x}\right)\Big|_{t=T} \\ &:= \langle C'_\Omega\,\mathbf{w}\ ,\ j_\Omega\rangle_{[0,T]\times\Omega} + \langle C'_\Gamma\,\mathbf{w}\ ,\ j_\Gamma\rangle_{[0,T]\times\Gamma} + \langle E'_\Omega\,\mathbf{w}\ ,\ k_\Omega\rangle_\Omega\big|_{t=T} \ . \end{aligned}$$

To compute the variation (5.45) due to $\delta\mathbf{u}^0$, $\delta\mathbf{f}$, and $\delta\mathbf{g}$, one runs the TLM (5.43) to obtain $\delta\mathbf{u}$, and uses it in (5.45) to compute $\delta\mathcal{J}$. A new TLM solution is needed for each set of perturbations $\delta\mathbf{u}^0$, $\delta\mathbf{f}$, and $\delta\mathbf{g}$.

5.4.2 The adjoint PDE

We wish to express the variation (5.45) as

$$\delta\mathcal{J} \ = \ \Big(C^{\rm adj}_\Omega\,\boldsymbol{\lambda}\ ,\ \delta\mathbf{f}\Big\rangle_{[0,T]\times\Omega} + \Big\langle C^{\rm adj}_\Gamma\,\boldsymbol{\lambda}\ ,\ \delta\mathbf{g}\Big\rangle_{[0,T]\times\Gamma} + \Big\langle E^{\rm adj}_\Omega\,\boldsymbol{\lambda}|_{t=0}\ ,\ \delta\mathbf{u}_0\Big\rangle_\Omega \ , \tag{5.46}$$

for any perturbations $\delta\mathbf{u}^0$, $\delta\mathbf{f}$, and $\delta\mathbf{g}$. The adjoint variables $\boldsymbol{\lambda}$ are obtained by solving the dual problem

$$\begin{aligned} -\boldsymbol{\lambda}_t &= L^*\,\boldsymbol{\lambda} + \mathbf{f}^{\rm adj}\ ,\ \mathbf{x}\in\Omega\ ,\ t\in[0,T] \\ B^{\rm adj}\,\boldsymbol{\lambda} &= \mathbf{g}^{\rm adj}\ ,\ \mathbf{x}\in\Gamma\ ,\ t\in[0,T] \\ \boldsymbol{\lambda}(t=T,\mathbf{x}) &= E^{\rm adj}_\Omega\,k_\Omega\ ,\ \mathbf{x}\in\Omega \ . \end{aligned} \tag{5.47}$$

The operators $B^{\rm adj}$ and $C^{\rm adj}_\Gamma$ need to be chosen such that (5.46) and (5.45) are equivalent. Note that duality implies that the following relations hold:

- The domain weight j_Ω in the forward linearized cost function (5.45) determines the adjoint domain forcing $\mathbf{f}^{\rm adj}$. The domain forcing $\delta\mathbf{f}$ in the forward linearized problem determines the domain weight in the adjoint expression of the cost function (5.46).
- The boundary weight j_Γ in the forward linearized cost function (5.45) determines the adjoint boundary forcing $\mathbf{g}^{\rm adj}$. The boundary forcing $\delta\mathbf{g}$ in the forward linearized problem determines the boundary weight in the adjoint expression of the cost function (5.46).
- The domain forcing at the final time k_Ω in the forward linearized cost function (5.45) determines the final value of the adjoint variable $\boldsymbol{\lambda}(t=T,\mathbf{x})$.

The time-space weak form of (5.48) is

$$\begin{aligned} -\left\langle \mathbf{w}\ ,\ \boldsymbol{\lambda}_t\right\rangle_{[0,T]\times\Omega} &= \left\langle \mathbf{w}\ ,\ L^*\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Omega} + \left\langle \mathbf{w}\ ,\ \mathbf{f}^{\rm adj}\right\rangle_{[0,T]\times\Omega} \\ \left\langle \mathbf{w}\ ,\ B^{\rm adj}\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma} &= \left\langle \mathbf{w}\ ,\ \mathbf{g}^{\rm adj}\right\rangle_{[0,T]\times\Gamma} \\ \left\langle \mathbf{w}\ ,\ \boldsymbol{\lambda}|_{t=T}\right\rangle_{\Omega} &= \left\langle \mathbf{w}\ ,\ \boldsymbol{\lambda}^{\rm F}\right\rangle_{\Omega}\ . \end{aligned} \tag{5.48}$$

5.4.3 Compatibility conditions

Consider the integration by parts formulas

$$\left\langle L\,\mathbf{w}\,,\,\mathbf{v}\right\rangle_\Omega = \left\langle \mathbf{w}\,,\,L^*\,\mathbf{v}\right\rangle_\Omega + \sum_i \left\langle F_i^L\,\mathbf{w}\,,G_i^L\,\mathbf{v}\right\rangle_\Gamma,\quad \forall\ \mathbf{w},\mathbf{v}\in\mathcal{U} \tag{5.49a}$$

$$\left\langle C'_\Omega\,\mathbf{w}\,,\,\mathbf{v}\right\rangle_\Omega = \left\langle \mathbf{w}\,,\,C'^*_\Omega\,\mathbf{v}\right\rangle_\Omega + \sum_i \left\langle F_i^C\,\mathbf{w}\,,G_i^C\,\mathbf{v}\right\rangle_\Gamma,\quad \forall\ \mathbf{w},\mathbf{v}\in\mathcal{U} \tag{5.49b}$$

$$\left\langle E'_\Omega\,\mathbf{w}\,,\,\mathbf{v}\right\rangle_\Omega = \left\langle \mathbf{w}\,,\,E'^*_\Omega\,\mathbf{v}\right\rangle_\Omega + \sum_i \left\langle F_i^E\,\mathbf{w}\,,G_i^E\,\mathbf{v}\right\rangle_\Gamma,\quad \forall\ \mathbf{w},\mathbf{v}\in\mathcal{U} \tag{5.49c}$$

where $F_i^{L,C,E}$, $G_i^{L,C,E}$ are boundary linear differential operators that come from the integration by parts of the linear operators L, C'_Ω, and E'_Ω, respectively. We impose a first compatibility condition which ensures that the boundary terms coming from the integration by parts of C'_Ω vanish for all w that satisfy the boundary condition (5.44) of the tangent linear model (5.43):

$$\begin{aligned} \text{Compatibility condition 1}: \qquad & \sum_i \left\langle F_i^C\,\mathbf{w}\,,G_i^C\,\mathbf{v}\right\rangle_\Gamma = 0\ , \\ & \forall\ \mathbf{v}\in\mathcal{U}\ ,\ \forall\ \mathbf{w}\in\mathcal{U}^{\rm tlm}\ . \end{aligned} \tag{5.50}$$

The second compatibility condition ensures that the boundary terms coming from the integration by parts of E'_Ω vanish for all perturbations consistent with (5.44):

$$\begin{aligned} \text{Compatibility condition 2}: \qquad & \sum_i \left\langle F_i^E\,w\,,G_i^E\,v\right\rangle_\Gamma = 0\ , \\ & \forall\ \mathbf{v}\in\mathcal{U}\ ,\ \forall\ \mathbf{w}\in\mathcal{U}^{\rm tlm}\ . \end{aligned} \tag{5.51}$$

The compatibility conditions (5.50) and (5.51) simplify the integration by parts formulas (5.49) to

$$\langle L\,\mathbf{w}\,,\,\mathbf{v}\rangle_\Omega = \langle \mathbf{w}\,,\,L^*\,\mathbf{v}\rangle_\Omega + \sum_i \left\langle F_i^L\,\mathbf{w}\,, G_i^L\,\mathbf{v}\right\rangle_\Gamma, \quad \forall\,\mathbf{w},\mathbf{v}\in\mathcal{U} \tag{5.52a}$$

$$\langle C_\Omega'\,\mathbf{w}\,,\,\mathbf{v}\rangle_\Omega = \langle \mathbf{w}\,,\,C_\Omega'^*\,\mathbf{v}\rangle_\Omega\ , \quad \forall\,\mathbf{w}\in\mathcal{U}^{\rm tlm}\ ,\ \ \forall\,\mathbf{v}\in\mathcal{U} \tag{5.52b}$$

$$\langle E_\Omega'\,\mathbf{w}\,,\,\mathbf{v}\rangle_\Omega = \langle \mathbf{w}\,,\,E_\Omega'^*\,\mathbf{v}\rangle_\Omega\ , \quad \forall\,\mathbf{w}\in\mathcal{U}^{\rm tlm}\ ,\ \ \forall\,\mathbf{v}\in\mathcal{U}\ . \tag{5.52c}$$

After integration by parts the TLM (5.43) becomes:

$$\begin{aligned}
-\langle \mathbf{w}\,,\,\boldsymbol{\lambda}_t\rangle_{[0,T]\times\Omega} + \langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_\Omega\big|_0^T &= \langle \mathbf{w}\,,\,L^*\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Omega} \\
&\quad + \sum_i \left\langle F_i^L\,\mathbf{w}\,, G_i^L\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma} \\
&\quad + \langle \delta\mathbf{f}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Omega} \\
\langle B'\,\mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Gamma} &= \langle \delta\mathbf{g}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Gamma}\ , \\
\langle \mathbf{w}|_{t=0}\,,\,\boldsymbol{\lambda}\rangle_\Omega &= \left\langle \delta\mathbf{u}^0\,,\,\boldsymbol{\lambda}\right\rangle_\Omega\ .
\end{aligned} \tag{5.53}$$

Equations (5.53) and (5.49) lead to

$$\begin{aligned}
\left\langle \mathbf{w}\,,\,\mathbf{f}^{\rm adj}\right\rangle_{[0,T]\times\Omega} + \langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_\Omega\big|_{t=T} &= \langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_\Omega\big|_{t=0} + \langle \delta\mathbf{f}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Omega} \\
&\quad + \sum_i \left\langle F_i^L\,\mathbf{w}\,, G_i^L\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma}\ .
\end{aligned} \tag{5.54}$$

The variation of the cost functional (5.45) can be rewritten as:

$$\begin{aligned}
\mathcal{J}'[\mathbf{u}]\,\mathbf{w} &= \langle C_\Omega'\,\mathbf{w}\,,\,j_\Omega\rangle_{[0,T]\times\Omega} + \langle C_\Gamma'\,\mathbf{w}\,,\,j_\Gamma\rangle_{[0,T]\times\Gamma} + \langle E_\Omega'\,\mathbf{w}\,,\,k_\Omega\rangle_\Omega\big|_{t=T} \\
&= \langle \mathbf{w}\,,\,C_\Omega'^*\,j_\Omega\rangle_{[0,T]\times\Omega} + \langle C_\Gamma'\,\mathbf{w}\,,\,j_\Gamma\rangle_{[0,T]\times\Gamma} + \langle \mathbf{w}\,,\,E_\Omega'^*\,k_\Omega\rangle_\Omega\big|_{t=T}\ .
\end{aligned} \tag{5.55}$$

We make the following identifications:

$$\begin{aligned}
\mathbf{f}^{\rm adj} &= C_\Omega'^* j_\Omega \\
\mathbf{g}^{\rm adj} &= j_\Gamma \\
\boldsymbol{\lambda}_F &= E_\Omega'^*\,k_\Omega\ .
\end{aligned}$$

Then, the adjoint problem (5.48) reads

$$\begin{aligned}
-\boldsymbol{\lambda}_t &= L^*\,\boldsymbol{\lambda} + C_\Omega'^*\,j_\Omega\ ,\ \mathbf{x}\in\Omega\ ,\ t\in[0,T] \\
B^{\rm adj}\,\boldsymbol{\lambda} &= j_\Gamma\ ,\ \mathbf{x}\in\Gamma\ ,\ t\in[0,T] \\
\boldsymbol{\lambda}(t=T,\mathbf{x}) &= E'^*\,k_\Omega\ ,\ \mathbf{x}\in\Omega\ ,
\end{aligned} \tag{5.56}$$

and equation (5.55) becomes:

$$\begin{aligned}
\mathcal{J}'[\mathbf{u}]\,\mathbf{w} &= -\langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_{\Omega}\big|_{t=T} + \langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_{\Omega}\big|_{t=0} + \langle \mathbf{w}_t - L\,\mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Omega} \\
&\quad - \sum_i \left\langle F_i^L\,\mathbf{w}\,, G_i^L\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma} + \sum_i \left\langle F_i^C\,\mathbf{w}\,, G_i^C\,j_\Omega\right\rangle_{[0,T]\times\Gamma} \\
&\quad + \left\langle C_\Gamma'\,\mathbf{w}\,,\; B^{\mathrm{adj}}\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma} + \langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_{\Omega}\big|_{t=T} \\
&= \langle \mathbf{w}\,,\,\boldsymbol{\lambda}\rangle_{\Omega}\big|_{t=0} + \langle \delta\mathbf{f}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Omega} \\
&\quad + \left\langle C_\Gamma'\,\mathbf{w}\,,\; B^{\mathrm{adj}}\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma} + \sum_i \left\langle F_i^L\,\mathbf{w}\,, G_i^L\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma} \\
&\quad + \sum_i \left\langle F_i^C\,\mathbf{w}\,, G_i^C\,j_\Omega\right\rangle_{[0,T]\times\Gamma}\;.
\end{aligned}$$

If the adjoint boundary condition is defined by the relation

$$\begin{aligned}
\left\langle B^{\mathrm{adj}}\,\boldsymbol{\lambda}\,,\; C_\Gamma'\mathbf{w}\right\rangle_{[0,T]\times\Gamma} &= \left\langle C_\Gamma^{\mathrm{adj}}\,\boldsymbol{\lambda}\,,\; B'\,\mathbf{w}\right\rangle_{[0,T]\times\Gamma} \\
&\quad - \sum_i \left\langle F_i^L\,\mathbf{w}\,, G_i^L\,\boldsymbol{\lambda}\right\rangle_{[0,T]\times\Gamma}\;, \qquad (5.57)
\end{aligned}$$

then

$$\mathcal{J}'[\mathbf{u}]\,\mathbf{w} = \left\langle \delta\mathbf{u}^0\,,\,\boldsymbol{\lambda}\right\rangle_{\Omega}\big|_{t=0} + \langle \delta\mathbf{f}\,,\,\boldsymbol{\lambda}\rangle_{[0,T]\times\Omega} + \left\langle C_\Gamma^{\mathrm{adj}}\,\boldsymbol{\lambda}\,,\; \delta\mathbf{g}\right\rangle_{[0,T]\times\Gamma}\;. \qquad (5.58)$$

Equation (5.57) is ensured by the third compatibility condition. There exist well defined boundary operators B^{adj} and C_Γ^{adj} such that the following holds:

$$\begin{aligned}
&\text{Compatibility condition 3 :} \\
&(L\,\mathbf{w}\,,\,\mathbf{v})_\Omega - \left(B'\,\mathbf{w}\,,\; C_\Gamma^{\mathrm{adj}}\,\mathbf{v}\right)_\Gamma = (\mathbf{w}\,,\,L^*\,\mathbf{v})_\Omega - \left(C_\Gamma'\mathbf{w}\,,\; B^{\mathrm{adj}}\,\mathbf{v}\right)_\Gamma \\
&\qquad\qquad \forall\;\mathbf{w}\in\mathcal{U}^{\mathrm{tlm}}\,,\;\mathbf{v}\in\mathcal{U}\;. \qquad (5.59)
\end{aligned}$$

Here we have used (5.52) to relate (5.57) and (5.59).

Definition 5.4.1 (Compatibility)**.** *We say that the cost function and the PDE are compatible if the three compatibility conditions (5.50), (5.51), and (5.59) hold.*

Remark. The third compatibility condition (5.59) is discussed in [50, 28]. The authors assume $C = I$ (the identity operator), and $E = 0$, therefore (5.50), and (5.51) trivially hold. Equation (5.59) is the only compatibility condition needed in this simpler setting.

5.4.4 An example: the linear advection-diffusion equation

As an example, we will consider the linear advection-diffusion problem:

$$\begin{aligned} \mathbf{u}_t &= -\nabla\cdot(\vec{a}\,\mathbf{u}) + \Delta\mathbf{u} + \mathbf{f}\,,\ \mathbf{x}\in\Omega\,,\ t\in[0,\,T] \\ \mathbf{u} &= \mathbf{g}_D\,,\ \mathbf{x}\in\Gamma_D = \Gamma_- \\ \vec{\mathbf{n}}\cdot\nabla\mathbf{u} &= \mathbf{g}_N\,,\ \mathbf{x}\in\Gamma_N = \Gamma\setminus\Gamma_D \\ \mathbf{u}(t=0,\mathbf{x}) &= \mathbf{u}^0\,. \end{aligned} \tag{5.60}$$

Here $\Gamma_- = \{\mathbf{x}\in\Gamma\,|\,\vec{a}(\mathbf{x})\cdot\vec{\mathbf{n}}(\mathbf{x})<0\}$ is the advective inflow boundary. The nonlinear cost functional quantifies the mismatch between the model trajectory $\mathbf{u}(t,\mathbf{x})$, and a given reference state $\mathbf{u}^{\rm ref}$. We penalize high variations in the boundary derivatives (to avoid boundary layers in our numerical solution). Thus, $\mathcal{J}$ is defined as:

$$\mathcal{J}(\mathbf{u}) = \frac{1}{2}\int_0^T\int_\Omega \left\|\mathbf{u}-\mathbf{u}^{\rm ref}\right\|_2^2\,\mathrm{d}\mathbf{x}\,\mathrm{d}t + \frac{1}{2}\int_0^T\int_{\Gamma_D}(\nabla\mathbf{u}\cdot\vec{\mathbf{n}})^2\,\mathrm{d}\mathbf{s}\,\mathrm{d}t\,. \tag{5.61}$$

We immediately identify the operators

$$\begin{aligned} &J_\Omega\big[C_\Omega\,\mathbf{u}\big] = \frac{1}{2}\left\|\mathbf{u}-\mathbf{u}^{\rm ref}\right\|_2^2\ ;\quad j_\Omega = \mathbf{u}-\mathbf{u}^{\rm ref}\ ; \\ &C_{\Gamma_D}(\mathbf{u}) = \nabla\mathbf{u}\cdot\vec{\mathbf{n}}\ ;\quad C'_{\Gamma_D}(\mathbf{w}) = \nabla\mathbf{w}\cdot\vec{\mathbf{n}}\ ; \\ &J_{\Gamma_D}\big[C_{\Gamma_D}\,\mathbf{u}\big] = \frac{1}{2}(\nabla\mathbf{u}\cdot\vec{\mathbf{n}})^2\ ;\quad j_{\Gamma_D} = \nabla\mathbf{u}\cdot\vec{\mathbf{n}}\,. \end{aligned}$$

The tangent linear PDE

The TLM of (5.60) reads:

$$\begin{aligned} \mathbf{w}_t &= -\nabla\cdot(\vec{a}\,\mathbf{w}) + \Delta\mathbf{w} + \delta\mathbf{f}\,,\ \mathbf{x}\in\Omega\,,\ t\in[0,\,T] \\ B'_D\,\mathbf{w} := \mathbf{w} &= \delta\mathbf{g}_D\,,\ \mathbf{x}\in\Gamma_D \\ B'_N\,\mathbf{w} := \vec{\mathbf{n}}\cdot\nabla\mathbf{w} &= \delta\mathbf{g}_N\,,\ \mathbf{x}\in\Gamma_N \\ \mathbf{w}(t=0,\mathbf{x}) &= \delta\mathbf{u}^0\,, \end{aligned}$$

and the Fréchet derivative of the cost functional $\mathcal{J}$ in the direction $\mathbf{w}$ can be written as:

$$\mathcal{J}'[\mathbf{u}]\,\mathbf{w} = \left\langle\mathbf{u}-\mathbf{u}^{\rm ref},\mathbf{w}\right\rangle_{[0,T]\times\Omega} + \left\langle\frac{\partial\mathbf{u}}{\partial\vec{\mathbf{n}}},\frac{\partial\mathbf{w}}{\partial\vec{\mathbf{n}}}\right\rangle_{[0,T]\times\Gamma_D}\,.$$

The adjoint PDE

The integration by parts formula becomes

$$\begin{aligned}\langle -\nabla\cdot(\vec{a}\,\mathbf{w})+\Delta\mathbf{w}\,,\,\mathbf{v}\rangle_{[0,T]\times\Omega} &= \langle \mathbf{w}\,,\,\vec{a}\cdot\nabla\mathbf{v}+\Delta\mathbf{v}\rangle_{[0,T]\times\Omega}\\ &+\left\langle \mathbf{v}\,,\,-\mathbf{w}\,\vec{a}\cdot\vec{\mathbf{n}}+\frac{\partial\mathbf{w}}{\partial\vec{\mathbf{n}}}\right\rangle_{[0,T]\times\Gamma}\\ &+\left\langle -\frac{\partial\mathbf{v}}{\partial\vec{\mathbf{n}}}\,,\,\mathbf{w}\right\rangle_{[0,T]\times\Gamma}\,.\end{aligned}$$

Then, we can easily identify the volume and boundary operators:

$$\begin{aligned}L\,\mathbf{w} &= -\nabla\cdot(\vec{a}\,\mathbf{w})+\Delta\mathbf{w}\\ L^*\,\mathbf{v} &= \vec{a}\cdot\nabla\mathbf{v}+\Delta\mathbf{v}\\ F_1\,\mathbf{w} &= -\mathbf{w}\,\vec{a}\cdot\vec{\mathbf{n}}+\frac{\partial\mathbf{w}}{\partial\vec{\mathbf{n}}}\,,\quad G_1\,\mathbf{v}=\mathbf{v}\\ F_2\,\mathbf{w} &= \mathbf{w}\,,\quad G_2\,\mathbf{v}=-\frac{\partial\mathbf{v}}{\partial\vec{\mathbf{n}}}\,.\end{aligned}$$

The compatibility conditions (5.50) and (5.51) are trivially satisfied. The operators B^{adj} and C^{adj}_{Γ} are determined by the third compatibility condition, which for our example reduces to:

$$\begin{aligned}&\langle \mathbf{v}\,,\,-\mathbf{w}\,\vec{a}\cdot\vec{\mathbf{n}}+\vec{\mathbf{n}}\cdot\nabla\mathbf{w}\rangle_{[0,T]\times\Gamma}+\langle -\vec{\mathbf{n}}\cdot\nabla\mathbf{v}\,,\,\mathbf{w}\rangle_{[0,T]\times\Gamma}\\ &=\left\langle \mathbf{w}\,,\,C^{\text{adj}}_{\Gamma_D}\,\mathbf{v}\right\rangle_{[0,T]\times\Gamma_D}-\left\langle \vec{\mathbf{n}}\cdot\nabla\mathbf{w}\,,\,B^{\text{adj}}_{\Gamma_D}\,\mathbf{v}\right\rangle_{[0,T]\times\Gamma_D}\\ &+\left\langle \vec{\mathbf{n}}\cdot\nabla\mathbf{w}\,,\,C^{\text{adj}}_{\Gamma_N}\,\mathbf{v}\right\rangle_{[0,T]\times\Gamma_N}-\left\langle \mathbf{w}\,,\,B^{\text{adj}}_{\Gamma_N}\,\mathbf{v}\right\rangle_{[0,T]\times\Gamma_N}\,.\end{aligned}$$

Using the linearity of the inner product integrals, we can establish that:

$$B^{\text{adj}}_{\Gamma_D}\mathbf{v} := -\mathbf{v}\,,\ \mathbf{x}\in\Gamma_D \tag{5.62}$$

$$B^{\text{adj}}_{\Gamma_N}\mathbf{v} := \vec{a}\cdot\vec{\mathbf{n}}\,\mathbf{v}+\frac{\partial\mathbf{v}}{\partial\vec{\mathbf{n}}}\,,\ \mathbf{x}\in\Gamma_N \tag{5.63}$$

$$C^{\text{adj}}_{\Gamma_D}\mathbf{v} := -\vec{a}\cdot\vec{\mathbf{n}}\,\mathbf{v}-\frac{\partial\mathbf{v}}{\partial\vec{\mathbf{n}}}\,,\ \mathbf{x}\in\Gamma_D \tag{5.64}$$

$$C^{\text{adj}}_{\Gamma_N}\mathbf{v} := \mathbf{v}\,,\ \mathbf{x}\in\Gamma_N\,, \tag{5.65}$$

and write the adjoint final value problem for (5.60):

$$\begin{aligned}-\boldsymbol{\lambda}_t &= \vec{a}\cdot\nabla\boldsymbol{\lambda}+\Delta\boldsymbol{\lambda}+\mathbf{u}-\mathbf{u}^{\text{ref}}\,,\ \mathbf{x}\in\Omega\,,\ t\in(T,\,0]\\ \boldsymbol{\lambda} &= -\nabla\mathbf{u}\cdot\vec{\mathbf{n}}\,,\ \mathbf{x}\in\Gamma_D\\ \vec{\mathbf{n}}\cdot\nabla\boldsymbol{\lambda}+\vec{a}\cdot\vec{\mathbf{n}}\,\boldsymbol{\lambda} &= 0\,,\ \mathbf{x}\in\Gamma_N\\ \boldsymbol{\lambda}(t=T,\mathbf{x}) &= 0\,.\end{aligned}$$

We will revisit this example below, in the context of fully discrete models.

5.5 Duality relations and space-time adjoints for discrete models

If the time dimension is discretized by DG (see, e.g., [151, 152]), then we have a space-time DG discretization. The consistency analysis follows closely the one presented in [28]. The only difference is that the integrals are taken in space-time.

We now consider a time discretization by Runge Kutta methods. A semi-discretization in space of the continuous primal problem (5.39) leads to the following semi-discrete model [28]:

Find $\mathbf{u}^h \in \mathcal{L}^2\left([0,\, T]\,;\, \mathcal{U}_h\right)$ such that $(u^h)_t \in \mathcal{L}^2\left([0,\, T]\,;\, \mathcal{U}_h\right)$ and

$$\begin{aligned}\left\langle \frac{\partial \mathbf{u}^h}{\partial t},\, \mathbf{v}^h(t) \right\rangle_{\Omega} &= \mathcal{N}\left(t; \mathbf{u}^h,\, \mathbf{v}^h\right) + \left\langle \mathbf{f},\, \mathbf{v}^h \right\rangle_{\Omega} + \mathcal{B}\left(\mathbf{g},\, \mathbf{v}^h\right) \\ &\qquad \forall\, \mathbf{v}^h \in \mathcal{L}^2\left([0,T];\mathcal{U}_h\right)\ ,\ \text{a.a. } t \in [0,T]\ . \end{aligned} \tag{5.66}$$

Here the semi-linear form $\mathcal{N}$ is nonlinear in $\mathbf{u}^h$, and linear in $\mathbf{v}^h$. $\mathcal{B}\left(\cdot,\cdot\right)$ is a bilinear form defined on the boundary Γ, which depends on the prescribed boundary data $\mathbf{g}^h$. Let $\mathcal{N}'[\mathbf{u}^h] := \partial\mathcal{N}/\partial\mathbf{u}^h$ be the Fréchet derivative of $\mathcal{N}$ with respect to $\mathbf{u}^h$. The TLM of (5.66) reads

$$\begin{aligned}\left\langle \frac{\partial \mathbf{w}^h}{\partial t}, \mathbf{v}^h(t) \right\rangle_{\Omega} &= \mathcal{N}'[\mathbf{u}^h]\left(t; \mathbf{w}^h(t)\,,\, \mathbf{v}^h\right) + \left\langle \delta\mathbf{f}(t), \mathbf{v}^h \right\rangle + \mathcal{B}\left(\delta\mathbf{g},\, \mathbf{v}^h\right) \\ &\qquad \forall\, \mathbf{v}^h \in \mathcal{L}^2\left([0,T];\mathcal{U}_h^{\mathrm{tlm}}\right)\ ,\ \text{a.a. } t \in [0,T]\ , \end{aligned} \tag{5.67}$$

where the TLM solution $\mathbf{w}^h \in \mathcal{L}^2\left([0,T];\mathcal{U}_h\right)$. The semi-discrete cost functional

$$\begin{aligned}\mathcal{J}_h(\mathbf{u}^h) &= \int_0^T \int_\Omega j_\Omega\left[C_\Omega\, \mathbf{u}^h\right] \mathrm{d}\mathbf{x}\, \mathrm{d}t + \int_0^T \int_\Gamma j_\Gamma\left[C_\Gamma\, \mathbf{u}^h\right] \mathbf{ds}\, \mathrm{d}t \\ &\quad + \int_\Omega k_\Omega\left[E_\Omega\, \mathbf{u}^h\,\right]_{t=T} \mathrm{d}\mathbf{x}\ , \end{aligned} \tag{5.68}$$

is a discretization of the continuous functional $\mathcal{J}$ in equation (5.40), and has a variation given by

$$\begin{aligned}\mathcal{J}_h'\, \mathbf{w}^h &= \int_0^T \left\langle \left(j_\Omega\left[C_\Omega\, \mathbf{u}^h(t)\right]\right)^T\ ,\ C_\Omega'\, \mathbf{w}^h(t) \right\rangle_\Omega \mathrm{d}t \\ &\quad + \int_0^T \left\langle \left(j_\Gamma\left[C_\Gamma\, \mathbf{u}^h(t)\right]\right)^T\ ,\ C_\Gamma'\, \mathbf{w}^h(t) \right\rangle_\Gamma \mathrm{d}t \\ &\quad + \left\langle \left(k_\Omega'\left[E_\Omega\,,\, \mathbf{u}^h(T)\,\right]\right)^T\ ,\ E_\Omega'\, \mathbf{w}^h(T) \right\rangle_\Omega\ . \end{aligned} \tag{5.69}$$

A full discretization of the PDE is obtained by discretizing the time derivative in (5.66) using a Runge–Kutta method [131]. In the following, $\mathbf{u}^n \in \mathcal{U}_h$ is the fully discrete solution at t^n,

$\mathbf{U}_i^n \in \mathcal{U}_h$ is the i-th stage vector at time step n, and $T_i^n = t^n + c_i\, h^{n+1}$ is the stage time moment. The time grid has $N+1$ points: from $t^0 = 0$, up to $t^N = T$, and $t^{n+1} = t^n + \tau^{n+1}$. For simplicity of notation, we omit the discrete space superscripts in the following discussion. The Runge–Kutta discretization of (5.66) reads:

$$\begin{aligned}
\langle \mathbf{U}_i^n\,,\, \mathbf{v}\rangle_\Omega &= \langle \mathbf{u}^n, \mathbf{v}\rangle_\Omega + \tau^{n+1} \sum_{j=1}^{s} a_{i,j}\, \left[\mathcal{N}\left(T_j^n; \mathbf{U}_j^n,\, \mathbf{v}\right) \right. \\
&\quad \left. + \left\langle \mathbf{f}_j^n,\, \mathbf{v}\right\rangle + \mathcal{B}\left(\mathbf{g}_j^n,\, \mathbf{v}\right) \right] \;,\; \forall\, \mathbf{v} \in \mathcal{U}_h^{\mathrm{tlm}} \qquad (5.70) \\
\left\langle \mathbf{u}^{n+1}, \mathbf{v}\right\rangle_\Omega &= \langle \mathbf{u}^n, \mathbf{v}\rangle_\Omega + \tau^{n+1} \sum_{i=1}^{s} b_i\, \left[\mathcal{N}\left(T_i^n; \mathbf{U}_i^n,\, \mathbf{v}\right) + \left\langle \mathbf{f}_i^n,\, \mathbf{v}\right\rangle + \mathcal{B}\left(\mathbf{g}_i^n,\, \mathbf{v}\right) \right] .
\end{aligned}$$

Proposition 5.5.1. *Assuming that the spatial semi-discretization (5.66) is dual consistent, where consistency is defined as in [28]. Then the space-time discretization (5.71) is also dual consistent in time. Moreover, it inherits the temporal order of accuracy from the primal discretization.*

Proof. Due to the linearity of the Runge–Kutta procedure, the TLM of the fully discrete system reads:

$$\begin{aligned}
\langle \mathbf{W}_i^n\,,\, \mathbf{v}\rangle_\Omega &= (\mathbf{w}^n, \mathbf{v})_\Omega + \tau^{n+1} \sum_{j=1}^{s} a_{i,j}\, \left[\mathcal{N}'[\mathbf{U}_j^n]\left(T_j^n; \mathbf{W}_j^n\,,\, \mathbf{v}\right) \right. \\
&\quad \left. + \left\langle \delta\mathbf{f}_j^n,\, \mathbf{v}\right\rangle + \mathcal{B}\left(\delta\mathbf{g}_j^n,\, \mathbf{v}\right) \right] \;,\; \forall\, \mathbf{v} \in \mathcal{U}_h \\
\left\langle \mathbf{w}^{n+1},\, \mathbf{v}\right\rangle_\Omega &= \langle \mathbf{w}^n,\, \mathbf{v}\rangle_\Omega + \tau^{n+1} \sum_{i=1}^{s} b_i\, \left[\mathcal{N}'[\mathbf{U}_i^n]\left(T_i^n; \mathbf{W}_i^n,\, \mathbf{v}\right) \right. \\
&\quad \left. + \left\langle \delta\mathbf{f}_i^n,\, \mathbf{v}\right\rangle + \mathcal{B}\left(\delta\mathbf{g}_i^n,\, \mathbf{v}\right) \right] .
\end{aligned}$$

The time integration of the cost functional is discretized according to the Runge–Kutta quadrature. The variation of the fully discrete cost functional is:

$$\begin{aligned}
\mathcal{J}_h'\,\mathbf{w} &= \sum_{n=0}^{N-1} \tau^{n+1} \sum_{i=1}^{s} b_i \left\langle \left(j_\Omega\left[C_\Omega\, \mathbf{U}_i^n\right]\right)^T \,,\, C_\Omega'\, \mathbf{w}(T_i^n)\right\rangle_\Omega \\
&\quad + \sum_{n=0}^{N-1} \tau^{n+1} \sum_{i=1}^{s} b_i \left\langle \left(j_\Gamma\left[C_\Gamma\, \mathbf{U}_i^n\right]\right)^T \,,\, C_\Gamma'\, \mathbf{w}(T_i^n)\right\rangle_\Gamma \\
&\quad + \left\langle \left(k_\Omega'\left[E_\Omega\, \mathbf{u}^N\right]\right)^T \,,\, E_\Omega'\, \mathbf{w}(t^N)\right\rangle_\Omega \,.
\end{aligned}$$

We rewrite the TLM of the fully discrete system, to outline the use of different discrete test

functions $\boldsymbol{\lambda}(t^n, \mathbf{x}) \in \mathcal{U}_h$ (which will later be interpreted as the adjoint variables):

$$\begin{aligned}
\left\langle \mathbf{w}^0, \boldsymbol{\lambda}^0 \right\rangle_\Omega &= \left\langle \delta\mathbf{u}^0,\, \boldsymbol{\lambda}^0 \right\rangle_\Omega \;,\; \forall\, \boldsymbol{\lambda}^0 \in \mathcal{U}_h \\
\langle \mathbf{W}_i^n \,,\, \boldsymbol{\theta}_i^n \rangle_\Omega &= \langle \mathbf{w}^n, \boldsymbol{\theta}_i^n \rangle_\Omega + \tau^{n+1} \sum_{j=1}^{s} a_{i,j} \,\Big[\mathcal{N}'[\mathbf{U}_j^n] \left(T_j^n; \mathbf{W}_j^n, \boldsymbol{\theta}_i^n\right) \\
&\quad + \left\langle \delta\mathbf{f}_j^n,\, \boldsymbol{\theta}_i^n \right\rangle + \mathcal{B}\left(\delta\mathbf{g}_j^n,\, \boldsymbol{\theta}_i^n\right) \Big] \;,\; \forall\, \boldsymbol{\theta}_i^n \in \mathcal{U}_h \\
\left\langle \mathbf{w}^{n+1}, \boldsymbol{\lambda}^{n+1} \right\rangle_\Omega &= \left\langle \mathbf{w}^n, \boldsymbol{\lambda}^{n+1} \right\rangle_\Omega + \tau^{n+1} \sum_{i=1}^{s} b_i \,\Big[\mathcal{N}'[\mathbf{U}_i^n] \left(T_i^n; \mathbf{W}_i^n,\, \boldsymbol{\lambda}^{n+1}\right) \\
&\quad + \left\langle \delta\mathbf{f}_i^n,\, \boldsymbol{\lambda}^{n+1} \right\rangle + \mathcal{B}\left(\delta\mathbf{g}_i^n,\, \boldsymbol{\lambda}^{n+1}\right) \Big] \;,\; \forall\, \boldsymbol{\lambda}^{n+1} \in \mathcal{U}_h \;.
\end{aligned}$$

Consider all of the above relations for $n = 0, \ldots, N-1$. We identify the terms involving the same $\mathbf{W}_i^n$, and $\mathbf{w}^n$ arguments on the left and right hand sides, and obtain the following correspondence:

$$\begin{aligned}
\langle \mathbf{W}_i^n \,,\, \boldsymbol{\theta}_i^n \rangle_\Omega \;&\leftrightarrow\; \tau^{n+1} \sum_{\ell=1}^{s} a_{\ell,i}\, \mathcal{N}'[\mathbf{U}_i^n] \left(T_i^n; \mathbf{W}_i^n,\, \boldsymbol{\theta}_{n,\ell}\right) \\
&\quad + \tau^{n+1}\, b_i\, \mathcal{N}'[\mathbf{U}_i^n] \left(T_i^n; \mathbf{W}_i^n,\, \boldsymbol{\lambda}^{n+1}\right) \\
\langle \mathbf{w}^n, \boldsymbol{\lambda}^n \rangle_\Omega \;&\leftrightarrow\; \left\langle \mathbf{w}^n,\, \boldsymbol{\lambda}^{n+1} \right\rangle_\Omega + \sum_{i=1}^{s} \langle \mathbf{w}^n,\, \boldsymbol{\theta}_i^n \rangle_\Omega \;.
\end{aligned}$$

We now define the discrete adjoint system as:

$$\begin{aligned}
(\mathbf{w} \,,\, \boldsymbol{\theta}_i^n)_\Omega &= \tau^{n+1}\, \mathcal{N}'[\mathbf{U}_i^n] \left(T_i^n; \mathbf{w} \,,\, b_i \boldsymbol{\lambda}^{n+1} + \sum_{\ell=1}^{s} a_{\ell,i}\, \boldsymbol{\theta}_\ell^n \right) \\
&\quad - \tau^{n+1} \sum_{i=1}^{s} b_i \left\langle \left(j_\Omega\big[C_\Omega\, \mathbf{U}_i^n\big] \right)^T \,,\, C_\Omega'\, \mathbf{w} \right\rangle_\Omega \\
&\quad - \tau^{n+1} \sum_{i=1}^{s} b_i \left\langle \left(j_\Gamma\big[C_\Gamma\, \mathbf{U}_i^n\big] \right)^T \,,\, C_\Gamma'\, \mathbf{w} \right\rangle_\Gamma \;,\; \forall\, \mathbf{w} \in \mathcal{U}_h^{\mathrm{tlm}} \\
\langle \mathbf{w}, \boldsymbol{\lambda}^n \rangle_\Omega &= \left\langle \mathbf{w},\, \boldsymbol{\lambda}^{n+1} \right\rangle_\Omega + \sum_{i=1}^{s} \langle \mathbf{w},\, \boldsymbol{\theta}_i^n \rangle_\Omega \;,\; \forall\, \mathbf{w} \in \mathcal{U}_h^{\mathrm{tlm}}
\end{aligned} \tag{5.71}$$

The sum of the TLM relations for $n = 0, \ldots, N-1$ gives:

$$\mathcal{J}_h'\, \mathbf{w}^0 = \left\langle \delta\mathbf{u}^0,\, \boldsymbol{\lambda}^0 \right\rangle_\Omega - \left\langle \mathbf{w}^N,\, \boldsymbol{\lambda}^N \right\rangle_\Omega + \left\langle \left(k_\Omega'\big[E_\Omega\, \mathbf{u}^N\big] \right)^T \,,\, E_\Omega'\, \mathbf{w}^N \right\rangle_\Omega + S_{\mathbf{f}} + S_{\mathbf{g}} \;, \tag{5.72}$$

where

$$S_{\mathbf{f}} = \sum_{n=0}^{N-1} \tau^{n+1} \sum_{i,j=1}^{s} a_{i,j} \left\langle \delta\mathbf{f}_j^n,\, \boldsymbol{\theta}_i^n \right\rangle_\Omega + \sum_{n=0}^{N-1} \tau^{n+1} \sum_{i=1}^{s} b_i \left\langle \delta\mathbf{f}_i^n,\, \boldsymbol{\lambda}^{n+1} \right\rangle_\Omega \;, \tag{5.73}$$

and

$$S_{\mathbf{g}} \;=\; \sum_{n=0}^{N-1} \tau^{n+1} \sum_{i,j=1}^{s} a_{i,j}\, \mathcal{B}\left(\delta \mathbf{g}^{n,j},\, \boldsymbol{\theta}_i^n\right) + \sum_{n=0}^{N-1} \tau^{n+1} \sum_{i=1}^{s} b_i\, \mathcal{B}\left(\delta \mathbf{g}_i^n,\, \boldsymbol{\lambda}^{n+1}\right) \,. \tag{5.74}$$

We examine in more detail the terms $S_{\mathbf{f}}$ and $S_{\mathbf{g}}$. From the correspondence between the discrete adjoint "stages"

$$\boldsymbol{\theta}_j^n \leftrightarrow \tau^{n+1}\, b_j\, \boldsymbol{\lambda}^{n+1} + \tau^{n+1} \sum_{i=1}^{s} a_{i,j}\, \boldsymbol{\theta}_i^n \,, \tag{5.75}$$

it follows that

$$S_{\mathbf{f}} = \sum_{n=0}^{N-1} \sum_{j=1}^{s} \left\langle \delta \mathbf{f}_j^n,\, \boldsymbol{\theta}_j^n \right\rangle_{\Omega} \tag{5.76a}$$

$$S_{\mathbf{g}} = \sum_{n=0}^{N-1} \sum_{j=1}^{s} \mathcal{B}\left(\delta \mathbf{g}_j^n,\, \boldsymbol{\theta}_j^n\right) \,. \tag{5.76b}$$

Following Hager [33], change variables in (5.76) using the correspondence (5.75). First, from (5.75), assuming all Runge-Kutta coefficients $b_i \neq 0$, one gets

$$\frac{1}{\tau^{n+1}\, b_j} \boldsymbol{\theta}_j^n \leftrightarrow \boldsymbol{\lambda}^{n+1} + \sum_{\ell=1}^{s} \frac{a_{\ell,j}}{b_j} \boldsymbol{\theta}_\ell^n \,.$$

Let $\widetilde{\boldsymbol{\theta}}_j^n$ denote the stages of the formal adjoint Runge–Kutta method (see [94]), where

$$\widetilde{\boldsymbol{\theta}}_j^n := \frac{1}{\tau^{n+1}\, b_j} \boldsymbol{\theta}_j^n \,.$$

Then, the formal adjoint stage correspondence becomes

$$\widetilde{\boldsymbol{\theta}}_j^n \leftrightarrow \boldsymbol{\lambda}^{n+1} + \tau^{n+1} \sum_{\ell=1}^{s} \frac{a_{\ell,j}\, b_\ell}{b_j}\, \widetilde{\boldsymbol{\theta}}_\ell^n \,.$$

Replacing this expression in equation (5.76), we arrive at:

$$S_{\mathbf{f}} = \sum_{n=0}^{N-1} \tau^{n+1} \sum_{j=1}^{s} b_j \left\langle \widetilde{\boldsymbol{\theta}}_j^n,\, \delta \mathbf{f}_j^n \right\rangle_{\Omega} \approx \left\langle \boldsymbol{\lambda},\, \delta \mathbf{f} \right\rangle_{[0,\,T]\times\Omega} \,. \tag{5.77}$$

The last (approximate) equality follows from the consistence theory of Runge–Kutta quadratures for time integrals [131]. We note that for control problems (unlike inverse problems),

some additional order conditions are needed for the formal adjoints of Runge–Kutta methods to achieve orders 3 and above [33]. A similar result can be derived for $S_{\mathbf{g}}$, namely:

$$S_{\mathbf{g}} \approx \int_0^T \mathcal{B}(\delta\mathbf{g},\, \boldsymbol{\lambda})\, \mathrm{d}t \ .$$

Define the final adjoint condition by

$$\left\langle \boldsymbol{\lambda}^N,\, \mathbf{w} \right\rangle_{\Omega} = \left\langle \left(k'_{\Omega}\big[E_{\Omega}\, \mathbf{u}^N\big] \right)^T \ ,\ E'_{\Omega}\, \mathbf{w} \right\rangle_{\Omega} \ ,\ \forall\, \mathbf{w} \in \mathcal{U}_h^{\mathrm{tlm}} \ , \tag{5.78}$$

to get

$$\mathcal{J}'_h\, \mathbf{w} \approx \left\langle \delta\mathbf{u}^0,\, \boldsymbol{\lambda}^0 \right\rangle_{[0,\,T]\times\Omega} + \left\langle \delta\mathbf{f},\, \boldsymbol{\lambda} \right\rangle_{[0,\,T]\times\Omega} + \int_0^T \mathcal{B}\left(\delta\mathbf{g},\, \boldsymbol{\lambda}\right)\, \mathrm{d}t \ .$$

This completes the proof.

□

The discrete adjoint variables $\boldsymbol{\lambda}^n$ can yield different sensitivities, depending on the direction in which the Fréchet derivative of $\mathcal{J}_h$ is computed:

- Differentiation of $\mathcal{J}_h$ along $(\delta\mathbf{u}^0,\, 0,\, 0)$ yields the gradient of the cost functional with respect to the initial conditions:

$$\left(E_{\Omega}^{\mathrm{adj}}\right)^h\ \boldsymbol{\lambda}^0 = \frac{\mathrm{d}\mathcal{J}^h}{\mathrm{d}\mathbf{u}^0} \ . \tag{5.79}$$

- If the tangent linear model is obtained by linearization around $(0,\, \delta\mathbf{f},\, 0)$, then we obtain the sensitivities with respect to changes in the primal equation volume forcing:

$$\left(C_{\Omega}^{\mathrm{adj}}\right)^h\ \boldsymbol{\lambda}^n = \frac{\mathrm{d}\mathcal{J}^h}{\mathrm{d}\mathbf{f}^n} \ . \tag{5.80}$$

- The consistency of the boundary sensitivities does not follow directly from the consistency of the dual discretization. Indeed, for the example given in the next section, we obtain inconsistent boundary sensitivities from a dual consistent DG discretization. Along with dual consistency of the DG discretization, one also needs adjoint consistency for the boundary functional $\mathcal{B}$ (defined by the primal discretization). We say that $\mathcal{B}$ is dual consistent *iff*, for any admissible boundary perturbation $\delta\mathbf{g}$, there exists a consistent discretization $\left(C_{\Gamma}^{\mathrm{adj}}\right)^h$ of the continuous differential operator $C_{\Gamma}^{\mathrm{adj}}$, such that

$$\mathcal{B}(\delta\mathbf{g},\, \boldsymbol{\lambda}) = \left\langle \delta\mathbf{g},\, \left(C_{\Gamma}^{\mathrm{adj}}\right)^h\ \boldsymbol{\lambda} \right\rangle_{\Gamma} \ . \tag{5.81}$$

This does not hold true for all discretizations. The next section will look at the symmetric interior penalty DG discretization of the advection-diffusion system (5.60). While the discretization itself is dual consistent, the boundary functional $\mathcal{B}^h$ is shown to be adjoint inconsistent.

Note that the discrete adjoint model (5.71)–(5.78) is obtained by applying the discrete Runge–Kutta adjoint numerical method to the semi-discrete adjoint system

$$\begin{aligned}\left\langle \mathbf{w},\, \frac{\partial \boldsymbol{\lambda}}{\partial t} \right\rangle_{\Omega} \;&=\; \mathcal{N}'[\mathbf{u}]\,(t; \mathbf{w}, \boldsymbol{\lambda}) - \langle \mathbf{w},\, j_{\Omega}[\, C_{\Omega}\mathbf{u}\,]\rangle_{\Omega} \\ &\qquad \forall\, \mathbf{w} \in \mathcal{L}^2\left([0,T]\,; \mathcal{U}_h^{\mathrm{tlm}}\right)\ ,\ \text{a.a.}\, t \in [0,T]\ . \end{aligned} \tag{5.82}$$

According to [122] the discrete adjoint Runge Kutta method provides the same order of consistency as the forward Runge Kutta method.

In conclusion, the fully discrete adjoint model (5.71)–(5.78) is equivalent to applying a method of lines discretization to the continuous adjoint PDE. The space discretization is done with the discrete adjoint DG method, and is consistent with the same order as the forward DG discretization. The time discretization is done with the discrete Runge Kutta adjoint method; the time consistency of the adjoint discretization is the same as the one of the forward method.

5.5.1 Space-time consistency analysis of the upwind SIPG advection-diffusion DG discretization

The upwind symmetric interior penalty (SIPG) DG semi-discretization [28] for the advection-diffusion PDE (5.60) reads:

$$\left\langle \frac{\partial \mathbf{u}^h}{\partial t},\, \mathbf{v}^h \right\rangle_{\Omega} = \mathcal{N}(\mathbf{u}^h,\, \mathbf{v}^h) + \mathcal{B}(\mathbf{g}^h,\, \mathbf{v}^h) - \mathcal{L}(\mathbf{f}^h,\, \mathbf{v}^h)\ , \tag{5.83}$$

with

$$\mathcal{N}(\mathbf{u}^h,\, \mathbf{v}^h) := \mathcal{N}_{\mathrm{diff}}(\mathbf{u}^h,\, \mathbf{v}^h) + \mathcal{N}_{\mathrm{adv}}(\mathbf{u}^h,\, \mathbf{v}^h)\ ,$$

and

$$\begin{aligned}
\mathcal{N}_{\text{adv}}(\mathbf{u}^h,\,\mathbf{v}^h) &= -\int_\Omega \mathbf{u}^h\,\vec{a}\cdot\nabla\mathbf{v}^h\,\mathrm{d}\mathbf{x} + \sum_{D^k}\int_{\partial D^k_-\setminus\Gamma}\vec{a}\cdot\vec{\mathbf{n}}\,\mathbf{u}^h_-\mathbf{v}^h_+\,\mathrm{d}\mathbf{s}\\
&\quad+\sum_{D^k}\int_{\partial D^k_+}\vec{a}\cdot\vec{\mathbf{n}}\,\mathbf{u}^h_+\mathbf{v}^h_+\,\mathrm{d}\mathbf{s}\,.\\
\mathcal{N}_{\text{diff}}(\mathbf{u}^h,\,\mathbf{v}^h) &= \int_\Omega\nabla\mathbf{u}^h\cdot\nabla\mathbf{v}^h\,\mathrm{d}\mathbf{x} - \sum_{D^k}\int_{\partial D^k\setminus\Gamma}\frac{1}{2}[\![\mathbf{u}^h]\!]\cdot\nabla\mathbf{v}^h\,\mathrm{d}\mathbf{s}\\
&\quad-\sum_{D^k}\int_{\partial D^k\setminus\Gamma_N}\left\{\nabla\mathbf{u}^h\right\}\cdot\vec{\mathbf{n}}\,\mathbf{v}^h\,\mathrm{d}\mathbf{s}+\sum_{D^k}\int_{\partial D^k}\phi\,[\![\mathbf{u}^h]\!]\cdot\vec{\mathbf{n}}\,\mathbf{v}^h\,\mathrm{d}\mathbf{s}\\
&\quad-\int_{\Gamma_D}\mathbf{u}^h\,\vec{\mathbf{n}}\cdot\nabla\mathbf{v}^h\,\mathrm{d}\mathbf{s}\,,
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\mathcal{L}(\mathbf{f}^h,\mathbf{v}^h) &= \int_\Omega\mathbf{f}^h\,\mathbf{v}^h\,\mathrm{d}\mathrm{x}\\
\mathcal{B}(\mathbf{g}^h,\,\mathbf{v}^h) &= \int_{\Gamma_D}\vec{a}\cdot\vec{\mathbf{n}}\,\mathbf{g}^h_D\,\mathbf{v}^h\,\mathrm{d}\mathbf{s}+\int_{\Gamma_D}\mathbf{g}^h_D\,\nabla\mathbf{v}^h\cdot\vec{\mathbf{n}}\,\mathrm{d}\mathbf{s}\\
&\quad-\int_{\Gamma_D}\phi\,\mathbf{g}^h_D\mathbf{v}^h\,\mathrm{d}\mathbf{s}-\int_{\Gamma_N}\mathbf{g}^h_N\,\mathbf{v}^h\,\mathrm{d}\mathbf{s}\,.
\end{aligned}$$

Here we denote the penalization parameter by $\phi\geq\phi_0>0$. The residual form discrete adjoint of the bilinear forms $\mathcal{N}_{\text{diff}}$ and $\mathcal{N}_{\text{adv}}$ follows from integrating by parts the primal discretizations [28]:

$$\begin{aligned}
\mathcal{N}^*_{\text{adv}}(\mathbf{w}^h,\,\boldsymbol{\lambda}^h) &:= -\int_\Omega\mathbf{w}^h\,\vec{a}\cdot\nabla\boldsymbol{\lambda}^h\,\mathrm{d}\mathbf{x}+\sum_{D^k}\int_{\partial D^k_+\setminus\Gamma}\mathbf{w}^h_+\,\vec{a}\cdot[\![\boldsymbol{\lambda}^h]\!]\,\mathrm{d}\mathbf{s}\\
&\quad+\int_{\Gamma_N}\mathbf{w}^h\,\vec{a}\cdot\vec{\mathbf{n}}\,\boldsymbol{\lambda}^h\,\mathrm{d}\mathbf{s}\,.\\
\mathcal{N}^*_{\text{diff}}(\mathbf{w}^h,\,\boldsymbol{\lambda}^h) &:= -\int_\Omega\mathbf{w}^h\,\Delta\boldsymbol{\lambda}^h\,\mathrm{d}\mathbf{x}\\
&\quad+\sum_{D^k}\int_{\partial D^k\setminus\Gamma}\mathbf{w}^h\left[\frac{1}{2}[\![\nabla\boldsymbol{\lambda}^h]\!]+\phi[\![\boldsymbol{\lambda}^h]\!]\cdot\vec{\mathbf{n}}\right]\mathrm{d}\mathbf{s}\\
&\quad-\sum_{D^k}\int_{\partial D^k\setminus\Gamma}\frac{1}{2}\nabla\mathbf{w}\cdot[\![\boldsymbol{\lambda}^h]\!]\,\mathrm{d}\mathbf{s}+\int_{\Gamma_N}\mathbf{w}^h\,\vec{\mathbf{n}}\cdot\nabla\boldsymbol{\lambda}^h\,\mathrm{d}\mathbf{s}\\
&\quad+\int_{\Gamma_D}\phi\,\boldsymbol{\lambda}^h\,\mathbf{w}^h\,\mathrm{d}\mathbf{s}-\int_{\Gamma_D}\nabla\mathbf{w}^h\cdot\vec{\mathbf{n}}\,\boldsymbol{\lambda}^h\,\mathrm{d}\mathbf{s}\,.
\end{aligned}$$

The semi-discrete formulation of (5.61) reads:

$$\mathcal{J}^h(\mathbf{u}^h) \;:=\; \frac{1}{2}\int_0^T\int_\Omega \left\|\mathbf{u}^h - \mathbf{u}^{\text{ref}}\right\|^2 \,\mathrm{d}\mathbf{x} + \frac{1}{2}\int_0^T\int_{\Gamma_D} \left(\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}}\right)^2 \,\mathrm{d}\mathbf{s}\;,$$

hence its Fréchet derivative is calculated as

$$\mathcal{J}_h'[\mathbf{u}^h](\mathbf{w}^h) \;=\; \left\langle \mathbf{u}^h - \mathbf{u}^{\text{ref}},\, \mathbf{w}^h\right\rangle_{[0,T]\times\Omega} + \left\langle \nabla\mathbf{u}^h\cdot\vec{\mathbf{n}},\, \nabla\mathbf{w}^h\cdot\vec{\mathbf{n}}\right\rangle_{[0,T]\times\Gamma_D}\;.$$

The semi-discrete adjoint equation has the following form:

$$-\left\langle \mathbf{w}^h,\, \frac{\partial\boldsymbol{\lambda}^h}{\partial t}\right\rangle_\Omega \;=\; -\mathcal{N}^*_{\text{diff}}(\mathbf{w}^h,\,\boldsymbol{\lambda}^h) - \mathcal{N}^*_{\text{adv}}(\mathbf{w}^h,\,\boldsymbol{\lambda}^h) + \mathcal{J}_h'[\mathbf{u}^h](\mathbf{w}^h)\;. \tag{5.84}$$

To investigate the dual consistency of the adjoint residuals for our particular discretization, we must first recast (5.84) in residual-based form [28]. We get:

$$\begin{aligned}&\int_\Omega \mathbf{w}^h\,\mathbf{R}^*_\Omega(\boldsymbol{\lambda}^h)\,\mathrm{d}\mathbf{x} + \sum_{D^k}\int_{\partial D^k\setminus\Gamma} \mathbf{w}^h\,\mathbf{r}^*_\Omega(\boldsymbol{\lambda}^h) + \nabla\mathbf{w}^h\cdot\rho^*_\Omega(\boldsymbol{\lambda}^h)\,\mathrm{d}\mathbf{s}\\ &+\int_\Gamma \mathbf{w}^h\,\mathbf{r}^*_\Gamma(\boldsymbol{\lambda}^h) + \nabla\mathbf{w}^h\cdot\rho^*_\Gamma(\boldsymbol{\lambda}^h)\,\mathrm{d}\mathbf{s} = 0\;,\;\forall\,\mathbf{w}^h\in\mathcal{U}_h\;.\end{aligned} \tag{5.85}$$

From (5.84)–(5.85), we identify the following dual residuals:

- Inside Ω:

  $$\mathbf{R}^*_\Omega(\boldsymbol{\lambda}^h) \;:=\; -\frac{\partial\boldsymbol{\lambda}^h}{\partial t} + \Delta\boldsymbol{\lambda}^h + \vec{a}\cdot\nabla\boldsymbol{\lambda}^h + (\mathbf{u}^h - \mathbf{u}^{\text{ref}})\;.$$

  From the continuous adjoint equation inside Ω we see that $\mathbf{R}^*_\Omega(\boldsymbol{\lambda}) = 0$, so the volume terms of the adjoint semi-discretization (5.84) are dual consistent.

- On the inter-element boundaries (excluding the domain boundary):

  $$\begin{aligned}\mathbf{r}^*_\Omega(\boldsymbol{\lambda}^h) &= -\vec{a}\cdot[\![\boldsymbol{\lambda}^h]\!] - \frac{1}{2}[\![\nabla\boldsymbol{\lambda}^n]\!] + \phi[\![\boldsymbol{\lambda}^h]\!]\cdot\vec{\mathbf{n}}\;,\\ \rho^*_\Omega(\boldsymbol{\lambda}^h) &= \frac{1}{2}[\![\boldsymbol{\lambda}^h]\!]\;.\end{aligned}$$

  Using the continuity of the strong form adjoint solution $\boldsymbol{\lambda}$, we get that both dual residuals are zero when evaluated at $\boldsymbol{\lambda}$.

- On the outflow boundary (with respect to the advective flux), Γ_N:

  $$\begin{aligned}\mathbf{r}^*_{\Gamma_N}(\boldsymbol{\lambda}^h) &= -\boldsymbol{\lambda}^h\,\vec{a}\cdot\vec{\mathbf{n}} - \vec{\mathbf{n}}\cdot\nabla\boldsymbol{\lambda}^h\;,\\ \rho^*_{\Gamma_N}(\boldsymbol{\lambda}^h) &= 0\;.\end{aligned}$$

  Due to the boundary condition of the continuous adjoint system, we have that $\mathbf{r}^*_{\Gamma_N}(\boldsymbol{\lambda}) = 0$. Thus (5.84) is adjoint consistent on the outflow boundary.

- On the Dirichlet boundary Γ_D:

$$\mathbf{r}^*_{\Gamma_D}(\boldsymbol{\lambda}^h) = -\phi\,\boldsymbol{\lambda}^h\,, \tag{5.86a}$$
$$\rho^*_{\Gamma_D}(\boldsymbol{\lambda}^h) = (\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}}+\boldsymbol{\lambda}^h)\,\vec{\mathbf{n}}\,. \tag{5.86b}$$

  While these residuals do not cancel immediately when evaluated at the exact adjoint solution ($\phi > 0$), they can be made consistent through a change in the target functional $\mathcal{J}$ [28]. Let

$$\widetilde{\mathcal{J}}_h(\mathbf{u}^h) := \mathcal{J}_h(\mathbf{u}^h) - \int_{\Gamma_D}\phi\,(\mathbf{u}^h-\mathbf{g}^h_D)\,(\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}})\,\mathrm{d}\mathbf{s} \tag{5.87}$$

  be a consistent modification of $\mathcal{J}^h$, since $\widetilde{\mathcal{J}}(\mathbf{u}) = \mathcal{J}(\mathbf{u})$. The variation of the modified cost functional (5.87) is

$$\widetilde{\mathcal{J}}'_h[\mathbf{u}^h](\mathbf{w}^h) = \mathcal{J}'_h[\mathbf{u}^h](\mathbf{w}^h) + \left\langle\phi\,\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}},\,\mathbf{w}^h\right\rangle_{\Gamma_D} + \left\langle\phi\,(\mathbf{u}^h-\mathbf{g}^h_D),\,\nabla\mathbf{w}^h\cdot\vec{\mathbf{n}}\right\rangle_{\Gamma_D}\,.$$

  All the discrete adjoint residuals remain unchanged, except for (5.86)–(5.86), which now become:

$$\mathbf{r}^*_{\Gamma_D}(\boldsymbol{\lambda}^h) = -\phi\,\left(\boldsymbol{\lambda}^h+\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}}\right)\,, \tag{5.88a}$$
$$\rho^*_{\Gamma_D}(\boldsymbol{\lambda}^h) = (\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}}+\boldsymbol{\lambda}^h)\,\vec{\mathbf{n}}+\phi\,(\mathbf{u}^h-\mathbf{g}^h_D)\,\vec{\mathbf{n}}\,. \tag{5.88b}$$

  Both residuals (5.88)–(5.88) are now identically zero when evaluated at the exact adjoint solution $\boldsymbol{\lambda}$. We have thus proved dual consistency for the DG discretization (5.83) coupled with the modified functional (5.87).

Discrete adjoint boundary sensitivities Now let us consider the boundary bilinear form $\mathcal{B}(\mathbf{g}^h,\mathbf{v}^h)$. For any admissible perturbation $\delta\mathbf{g} := (\delta\mathbf{g}_D,\,\delta\mathbf{g}_N)$ in the boundary conditions, we get that:

$$\begin{aligned}\mathcal{B}(\delta\mathbf{g}^h,\boldsymbol{\lambda}^h) = {}& \left\langle\delta\mathbf{g}_D\,,\left(C^{\mathrm{adj}}_{\Gamma_D}\right)^h\boldsymbol{\lambda}^h\right\rangle_{\Gamma_D} + \left\langle\delta\mathbf{g}_N\,,\left(C^{\mathrm{adj}}_{\Gamma_N}\right)^h\boldsymbol{\lambda}^h\right\rangle_{\Gamma_N}\\ & -\phi\,\left\langle\delta\mathbf{g}_D,\,\boldsymbol{\lambda}^h\right\rangle_{\Gamma_D}\,,\end{aligned} \tag{5.89}$$

where $\left(C^{\mathrm{adj}}_{\Gamma_D}\right)^h$ and $\left(C^{\mathrm{adj}}_{\Gamma_N}\right)^h$ are consistent discretizations of the continuous differential operators defined in (5.65). Note the additional boundary penalty term: the nonzero penalty parameter ϕ ensures stability and convergence of the method. However, it also leads to inconsistencies in the adjoint boundary sensitivities, that must be removed by post-processing of the adjoint Runge–Kutta DG implementation. The adjoint Runge–Kutta time integration, albeit consistent, cannot remove the inconsistent term in the sensitivity formula (5.89). Proving dual consistency is a crucial step in the analysis of a dual DG discretization, allowing one to establish whether or not the adjoint variable corresponds to the true gradient of the discretized cost functional $\mathcal{J}^h$. However, if one also seeks derivatives with respect to the boundary values, further investigations pertaining to $\mathcal{B}$ are warranted, that go beyond establishing dual consistency of the primal discretization.

5.5.2 A one-dimensional test problem

The one-dimensional data assimilation problem is formulated as:

$$\min_{\mathbf{u}^0} \mathcal{J}^h \left(\mathbf{u}^0 \right) \ , \tag{5.90}$$

subject to

$$\mathbf{u}_t + 2\,\mathbf{u}_x = f(x) \ , \ x \in \Omega = [-3\pi,\, 3\pi] \ , \ t \in [0, T] \ , \tag{5.91}$$

with $\mathbf{u}^0(x) = \exp(-|x|)\sin(x)\cos(x)$, periodic boundary conditions, and $T = 0.5$. Equation (5.91) is discretized with a upwind discontinuous Galerkin method, coupled with a fourth order TVD Runge-Kutta scheme for time marching [155]. The space-time dual consistency for this discretization can be easily proven (see, e.g., [28] for the spatial discretization).

The discrete cost functional reads:

$$\begin{aligned} \mathcal{J}_h = \mathcal{J}_B^h + \mathcal{J}_O^h \ &= \ \frac{1}{2} \left(\mathbf{u}^0 - \mathbf{u}^B \right)^T B^{-1} \left(\mathbf{u}^0 - \mathbf{u}^B \right) \\ &+ \frac{1}{2} \sum_{k=1}^{K} \left(\mathcal{H}_k\, \mathbf{u}^k - y^k \right)^T R_k^{-1} \left(\mathcal{H}_k\, \mathbf{u}^k - y^k \right) \ . \end{aligned} \tag{5.92}$$

The background term $\mathcal{J}_B^h$ quantifies the departure of the inverse solution from a background state $\mathbf{u}_B^{h,0}$. It also acts as a regularization term that makes the inverse problem well-posed. $\mathcal{J}_O^h$ quantifies the mismatch between the model predictions and a set of *a priori* available observations y^k at selected grid locations and time points (see Figure 5.1). B and R_k are the error covariance matrices for the background state $\mathbf{u}_B^{h,0}$, and the observation values at t_k, respectively. Finally, $\mathcal{H}_k$ is a linear observation operator that maps the discrete model state to the observation space.

The setup is that of a *twin* experiment: the observations are recorded during a reference run of the model, starting from the reference solution $\mathbf{u}_{\text{ref}}^{h,0}$. The background state $\mathbf{u}_B^{h,0} = 1.4\,\mathbf{u}_{\text{ref}}^{h,0}$ is the initial guess for the optimization routine. We use a 5th order discontinuous basis for the reference run, whereas the inversion is done using only cubic approximations, to avoid an *inverse crime* [160]. The goal of the inversion process is to retrieve a good approximation to the reference initial condition as the *a posteriori* analysis state: $\mathbf{u}_A^{h,0} \approx \mathbf{u}_{\text{ref}}^{h,0}$.

Numerical results

The data assimilation procedure follows the description in [72]. The observation times are $t^1 = 0.25$, and $t^2 = 0.5$ (figure 5.1). Our tests were run with a C++ implementation [161] of the limited memory BFGS method by Nocedal et al. [162, 139]. Figure 5.2 shows the results of the assimilation experiment. It can be seen that the discrete adjoint gradient leads to a considerable decrease in the cost function value, as well as in the RMS error of the computed

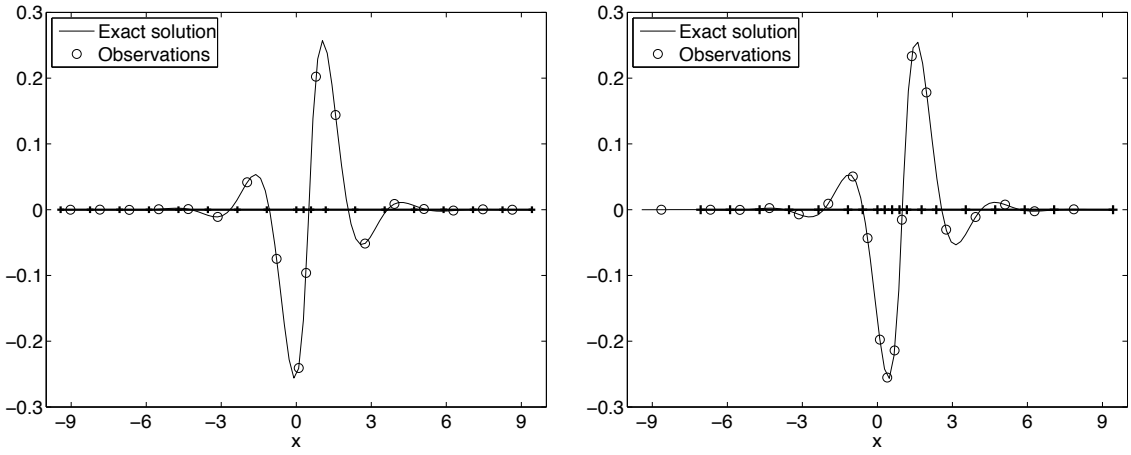


Figure 5.1: Reference observations (circles), and the exact solution (continuous line) of (5.91) at $t^1 = 0.25$ (left) and $t^2 = 0.5$ (right). The adaptive spatial mesh is marked on the x-axis. It is locally refined in areas of high variations in the primal solution $\mathbf{u}^{h,n}$. The refinement is done using an element-wise error estimator based on a finite-difference approximation to the solution gradient $\mathbf{u}_x$.

a posteriori analysis $\mathbf{u}_A^{h,0}$. We computed the analysis root-mean square error (RMSE) ϵ using the formula:

$$\epsilon := \frac{\left\| \mathbf{u}_A^{h,0} - \mathbf{u}_{\text{ref}}^{h,0} \right\|_{L^2(\Omega)}}{\left\| \mathbf{u}_{\text{ref}}^{h,0} \right\|_{L^2(\Omega)}} . \tag{5.93}$$

Figure 5.3 shows that the quality of the analysis obtained through the 4D-Var process is much better than that of the initial guess (the background $\mathbf{u}_B^{h,0}$).

5.5.3 A two-dimensional inverse problem

Problem description

The second test problem is built around the two-dimensional advection equation:

$$\begin{aligned} \mathbf{u}_t + \nabla \cdot (\vec{\beta}\mathbf{u}) &= \mathbf{f} \\ \mathbf{u}(t,\mathbf{x})|_{\Gamma_{\text{in}}} &= \mathbf{g} \\ \mathbf{u}(t^0,\mathbf{x}) &= \mathbf{u}_0(\mathbf{x}) \ , \ t^0 \le t \le t^N \ , \ \mathbf{x} \in \Omega \ . \end{aligned} \tag{5.94}$$

Here $\Omega = [0,1]^2$, $\vec{\beta} := \mathbf{x} \,/\, \|\mathbf{x}\|$, and $\Gamma_{\text{in}} := \left\{ \mathbf{x} \in \Gamma \mid \vec{\mathbf{n}} \cdot \vec{\beta} < 0 \right\}$.

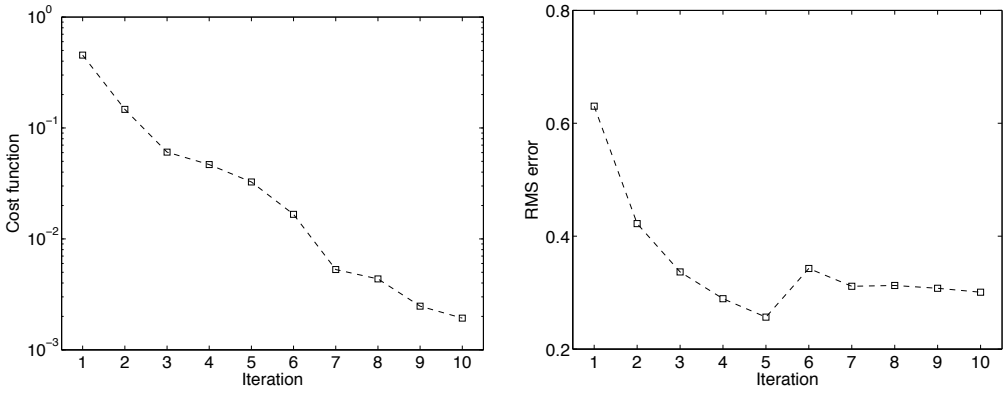


Figure 5.2: Relative decrease (i.e. ratio to the background values) of the cost function $\mathcal{J}^h$ (left), and of the RMSE (right).

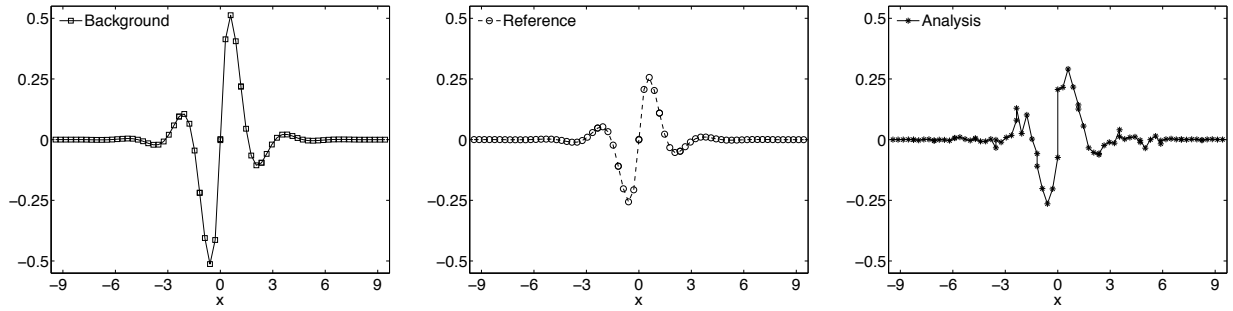


Figure 5.3: Background, reference, and analysis states at $t = 0$ for the problem (5.90)–(5.91).

The discrete cost functional $\mathcal{J}_h$ is defined by (5.92). The particular choice of observation mesh for this experiment is shown in figure 5.4(a).

The experimental setup

The primal and adjoint RK-DG discretizations are implemented with the DEAL.II library [163]. The optimization routine is a C++ implementation [161] of the well-known L-BFGS-B algorithm [139]. The mesh adaptation is driven by an error estimation mechanism based on a numerical approximation of the gradient $\dfrac{\partial \mathbf{u}^h}{\partial \mathbf{x}^h}$ [147]. The optimization mesh Ω_0^h holds the inversion variables throughout the optimization process. It is shown in figure 5.4(b) to be locally refined in regions of high variation in the background state. The final time in the forward simulation is $T = 0.48$, while the observation times are $t^k = 0.03 \times k$, $k = 1 \ldots 16$.

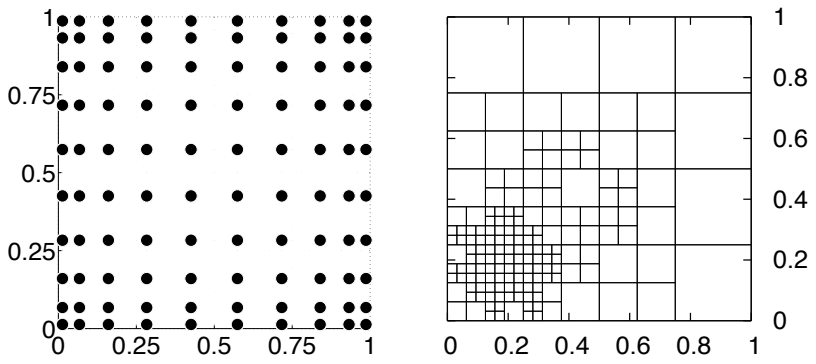


Figure 5.4: (a) Observation grid for the two-dimensional assimilation problem. (b) Optimization grid Ω_0^h that holds the parameters $\mathbf{q}^h = \mathbf{u}^{h,0}$ throughout the inversion process.

Space-time consistency and accuracy of the discrete adjoint solution

To check the consistency and the empirical order of accuracy of the discrete adjoint solver, we will derive the corresponding continuous adjoint problem. Let $\delta(t)$ denote the Dirac delta distribution. We can rewrite the 4D-Var discrete cost functional as

$$\mathcal{J}^h \;:=\; \sum_{k=0}^{K} \widehat{\mathcal{J}}_h(\mathbf{u}^{h,k}) \;:=\; \int_{t^0}^{t^N} \widehat{\mathcal{J}}_h(\mathbf{u}^h, t) \sum_{k=0}^{K} \delta(t - t^k)\, \mathrm{d}t \;. \tag{5.95}$$

Then, the strong form adjoint of (5.94)–(5.95) reads:

$$\begin{aligned} -\boldsymbol{\lambda}_t - \vec{\beta} \cdot (\nabla \boldsymbol{\lambda}) &= \frac{\partial \widehat{\mathcal{J}}}{\partial \mathbf{u}} \sum_{k=0}^{K} \delta(t - t^k) \;, && \mathbf{x} \in \Omega \,,\; t \in [t^N,\, 0] \\ B_\Gamma^{\mathrm{adj}} \boldsymbol{\lambda} := \vec{\beta} \cdot \vec{\mathbf{n}}\, \boldsymbol{\lambda} &= 0 \;, && \mathbf{x} \in \Gamma_{\mathrm{out}} = \Gamma \setminus \Gamma_{\mathrm{in}} \\ \boldsymbol{\lambda}(t^N, \mathbf{x}) &= 0 \;. \end{aligned}$$

The exact solution to the inverse problem is chosen to be

$$\mathbf{u}^0(x, y) = A \exp\left(-\frac{(xs - x_c)^2}{\sigma^2} \right) \exp\left(-\frac{(ys - y_c)^2}{\sigma^2} \right) , \tag{5.96}$$

where $A = 10$, $s = 20$, $\sigma = 2$, and $x_c = y_c = 4$.

We use the dual consistent upwind spatial discretization given in [28]. The particular form of the cost functional (5.92) implies that the discrete adjoint system has a forcing term only at the observation times t^k (5.96), where it is necessary to add the observation mismatch [72]:

$$\boldsymbol{\lambda}^{h,k} = \boldsymbol{\lambda}^{h,k} + \mathcal{H}_k^T \mathcal{R}^{-1} \left(\mathcal{H}_k \mathbf{u}^{h,k} - \mathbf{y}^{h,k} \right) \;,\; k = 1 \,\ldots\, K \;. \tag{5.97}$$

The equation (5.96) is not in conservation form. We rewrite the nonconservative term as

$$\vec{\beta} \cdot \nabla \mathbf{u} = \nabla \cdot (\vec{\beta}\mathbf{u}) - (\nabla \cdot \vec{\beta})\, \mathbf{u} \ ,$$

Through a Galerkin projection onto the discrete function space $\mathcal{U}_h$ and integration by parts, we arrive at the DG semi-discretization of (5.96):

$$\begin{aligned} &\text{For all } n, \text{ find } \bar{\boldsymbol{\lambda}}^{h,n} \text{ such that on } \Omega_n^h \ , \ \forall \bar{\mathbf{w}}^h \in \mathcal{U}_h \ : \\ &- \left\langle \frac{\partial \bar{\boldsymbol{\lambda}}^{h,n}}{\partial t}, \bar{\mathbf{w}}^h \right\rangle_{\Omega} + \sum_{D^k} \left(\left\langle \bar{\boldsymbol{\lambda}}^{h,n+1}, \vec{\beta} \cdot \nabla \bar{\mathbf{w}}^h \right\rangle_{D_n^k} - \left\langle \vec{\beta} \cdot \vec{\mathbf{n}}\, \bar{\boldsymbol{\lambda}}_-^{h,n+1}, \bar{\mathbf{w}}_+^h \right\rangle_{\partial D_{n+}^k} \right. \\ &\left. - \left\langle \vec{\beta} \cdot \vec{\mathbf{n}}\, \bar{\boldsymbol{\lambda}}_-^{h,n+1}, \bar{\mathbf{w}}_+^h \right\rangle_{\partial D_{n-}^k \setminus \Gamma_n^h} + \left\langle \nabla \cdot \vec{\beta}\, \bar{\boldsymbol{\lambda}}^{h,n+1}, \bar{\mathbf{w}}^h \right\rangle_{D_n^k} \right) = 0 \ . \end{aligned} \tag{5.98}$$

Again, at the observation times t^k, we add the mismatch term in (5.97) to the solution $\bar{\boldsymbol{\lambda}}^{h,n}$.

The dual consistency of the spatial discretization, together with a third order strong stability preserving Runge-Kutta method [164] for time integration, ensure space-time dual consistency of our RK-DG discretization for (5.94). Moreover, the spatial and temporal order of accuracy of the discrete adjoint solution match that of the corresponding discretization of the continuous model, i.e., in the limit of both the discretizations we have that:

$$\lim_{\Delta t^n, h \to 0} \frac{\left\| \boldsymbol{\lambda}^{h,n} - \boldsymbol{\lambda}(t^n) \right\|_{L^2(\Omega_n^h)}}{\left\| \bar{\boldsymbol{\lambda}}^{h,n} - \boldsymbol{\lambda}(t^n) \right\|_{L^2(\Omega_n^h)}} = \mathcal{O}(1) \ , \forall\, n = 1 \ldots N \ . \tag{5.99}$$

To verify this numerically, both the discrete adjoint, and the discretization of the adjoint problem (5.98) are compared against a predetermined exact solution

$$\boldsymbol{\lambda}(t, x, y) = \mathbf{u}^0(x - t,\, y - t) \ . \tag{5.100}$$

Here $\mathcal{U}_h$ are broken spaces of piecewise quadratic Lagrange polynomials. For the primal problem, the volume and boundary forcing terms $\mathbf{f}$ and $\mathbf{g}$ are chosen such that

$$\mathbf{u}(t, x, y) = \mathbf{u}^0(x - t,\, y - t) \ . \tag{5.101}$$

Figure 5.5 shows the order of convergence of the RK-DG discretizations on fixed spatial meshes. To illustrate the behavior of the discretization on adaptive meshes, we run the accuracy experiments on a variable mesh and report the numerical results in Figure 5.7. All of the numerical results fully confirm our theoretical derivations: both adjoint solutions are third order accurate in space and time. Hence, the adjoint of the primal discretization inherits the order of accuracy of the discrete forward solution.

Another approach to adjoint code validation is through a truncated Taylor expansion [71]:

$$\mathcal{J}^h(\mathbf{q}^h + \varepsilon\, \delta\mathbf{q}^h) = \mathcal{J}^h(\mathbf{q}^h) + \varepsilon\, \left\langle \boldsymbol{\lambda}^h \, , \, \delta\mathbf{q}^h \right\rangle_{\Omega^h} + \mathcal{O}(\varepsilon^2\, \|\delta\mathbf{q}^h\|^2) \ .$$

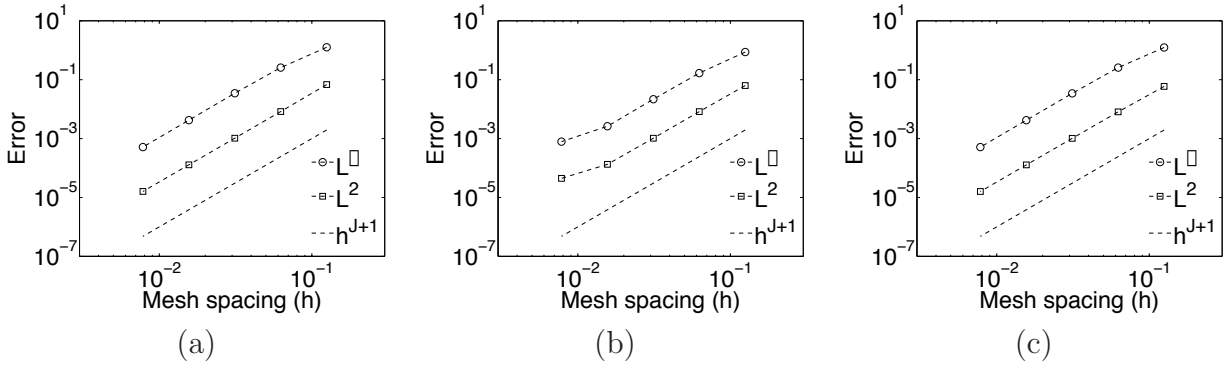


Figure 5.5: Time-averaged L^2 and L^∞ errors for: (a) the forward state $\mathbf{u}^{h,n}$, (b) the continuous adjoint solution $\bar{\boldsymbol{\lambda}}^{h,n}$, and (c) the discrete adjoint variables $\boldsymbol{\lambda}^{h,n}$. The exact solutions are given by (5.96), (5.100), and (5.101). We use a quadratic Lagrange basis over an uniform mesh. The time integration is performed with a third order fixed-step TVD Runge-Kutta method: $\tau^{n+1} = \tau,\ \forall n = 0 \ldots N-1$. The convergence order is $\mathcal{O}(h^3 + \tau^3)$ for all numerical approximations.

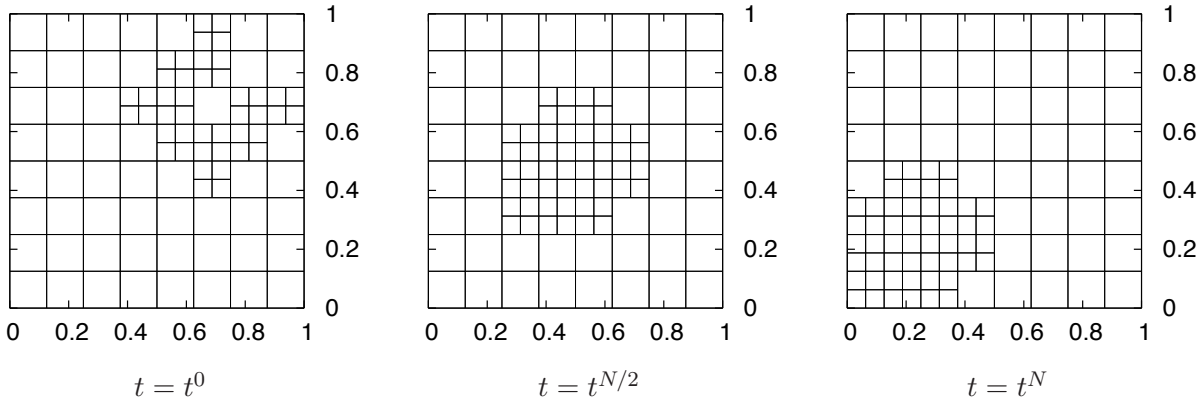


Figure 5.6: Adaptive spatial meshes used in the determination of the numerical order of accuracy for $\bar{\boldsymbol{\lambda}}^{h,n}$, and $\mathbf{u}^{h,n}$.

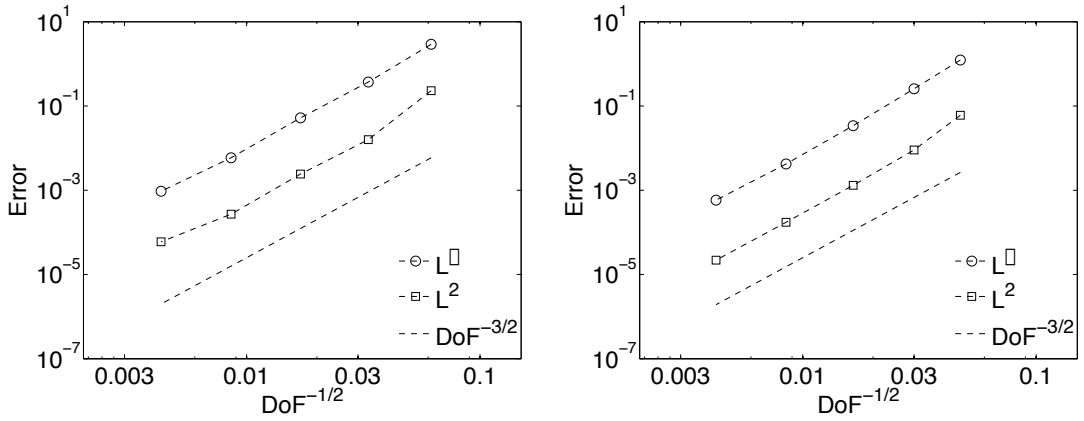


Figure 5.7: Time-averaged L^2 and L^∞ errors plotted against the mesh degrees of freedom for $\bar{\boldsymbol{\lambda}}^{h,n}$ (left), and $\boldsymbol{\lambda}^{h,n}$ (right). The exact solutions are given by (5.96), (5.100), and (5.101). We use a quadratic Lagrange basis over an adaptive spatial mesh, and a third order Runge–Kutta method for the time integration. The cubic convergence confirms the theoretical estimates for the adaptive DG discretization.

Hence, we numerically verify that the following limit holds for small values of ε:

$$\lim_{\varepsilon \to 0} R^h := \lim_{\varepsilon \to 0} \frac{\mathcal{J}^h(\mathbf{q}^h + \varepsilon\, \delta\mathbf{q}^h) - \mathcal{J}^h(\mathbf{q}^h)}{\varepsilon\, \langle \boldsymbol{\lambda}^h\,,\, \delta\mathbf{q}^h \rangle_{\Omega^h}} = 1\ . \tag{5.102}$$

As shown in figure 5.8, the variable mesh discrete adjoint solution is found numerically consistent. For $\varepsilon < 10^{-12}$, truncation errors degrade the quality of the approximation (5.102).

Numerical results

The numerical results for the two-dimensional data assimilation experiment are shown in Figures 5.9 and 5.10. Figure 5.9 (a)–(b) shows the background (the *a priori* state), and the reference solution. Parts (c) and (d) of the same figure illustrates the analysis state, and the analysis error, respectively. It is apparent that the quality of the solution approximation is improved significantly over that of the initial guess (the background).

Figure 5.10 quantifies these improvements. We plot both the decrease in the cost functional throughout the optimization procedure, relative to the initial value at the background state $\mathcal{J}_h(\mathbf{u}_B^{0,h}) := \mathcal{J}^B$. The third order accurate primal and dual solutions lead to a good reconstruction of the optimal solution in our *twin* experiment. The robustness and accuracy of the inversion procedure are tested with various levels of uniformly distributed noise in the observation values. As expected, the quality of the analysis solution degrades when the noise level is increased. However, as Figure 5.10 shows, we still get a significant decrease in

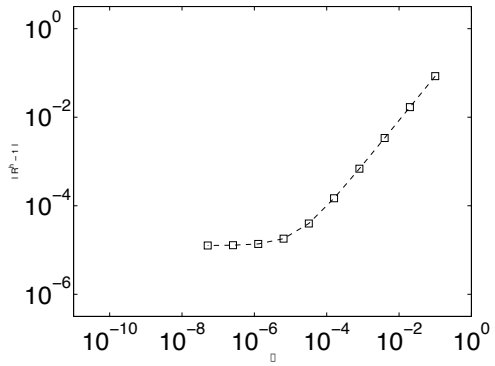


Figure 5.8: Numerical validation of the discrete adjoint solution using equation (5.102).

both the cost functional $\mathcal{J}^h$, and in the analysis error. This indicates good performance and robustness for the discrete adjoint-based adaptive inversion procedure.

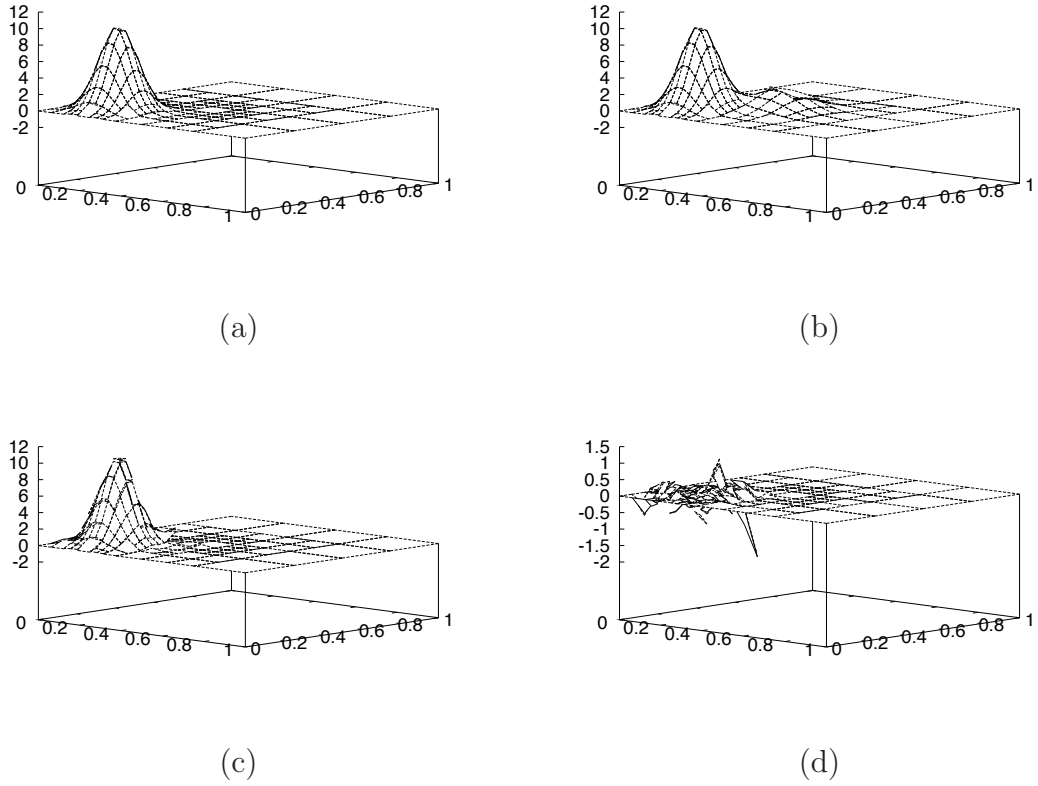


Figure 5.9: Reference (a), background (b), and analysis (c) states for the two-dimensional data assimilation problem, with a measurement noise level of 5%. The analysis error is shown in (d).

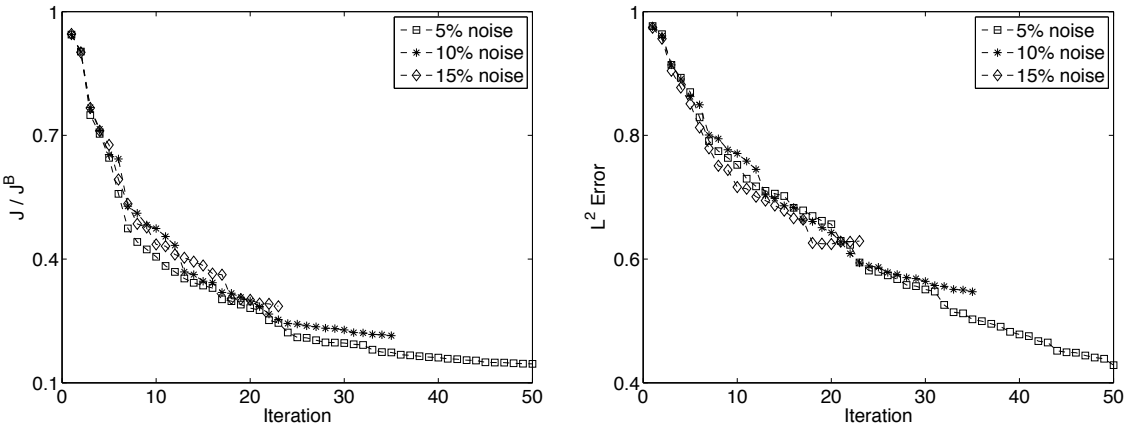


Figure 5.10: Relative decrease in the cost functional, and in the L^2-error for the two-dimensional data assimilation experiments, plotted against the number of optimization iterations. Various observation noise levels are shown.

Chapter 6

Consistency analysis and error estimation

Introduction

This chapter considers parameter identification problems. Unlike model calibration problems [111], where the goal is to minimize a physically significant cost functional, such as lift or drag in aerodynamics, parameter identification problems are directly concerned with determining the optimal values of the parameters. The framework for solving model calibration problems using a dual approach is well established [34]; that for parameter identification problems, much less so. This work focuses on the discrete adjoint approach to solving parameter identification problems. The starting point are the first order necessary conditions for a local minimum of the continuous and discrete optimization problems given in Chapter 1. We then extend the dual consistency framework given in [28] to the discrete optimality condition. This equation, part of the discrete KKT set, is obtained by linearization of the primal model along the inverse variables. Consistency and stability are both essential for convergence of the discrete optimal solution to its continuous counterpart. Similar to dual consistency, consistency of the discrete optimality equation does not hold for all discretizations. However, the analysis shows that consistent modifications [28] to the target functional, may be used to restore stability and consistency of the linearized discretization. The consistency analysis results are confirmed by the numerical experiments.

The second major contribution of this chapter concerns error analysis and estimation for adaptive mesh refinement in parameter identification problems. Energy norm estimates for the control error have been derived for a general class of such problems [42, 43]. However, such estimates are of very limited use in practice, since they rely on problem dependent stability constants, and coercivity estimates for the saddle–point problems stemming from the KKT equations [35]. Another approach is the one proposed by Becker and Vexler in [110]. The error estimates are constructed using the dual–weighted residual approach [35],

and apply to a modified problem defined in terms of an error functional. This error functional may be chosen to be a weighted average of the optimal solution. The authors of [110] assume that the parameter space is finite dimensional. We follow [110], and generalize their approach to the case of an infinite dimensional control space. For high computational efficiency, the exact Hessian of the modified problem is replaced by a quasi-Newton approximation.

This chapter will frequently use the notation, and reference the equations from Chapter 1.

6.1 The model problem

Let $\Omega \subset \mathbb{R}^d$ be a closed convex polyhedral domain with $d \in \{2, 3\}$. Γ is the boundary of Ω. Consider the following elliptic boundary-value problem, henceforth referred to as the *primal model*:

$$\begin{aligned} -\nabla \cdot (\mathbf{q}\, \nabla \mathbf{u}) &= \mathbf{f}\ ,\ \mathbf{x} \in \Omega\ , \\ \mathbf{u} &= \mathbf{g}\ ,\ \mathbf{x} \in \Gamma\ . \end{aligned} \tag{6.1}$$

Smoothness assumptions Let the volume forcing $\mathbf{f} \in \mathcal{L}^2(\Omega)$, and the Dirichlet boundary data $\mathbf{g} \in \mathcal{H}^{3/2}(\Gamma)$. Also, assume the inversion variables $\mathbf{q} \in \mathcal{H}^2(\Omega)$. The formulation of the primal problem requires that $\mathbf{q} \geq 0$, *a.e.* on Ω. Also, assume that any additional conditions on the domain boundary Γ [165] are satisfied, such that the exact primal solution $\mathbf{u} \in \mathcal{U} \subset \mathcal{H}^2(\Omega)$.

The smoothness of the solution space $\mathcal{U}$ guarantees existence and boundedness of the following trace operators for any function $\mathbf{v} \in \mathcal{U}$ [166]:

$$\begin{aligned} &\gamma^0 : \mathcal{H}^2(\Omega) \to \mathcal{H}^{3/2}(\Gamma) \\ &\gamma^0(\mathbf{v}) = \mathbf{v}|_\Gamma\ , \end{aligned}$$

and

$$\begin{aligned} &\gamma^1 : \mathcal{H}^2(\Omega) \to \mathcal{H}^{1/2}(\Gamma) \\ &\gamma^1(\mathbf{v}) = \nabla \mathbf{v} \cdot \vec{\mathbf{n}}|_\Gamma\ . \end{aligned}$$

Inner products of functions in $\mathcal{U}$ on Γ and Ω are defined as follows:

$$\langle \mathbf{u},\, \mathbf{v} \rangle_\Omega := \int_\Omega \mathbf{u}\, \mathbf{v}\, \mathrm{d}\mathbf{x} \qquad , \qquad \langle \mathbf{u},\, \mathbf{v} \rangle_\Gamma := \int_\Gamma \mathbf{u}\, \mathbf{v}\, \mathrm{d}\mathbf{s}\ ,\ \forall\ \mathbf{u},\, \mathbf{v} \in \mathcal{U}\ .$$

They induce the corresponding norms on $\mathcal{U}$:

$$\begin{aligned} \|\mathbf{u}\|_{\mathcal{L}^2(\Omega)} &:= \sqrt{\langle \mathbf{u},\, \mathbf{u} \rangle_\Omega}\ , \\ \|\mathbf{u}\|_{\mathcal{L}^2(\Gamma)} &:= \sqrt{\langle \mathbf{u},\, \mathbf{u} \rangle_\Gamma}\ ,\ \forall\ \mathbf{u} \in \mathcal{U}\ . \end{aligned}$$

Inner products on the space $\mathcal{Q}$, and on the product space $\mathcal{U} \times \mathcal{Q}$, are defined in a similar fashion.

The target functional reads:

$$\mathcal{J}[\mathbf{u}, \mathbf{q}] \;=\; \frac{1}{2}\|\mathcal{H}\mathbf{u} - \mathbf{o}\|^2_{\mathcal{L}^2(\Omega)} + \frac{1}{2}\|\nabla(\mathbf{q} - \mathbf{q}_B)\|^2_{\mathcal{L}^2(\Omega)} + \frac{\boldsymbol{\beta}}{2}\|\mathbf{q} - \mathbf{q}_B\|^2_{\mathcal{L}^2(\Omega)} \; .$$

The background control $\mathbf{q}_B \in \mathcal{Q}$ is positive *a.e.* on Ω. $\mathcal{H} : \mathcal{U} \to \mathcal{O} \subset \mathcal{L}^2(\Omega)$ is a linear observation operator. The exact choice of regularization parameter $\boldsymbol{\beta} > 0$ depends on the discretization, as well as on the properties of primal model and cost functional, and is the topic of current research: see, e.g., [167, 168, 169], and references therein. We will assume a constant regularization parameter.

With this choice of model and cost functional, the inverse problem (1.1) has the following form:

$$\text{Find } \mathbf{q}_* = \arg\min_{\mathbf{q}\in\mathcal{Q}} \mathcal{J}[\mathbf{u}, \mathbf{q}] \text{ , subject to (6.1) .} \tag{6.2}$$

Note that there are no explicit bound constraints on the inversion variables $\mathbf{q}$. Rather, we enforce positivity of the diffusion coefficient indirectly, by solving 6.2) in a sufficiently small neighborhood of the reference profile. Another approach to guarantee positivity is to explicitly enforce the bounds for the discrete parameter values in the optimization routine.

6.1.1 The continuous KKT system for the model problem

In what follows, adjoint operators will be denoted by a $*$ superscript. Assume no boundary terms arise in the definition of the adjoint operator $\mathcal{H}^*$ (compatibility condition 1 in [24]):

$$\langle \mathcal{H}\mathbf{u},\, \mathbf{o}\rangle_\Omega = \langle \mathbf{u}, \mathcal{H}^*\mathbf{o}\rangle_\Omega \;\; , \; \forall \; \mathbf{u} \in \mathcal{U},\, \mathbf{o} \in \mathcal{O} \; .$$

With this, the adjoint equation (1.2b) reads:

$$\begin{aligned} -\nabla \cdot (\mathbf{q}\,\nabla\boldsymbol{\lambda}) \;&=\; \mathcal{H}^*(\mathcal{H}\mathbf{u} - \mathbf{o}) \; , \; \mathbf{x} \in \Omega \\ \boldsymbol{\lambda} \;&=\; 0 \; , \; \mathbf{x} \in \Gamma \; . \end{aligned} \tag{6.3}$$

Remark With the smoothness assumptions on $\mathbf{q}$, $\mathbf{o}$, and $\mathbf{u}$, the exact dual solution is $\boldsymbol{\lambda} \in \mathcal{H}^2(\Omega)$.

For all test functions $\mathbf{z} \in \mathcal{Q}$ in the inversion variables $\mathbf{q}$, equation (1.2c) reads:

$$\langle \nabla\mathbf{q},\, \nabla\mathbf{z}\rangle_\Omega - \langle \nabla\mathbf{q}_B,\, \nabla\mathbf{z}\rangle_\Omega + \boldsymbol{\beta}\; \langle \mathbf{q} - \mathbf{q}_B,\, \mathbf{z}\rangle_\Omega + \langle \nabla\cdot(\mathbf{z}\,\nabla\mathbf{u}),\, \boldsymbol{\lambda}\rangle_\Omega = 0 \; .$$

The strong form of the optimality equation is then obtained through an integration by parts:

$$\begin{aligned} -\Delta\,\mathbf{q} + \boldsymbol{\beta}\,(\mathbf{q} - \mathbf{q}_B) &= -\Delta\mathbf{q}_B + \nabla\mathbf{u}\cdot\nabla\boldsymbol{\lambda}\;,\; \mathbf{x}\in\Omega\;, \\ \nabla\mathbf{q}\cdot\vec{\mathbf{n}} &= \nabla\mathbf{q}_B\cdot\vec{\mathbf{n}}\;,\; \mathbf{x}\in\Gamma\;. \end{aligned} \tag{6.4}$$

Remark Using the strong maximum principle [166], this Helmholtz boundary value problem can be shown to be well posed. Since the volume forcing $\left(-\Delta\mathbf{q}_B + \nabla\mathbf{u}\cdot\nabla\boldsymbol{\lambda}\right) \in \mathcal{L}^2(\Omega)$, and the Neumann boundary data $\nabla\mathbf{q}_B\cdot\vec{\mathbf{n}} \in \mathcal{H}^{1/2}(\Gamma)$, the exact solution $\mathbf{q}_* \in \mathcal{H}^2(\Omega)$. By the Sobolev imbedding theorem [166], $\mathbf{q}_*$ is bounded and continuous on Ω for $d \in \{2,\,3\}$. Furthermore, for sufficiently large $\mathbf{q}_B$, the solution $\mathbf{q}_* > 0$ *a.e.* on Ω.

6.1.2 The reduced gradient

This section illustrates in detail the derivation of the gradient of the reduced cost functional (1.4). Consider an arbitrary function $\boldsymbol{\delta}\mathbf{q} \in \mathcal{Q}$, and let $\boldsymbol{\delta}\mathbf{u} := \dfrac{\partial\mathbf{u}}{\partial\mathbf{q}}[\mathbf{q}](\boldsymbol{\delta}\mathbf{q})$, with $\boldsymbol{\delta}\mathbf{u} \in \mathcal{U}$. The reduced gradient $\nabla_{\mathbf{q}}\mathcal{J}$ is defined through identification from the following equality:

$$\langle \nabla_{\mathbf{q}}\mathcal{J},\,\boldsymbol{\delta}\mathbf{q}\rangle_\Omega := \frac{\partial\mathcal{J}}{\partial\mathbf{q}}[\mathbf{u},\mathbf{q}](\boldsymbol{\delta}\mathbf{q}) + \frac{\partial\mathcal{J}}{\partial\mathbf{u}}[\mathbf{u},\mathbf{q}](\boldsymbol{\delta}\mathbf{u})\;,\;\forall\,\boldsymbol{\delta}\mathbf{q}\in\mathcal{Q}\;. \tag{6.5}$$

We make use of the tangent linear equation, which is obtained from the primal model by differentiation in the direction $(\boldsymbol{\delta}\mathbf{u},\boldsymbol{\delta}\mathbf{q}) \in \mathcal{U}\times\mathcal{Q}$:

$$\begin{aligned} -\nabla\cdot(\boldsymbol{\delta}\mathbf{q}\,\nabla\mathbf{u}) - \nabla\cdot(\mathbf{q}\,\nabla\,\boldsymbol{\delta}\mathbf{u}) &= 0\;,\;\mathbf{x}\in\Omega \\ \boldsymbol{\delta}\mathbf{u} &= 0\;,\;\mathbf{x}\in\Gamma\;. \end{aligned} \tag{6.6}$$

Using integration by parts, the adjoint equation (6.3), and the tangent linear model (6.7), the reduced gradient formula becomes:

$$\begin{aligned} \langle \nabla_{\mathbf{q}}\mathcal{J},\,\boldsymbol{\delta}\mathbf{q}\rangle_\Omega &= \int_\Omega \mathcal{H}^*(\mathcal{H}\mathbf{u}-\mathbf{o})\,\boldsymbol{\delta}\mathbf{u}\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla\mathbf{q}\,\nabla\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{x} \\ &\quad - \int_\Omega \nabla\mathbf{q}_B\,\nabla\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{x} + \boldsymbol{\beta}\int_\Omega (\mathbf{q}-\mathbf{q}_B)\,\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{s} \\ &= -\int_\Omega \nabla\boldsymbol{\lambda}\cdot\nabla\mathbf{u}\,\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{x} - \int_\Omega \Delta\mathbf{q}\,\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{x} + \int_\Omega \Delta\mathbf{q}_B\,\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{x} \\ &\quad + \boldsymbol{\beta}\int_\Omega (\mathbf{q}-\mathbf{q}_B)\;\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{x} + \int_\Gamma \nabla\mathbf{q}\cdot\vec{\mathbf{n}}\,\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{s} - \int_\Gamma \nabla\mathbf{q}_B\cdot\vec{\mathbf{n}}\,\boldsymbol{\delta}\mathbf{q}\,\mathrm{d}\mathbf{s} \\ &:= \left\langle \nabla_{\mathbf{q}}\mathcal{J}|_\Omega\,,\,\boldsymbol{\delta}\mathbf{q}\right\rangle_\Omega + \left\langle \nabla_{\mathbf{q}}\mathcal{J}|_\Gamma\,,\,\boldsymbol{\delta}\mathbf{q}\right\rangle_\Gamma\;. \end{aligned} \tag{6.7}$$

We can immediately see that the gradient is identically zero at the optimal solution $\mathbf{q}_*$ that satisfies (6.4).

6.2 *A priori* error analysis for the discrete optimality system

This section investigates the consistency properties of the optimality system for the discrete counterpart of problem (6.2). The discontinuous Galerkin method [22] is used for the spatial discretization.

6.2.1 Notation and preliminaries for the discrete problems

Assume the discrete domains cover exactly the analytical ones, i.e., $\Omega^h \equiv \Omega$, and $\Gamma^h \equiv \Gamma$. Let $\mathcal{U}_h^p = \text{span}\{\boldsymbol{\chi}_j(\mathbf{x})\}_{j=1\ldots P} \subset \mathcal{U}$ denote (for any integer $p \geq 1$) the discrete solution space consisting on discontinuous piecewise polynomial functions of degree less than or equal to p. Similarly, $\mathcal{Q}_h^r = \text{span}\{\boldsymbol{\chi}_l^q(\mathbf{x})\}_{l=1\ldots R} \subset \mathcal{Q}$ is the space of piecewise polynomials of maximal degree $r \geq 1$.

Following the notation in [28], the domain Ω is divided into shape-regular meshes $\mathcal{T}_h = \bigcup_{j=1}^{J} \kappa_j$, and $\mathcal{T}_h^q = \bigcup_{i=1}^{I} \kappa_i^q$, comprised of polyhedral elements. The primal and dual mesh variables $\mathbf{u}^h$ and $\boldsymbol{\lambda}^h$ are defined on the mesh $\mathcal{T}_h$, while the triangulation $\mathcal{T}_h^q$ holds the discrete inversion variables $\mathbf{q}^h$. Let $\Gamma_{\mathcal{I}}$ and $\Gamma_{\mathcal{I}}^q$ be the union of all distinct interior edges of the $\mathcal{T}_h$, and $\mathcal{T}_h^q$, respectively.

From the definition of $\mathcal{U}_h^p$, the restriction of any mesh function $\mathbf{v}^h$ defined on $\mathcal{T}_h$ to an arbitrary element κ can be written as follows:

$$\mathbf{v}^h(\mathbf{x})\big|_{\kappa} = \sum_{j=1}^{P} v_{\kappa}^j \, \boldsymbol{\chi}_j(\mathbf{x}) \; . \tag{6.8}$$

The coefficients $\{v_{\kappa}^j\}$, $j = 1 \ldots P$, fully determine the numerical solution inside κ. Similarly, for arbitrary mesh functions in $\mathcal{Q}_h^r$ defined on the triangulation $\mathcal{T}_h^q$, one has:

$$\mathbf{z}^h(\mathbf{x})\big|_{\kappa^q} = \sum_{l=1}^{R} z_{\kappa^q}^l \, \boldsymbol{\chi}_l^q(\mathbf{x}) \; , \; \forall \, \mathbf{z}^h \in \mathcal{Q}_h^r \; , \; \forall \, \kappa^q \in \mathcal{T}_h^q \; . \tag{6.9}$$

Let h_{κ} denote the diameter of $\kappa \in \mathcal{T}_h$:

$$h_{\kappa} := \max_{\mathbf{x},\mathbf{y} \in \kappa} \|\mathbf{x} - \mathbf{y}\|_2 \; .$$

The area (or volume) of the element κ is $|\kappa|$, and its boundary is denoted by $\partial\kappa$. Furthermore, $h := \max_{\kappa \in \mathcal{T}_h} h_{\kappa}$, and $h_{\min} := \min_{\kappa \in \mathcal{T}_h} h_{\kappa}$, with the assumption that the ratio $h/h_{\min}$ is a

constant independent of h. Given two neighboring elements κ_- and κ_+ (with a common edge $e = \kappa_- \cap \kappa_+$), we let $\mathbf{u}^h_\pm := \mathbf{u}^h\big|_{\partial\kappa_\pm}$ denote the trace of $\mathbf{u}^h$ on e, taken from the interior of $\kappa_\pm$, respectively.

Let $e = \kappa_+ \cup \kappa_-$ denote an edge (or face) between two neighboring elements κ_+ and κ_-. We denote by $|e|$ the length (or area) of e. Furthermore, the jump in the solution over e is given by:

$$[\![\mathbf{u}^h]\!] := \mathbf{u}^h_+ \vec{\mathbf{n}}_+ + \mathbf{u}^h_- \vec{\mathbf{n}}_- \ ,$$

whereas the average at $\mathbf{x}^h \in \kappa_- \cap \kappa_+$ is

$$\{\mathbf{u}^h\} := \frac{\mathbf{u}^h_- + \mathbf{u}^h_+}{2} \ .$$

On a boundary edge, $\{\mathbf{u}^h\} := \mathbf{u}^h_+$, and $[\![\mathbf{u}]\!] := \mathbf{u}^h_+ \, \vec{\mathbf{n}}_+$.

Assume the mesh $\mathcal{T}_h$ is regular enough such that, for some α_1, $\alpha_2 > 0$,

$$\alpha_1 h_\kappa^{-1} \leq \frac{|e|}{|\kappa|} \leq \alpha_2 h_\kappa^{-1} \ ,$$

from which the classical trace inequalities for polynomial functions on κ [170, 22, 171] follow:

$$\|\mathbf{u}^h\|_{\partial\kappa} \ \leq \ C_1(p) h_\kappa^{-1/2} \|\mathbf{u}^h\|_\kappa \ , \tag{6.10a}$$

$$\|\nabla \mathbf{u}^h \cdot \vec{\mathbf{n}}\|_{\partial\kappa} \ \leq \ C_2(p) h_\kappa^{-1/2} \|\nabla \mathbf{u}^h\|_\kappa \ , \ \forall \ \mathbf{u}^h \in \mathcal{U}^p_h \ . \tag{6.10b}$$

Here the constants $C_1(p)$ and $C_2(p)$ depend only on the polynomial basis order p.

Finally, we define two discrete inner product functionals over the space $\mathcal{U}^p_h$:

$$\begin{aligned} \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_\kappa \ &:= \ \int_\kappa \mathbf{u}^h \, \mathbf{v}^h \, \mathrm{d}\mathbf{x} \ , \ \forall \, \kappa \in \mathcal{T}_h \ , \\ \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_e \ &:= \ \int_e \mathbf{u}^h \, \mathbf{v}^h \, \mathrm{d}\mathbf{s} \ , \ \forall \, e \in \Gamma \cup \Gamma_{\mathcal{I}} \ . \end{aligned}$$

They are valid for any mesh functions $\mathbf{u}^h$, $\mathbf{v}^h \in \mathcal{U}^p_h$. Inner products on Ω and inner/outer boundaries are defined by extension of the definitions above:

$$\begin{aligned} \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_\Omega \ &= \ \sum_{\kappa \in \mathcal{T}_h} \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_\kappa \ , \\ \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_\Gamma \ &= \ \sum_{e \in \Gamma} \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_e \ , \\ \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_{\Gamma_{\mathcal{I}}} \ &= \ \sum_{e \in \Gamma_{\mathcal{I}}} \left\langle \mathbf{u}^h, \, \mathbf{v}^h \right\rangle_e \ , \ \forall \, \mathbf{u}^h, \, \mathbf{v}^h \in \mathcal{U}^p_h \ . \end{aligned}$$

The inner products also induce $\mathcal{L}^2$ norms on mesh elements and edges.

Remark Similar assumptions and definitions hold for mesh functions in $\mathcal{Q}_h^r$ defined on the triangulation $\mathcal{T}_h^q$.

Nested meshes and inner products on $\mathcal{Q}_h^r \times \mathcal{U}_h^p$

To be able to use different meshes for the parameters and primal/dual variables, we make some simplifying assumptions on the structure of the triangulations $\mathcal{T}_h$ and $\mathcal{T}_h^q$ [13]. Specifically, we assume that $\mathcal{T}_h$ can be obtained from $\mathcal{T}_h^q$ by hierarchical mesh refinement. This implies that the basis functions for $\mathcal{Q}_h^r$ can be written as linear combinations of the bases for $\mathcal{U}_h^p$:

$$\chi_l^q = \sum_{i=1}^{P} \mathbf{M}_{li}\, \chi_i \ , \ \forall\, l = 1 \dots R \ .$$

With (6.8) and (6.9), we can define the inner product over $\mathcal{U}_h^p \times \mathcal{Q}_h^r$ for an arbitrary element $\kappa \in \mathcal{T}_h$:

$$\left\langle \mathbf{u}^h,\, \mathbf{z}^h \right\rangle_\kappa \ := \ \int_\kappa \mathbf{u}^h\, \mathbf{z}^h \,\mathrm{d}\mathbf{s} = \sum_{i=1}^{P} \sum_{l=1}^{R} \sum_{j=1}^{P} u_\kappa^i\, z_{\kappa^q}^l\, \mathbf{M}_{lj}\, \left\langle \chi_i,\, \chi_j \right\rangle_\kappa \ .$$

Remark A similar definition holds on any element $\kappa_q \in \mathcal{T}_h^q$, using the reverse mapping $\mathbf{M}^q \in \mathbb{R}^{P \times R}$.

We can extend these definitions to cover the whole discrete domain Ω, boundary Γ, or inner faces $\Gamma_\mathcal{I}$, and $\Gamma_\mathcal{I}^q$, respectively. To do so, it suffices to note that since $\mathcal{T}_h^q \subset \mathcal{T}_h$, the set of inner edges for the parameter mesh $\mathcal{T}_h^q$ is included in that of the primal/dual mesh $\mathcal{T}_h$, i.e.

$$\Gamma_\mathcal{I}^q \subset \Gamma_\mathcal{I} \ . \tag{6.11}$$

Equation (6.11) will also be useful in defining and analyzing the discrete weak formulations discussed in later sections.

6.2.2 The discrete KKT system

Consider the primal problem (6.1). Its SIPG discontinuous Galerkin discretization reads [28]:

$$\begin{aligned} &\text{Find } \mathbf{u}^h \in \mathcal{U}_h^p \text{ such that, for all test functions } \mathbf{w}^h \in \mathcal{U}_h^p, \text{ we have:} \\ &\mathcal{N}^h(\mathbf{u}^h, \mathbf{w}^h) = \int_\Omega \mathbf{f}^h\, \mathbf{w}^h \,\mathrm{d}\mathbf{x} + \mathcal{B}^h(\mathbf{g}^h, \mathbf{w}^h) \ , \ \forall\ \mathbf{w}^h \in \mathcal{U}_h \ , \end{aligned} \tag{6.12}$$

with

$$\begin{aligned}\mathcal{N}^h(\mathbf{u}^h,\mathbf{w}^h) &:= \int_\Omega \mathbf{q}^h\,\nabla\mathbf{u}^h\cdot\nabla\mathbf{w}^h\,\mathrm{d}\mathbf{x}+\int_{\Gamma_\mathcal{I}\cup\Gamma}\phi\,[\![\mathbf{u}^h]\!]\cdot[\![\mathbf{w}^h]\!]\,\mathrm{d}\mathbf{s}\\ &\quad-\int_{\Gamma_\mathcal{I}\cup\Gamma}\left([\![\mathbf{u}^h]\!]\cdot\{\mathbf{q}^h\,\nabla\mathbf{w}^h\}+\{\mathbf{q}^h\,\nabla\mathbf{u}^h\}\cdot[\![\mathbf{w}^h]\!]\right)\,\mathrm{d}\mathbf{s}\,,\\ \mathcal{B}^h(\mathbf{g}^h,\mathbf{w}^h) &:= -\int_\Gamma \mathbf{q}^h\,\mathbf{g}^h\,\nabla\mathbf{w}^h\cdot\vec{\mathbf{n}}\,\mathrm{d}\mathbf{s}+\int_\Gamma\phi\,\mathbf{g}^h\,\mathbf{w}^h\,\mathrm{d}\mathbf{s}\,.\end{aligned}$$

Here $\Gamma_\mathcal{I}$ denotes the union of all interior edges. The penalty parameter is $\phi=h_e^{-1}\,\hat{\phi}$, where, for any edge e, and dimension $d=2,\,3$, we define $h_e(\mathbf{x}):=|e|^{1/(d-1)}$.

We let $\mathcal{H}^s(\mathcal{T}_h)$ denote the broken Sobolev space, for any real number s:

$$\mathcal{H}^s(\mathcal{T}_h):=\left\{\mathbf{v}\in\mathcal{L}^2(\Omega)\,|\,\forall\,\kappa\in\mathcal{T}_h,\mathbf{v}|_\kappa\in\mathcal{H}^s(\kappa)\right\}\,.$$

The norm on this broken space is defined by

$$|\!|\!|\mathbf{v}|\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)}:=\left(\sum_{\kappa\in\mathcal{T}_h}\|\mathbf{v}\|_{\mathcal{H}^s(\kappa)}\right)^{1/2}\,. \tag{6.13}$$

Using the notation above, define the energy norm (or *natural* DG norm) [22, 171]:

$$\|\mathbf{v}\|_{\mathrm{DG}}:=\left(\sum_{\kappa\in\mathcal{T}_h}\int_\kappa\mathbf{q}\,\nabla\mathbf{v}\cdot\nabla\mathbf{v}\,\mathrm{d}\mathbf{x}+\sum_{e\in\Gamma_\mathcal{I}\cup\Gamma}\hat{\phi}\,h^{-1}\int_e[\![\mathbf{v}]\!]\cdot[\![\mathbf{v}]\!]\,\mathrm{d}\mathbf{s}\right)^{1/2}\,,\ \forall\,\mathbf{v}\in\mathcal{U}\,. \tag{6.14}$$

With this, we have the following theorem.

Theorem 6.2.1. *Assume the exact solution to (6.1) belongs to $\mathcal{H}^s(\Omega)$, $s\geq 2$. For sufficiently large penalty parameter $\hat{\phi}>0$, there exists a constant C independent of h such that*

$$\|\mathbf{u}-\mathbf{u}^h\|_{\mathrm{DG}}\leq C\,h^{\min(p+1,s)-1}\,|\!|\!|\mathbf{u}|\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)}\,, \tag{6.15}$$

and

$$\|\mathbf{u}-\mathbf{u}^h\|_{\mathcal{L}^2(\Omega)}\leq C\,h^{\min(p+1,s)}\,|\!|\!|\mathbf{u}|\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)}\,, \tag{6.16}$$

Proof. See [171, Chapter 2]. □

It is now useful to note that (6.14) implies:

$$\sum_{e\in\Gamma_\mathcal{I}}\left\|\hat{\phi}\,h^{-1/2}(\mathbf{u}-\mathbf{u}^h)\right\|_{\mathcal{L}^2(e)}=\left\|\hat{\phi}^{1/2}\,h^{-1/2}[\![\mathbf{u}-\mathbf{u}^h]\!]\right\|_{\mathcal{L}^2(\Gamma_\mathcal{I})}\leq\left\|\mathbf{u}-\mathbf{u}^h\right\|_{\mathrm{DG}}\,, \tag{6.17}$$

and

$$\sum_{e\in\Gamma}\left\|\hat{\phi}\,h^{-1/2}(\mathbf{u}-\mathbf{u}^h)\right\|_{\mathcal{L}^2(e)}=\left\|\hat{\phi}^{1/2}\,h^{-1/2}(\mathbf{u}-\mathbf{u}^h)\right\|_{\mathcal{L}^2(\Gamma)}\leq\left\|\mathbf{u}-\mathbf{u}^h\right\|_{\mathrm{DG}}\ . \tag{6.18}$$

The discrete counterpart of $\mathcal{J}$ reads:

$$\begin{aligned}\mathcal{J}^h(\mathbf{u}^h,\mathbf{q}^h) \;=\; & \frac{1}{2}\int_\Omega(\mathcal{H}^h\mathbf{u}^h-\mathbf{o}^h)^T(\mathcal{H}^h\mathbf{u}^h-\mathbf{o}^h)\,\mathrm{d}\mathbf{x}\\ & +\frac{\boldsymbol{\beta}}{2}\int_\Omega(\mathbf{q}^h-\mathbf{q}_B^h)^T(\mathbf{q}^h-\mathbf{q}_B^h)\,\mathrm{d}\mathbf{x}\\ & +\frac{1}{2}\int_\Omega(\nabla\mathbf{q}^h-\nabla\mathbf{q}_B^h)^T\,(\nabla\mathbf{q}^h-\nabla\mathbf{q}_B^h)\,\mathrm{d}\mathbf{x}\\ & +\mathcal{R}^h[\mathbf{u}^h,\mathbf{q}^h]\ .\end{aligned} \tag{6.19}$$

We have the following definition for $\mathcal{R}^h$ [28]:

Definition 6.2.1 (Consistent modification of the objective functional)**.** *The term $\mathcal{R}^h[\mathbf{u}^h,\mathbf{q}^h]$ indicates a modification to the discrete functional $\mathcal{J}^h$. $\mathcal{R}^h$ must be Fréchet differentiable in both of its arguments. This modification is said to be consistent if $\mathcal{R}^h$ cancels when evaluated at the exact solutions $\mathbf{u}$, $\mathbf{q}$ of (1.2a)–(1.2c):*

$$\mathcal{R}^h[\mathbf{u},\,\mathbf{q}]=0\ .$$

Remark. Any nontrivial consistent modification in (6.19) does impact both the value and the gradient of the discrete cost functional $\mathcal{J}^h$. The modification term $\mathcal{R}^h$ can only be expected to vanish in the limit of the discretization. Both $\mathcal{R}^h$ and its Fréchet derivative are nonzero for fixed values of $h>0$.

Consistent modifications have been shown in [28] to be required for certain inverse problems, to guarantee dual consistency of the primal discretization. This concept is extended here to the optimality equation. We show below that, for the model problem (6.2), consistent modifications to $\mathcal{J}^h$ may introduce the stabilization terms necessary for convergence of the optimality equation discretization 1.8c.

Given (6.19) and (6.12), we can formulate the discrete inverse problem as:

$$\text{Find }\ \mathbf{q}_*^h=\arg\min_{\mathbf{q}^h\in\mathcal{Q}_h^r}\mathcal{J}^h[\mathbf{u}^h,\mathbf{q}^h]\ ,\ \text{subject to (6.12)}\ . \tag{6.20}$$

The discrete Lagrangian functional for (6.20)–(6.12) reads as follows:

$$\mathcal{L}^h(\mathbf{u}^h,\boldsymbol{\lambda}^h,\mathbf{q}^h):=\mathcal{J}^h(\mathbf{u}^h,\mathbf{q}^h)-\mathcal{N}^h(\mathbf{u}^h,\boldsymbol{\lambda}^h)+\left\langle\mathbf{f}^h,\,\boldsymbol{\lambda}^h\right\rangle_\Omega+\mathcal{B}^h(\mathbf{g}^h,\boldsymbol{\lambda}^h)\ . \tag{6.21}$$

Then, the discrete adjoint problem corresponding to (6.19)–(6.12) is:

$$\begin{aligned} &\text{Find } \boldsymbol{\lambda} \in \mathcal{U}_h^p \text{ such that:} \\ &\mathcal{N}^{h\,*}(\mathbf{w}^h, \boldsymbol{\lambda}^h) = \frac{\partial \mathcal{J}^h}{\partial \mathbf{u}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{w}^h) \ , \ \forall\, \mathbf{w}^h \in \mathcal{U}_h \ , \end{aligned} \tag{6.22}$$

with

$$\begin{aligned} \mathcal{N}^{h\,*}(\mathbf{w}^h, \boldsymbol{\lambda}^h) \ := \ & \int_\Omega \mathbf{q}^h \nabla \mathbf{w}^h \cdot \nabla \boldsymbol{\lambda}^h \,\mathrm{d}\mathbf{x} + \int_{\Gamma_{\mathcal{I}} \cup \Gamma} \phi \, [\![\mathbf{w}^h]\!] \cdot [\![\boldsymbol{\lambda}^h]\!] \,\mathrm{d}\mathbf{s} \\ & - \int_{\Gamma_{\mathcal{I}} \cup \Gamma} \left([\![\mathbf{w}^h]\!] \cdot \{ \mathbf{q}^h \nabla \boldsymbol{\lambda}^h \} + \{ \mathbf{q}^h \nabla \mathbf{w}^h \} \cdot [\![\boldsymbol{\lambda}^h]\!] \right) \mathrm{d}\mathbf{s} \ , \end{aligned}$$

and

$$\frac{\partial \mathcal{J}^h}{\partial \mathbf{u}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{w}^h) \ := \ \int_\Omega (\mathcal{H}^h)^T (\mathcal{H}^h \mathbf{u}^h - \mathbf{z}^h)\, \mathbf{w}^h \,\mathrm{d}\mathbf{x} \ .$$

The primal DG discretization is shown in [28] to be dual consistent: its adjoint (6.22) is a consistent discretization of the continuous adjoint PDE (6.3). Given that the strong form of the adjoint equation is also an elliptic problem, and the dual discretization is of SIPG form, the $\mathcal{L}^2$-error bound in Theorem 6.2.1 transfers immediately to the dual problem. Specifically, we have the following corollary:

Corollary 6.2.1. *For sufficiently large penalty parameter $\widehat{\phi} > 0$, the dual discretization (6.22) is a consistent and stable discretization of the adjoint PDE. Moreover, the following a priori error bounds hold:*

$$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^h\|_{\mathrm{DG}} \leq C\, h^{\min(p+1,s)-1}\, |\!|\!| \boldsymbol{\lambda} |\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)} \ , \tag{6.23}$$

and

$$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^h\|_{\mathcal{L}^2(\Omega)} \leq C\, h^{\min(p+1,s)}\, |\!|\!| \boldsymbol{\lambda} |\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)} \ . \tag{6.24}$$

Here C is a constant independent of the mesh size h.

Remark. The discrete adjoint solution may be super-convergent in practice. However, we shall use the conservative error bounds (6.23)–(6.24) throughout our derivations.

It remains to be determined is whether the discrete optimality condition (1.8c) is a stable and consistent discretization of the strong form optimality condition (1.2c). The third equation in the KKT set for problem (6.20) reads:

$$\frac{\partial \mathcal{J}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{z}^h) = \frac{\partial \mathcal{N}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h, \boldsymbol{\lambda}^h](\mathbf{z}^h) - \frac{\partial \mathcal{B}^h}{\partial \mathbf{q}^h}[\mathbf{g}_D^h, \boldsymbol{\lambda}^h](\mathbf{w}^h) \ , \ \forall\, \mathbf{z}^h \in \mathcal{Q}_h^r \ . \tag{6.25}$$

Now, for all admissible test functionals $\mathbf{z}^h \in \mathcal{Q}_h^r$ (which also denote the direction of differentiation), one has:

$$\begin{aligned}
\frac{\partial \mathcal{N}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h, \boldsymbol{\lambda}^h](\mathbf{z}^h) &:= \int_\Omega \mathbf{z}^h \, \nabla \mathbf{u}^h \cdot \nabla \boldsymbol{\lambda}^h \, \mathrm{d}\mathbf{x} \\
&\quad - \int_{\Gamma_{\mathcal{I}} \cup \Gamma} \left([\![\mathbf{u}^h]\!] \cdot \{ \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \} + \{ \mathbf{z}^h \, \nabla \mathbf{u}^h \} \cdot [\![\boldsymbol{\lambda}^h]\!] \right) \, \mathrm{d}\mathbf{s} \, , \\
\frac{\partial \mathcal{B}^h}{\partial \mathbf{q}^h}[\mathbf{g}_D^h, \boldsymbol{\lambda}^h](\mathbf{z}^h) &:= - \int_\Gamma \mathbf{z}^h \, \mathbf{g}^h \, \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \, \mathrm{d}\mathbf{s} \, .
\end{aligned}$$

Since

$$\frac{\partial \mathcal{J}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{z}^h) = \int_\Omega \left(\nabla \mathbf{q}^h - \nabla \mathbf{q}_B^h \right) \, \mathbf{z}^h \, \mathrm{d}\mathbf{x} + \int_\Omega \left(\mathbf{q}^h - \mathbf{q}_B^h \right) \, \mathbf{z}^h \, \mathrm{d}\mathbf{x} \, ,$$

the discrete optimality condition reads:

$$\begin{aligned}
&\int_\Omega \nabla \left(\mathbf{q}^h - \mathbf{q}_B^h \right) \cdot \nabla \mathbf{z}^h \, \mathrm{d}\mathbf{x} + \boldsymbol{\beta} \int_\Omega (\mathbf{q}^h - \mathbf{q}_B^h) \, \mathbf{z}^h \, \mathrm{d}\mathbf{x} + \frac{\partial \mathcal{R}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{z}^h) \\
= &\int_\Omega \left(\nabla \mathbf{u}^h \cdot \nabla \boldsymbol{\lambda}^h \right) \, \mathbf{z}^h \, \mathrm{d}\mathbf{x} - \int_{\Gamma_{\mathcal{I}}} \left([\![\mathbf{u}^h]\!] \cdot \{ \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \} + \{ \mathbf{z}^h \, \nabla \mathbf{u}^h \} \cdot [\![\boldsymbol{\lambda}^h]\!] \right) \, \mathrm{d}\mathbf{s} \\
&- \int_\Gamma \left(\mathbf{u}^h - \mathbf{g}^h \right) \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \, \mathrm{d}\mathbf{s} - \int_\Gamma \boldsymbol{\lambda}^h \, \nabla \mathbf{u}^h \cdot \vec{\mathbf{n}} \, \mathbf{z}^h \, \mathrm{d}\mathbf{s} \, , \; \forall \, \mathbf{z}^h \in \mathcal{Q}_h^r \, .
\end{aligned}$$

We can recast this as an equation for

$$\widehat{\mathbf{q}}^h := \mathbf{q}^h - \mathbf{q}_B^h \, ,$$

to get

$$\begin{aligned}
&\int_\Omega \nabla \widehat{\mathbf{q}}^h \cdot \nabla \mathbf{z}^h \, \mathrm{d}\mathbf{x} + \boldsymbol{\beta} \int_\Omega \widehat{\mathbf{q}}^h \, \mathbf{z}^h \, \mathrm{d}\mathbf{x} + \frac{\partial \mathcal{R}^h}{\partial \widehat{\mathbf{q}}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{z}^h) \\
= &\int_\Omega \left(\nabla \mathbf{u}^h \cdot \nabla \boldsymbol{\lambda}^h \right) \, \mathbf{z}^h \, \mathrm{d}\mathbf{x} - \int_{\Gamma_{\mathcal{I}}} \left([\![\mathbf{u}^h]\!] \cdot \{ \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \} + \{ \mathbf{z}^h \, \nabla \mathbf{u}^h \} \cdot [\![\boldsymbol{\lambda}^h]\!] \right) \, \mathrm{d}\mathbf{s} \\
&- \int_\Gamma \left(\mathbf{u}^h - \mathbf{g}^h \right) \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \, \mathrm{d}\mathbf{s} - \int_\Gamma \boldsymbol{\lambda}^h \, \nabla \mathbf{u}^h \cdot \vec{\mathbf{n}} \, \mathbf{z}^h \, \mathrm{d}\mathbf{s} \, , \; \forall \, \mathbf{z}^h \in \mathcal{Q}_h^r \, .
\end{aligned} \tag{6.26}$$

6.2.3 *A priori* analysis of the discrete optimality equation

The SIPG DG discretization of (6.4) reads [171]:

$$\begin{aligned}
&\text{Find } \widehat{\mathbf{q}}^h \in \mathcal{Q}_h^p \text{ such that, for all } \mathbf{z}^h \in \mathcal{Q}_h^r: \\
&\int_\Omega \nabla \widehat{\mathbf{q}}^h \cdot \nabla \mathbf{z}^h \,\mathrm{d}\mathbf{x} + \boldsymbol{\beta} \int_\Omega \widehat{\mathbf{q}}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \\
&- \int_{\Gamma_{\mathcal{I}}^q} \left([\![\widehat{\mathbf{q}}^h]\!] \cdot \{ \nabla \mathbf{z}^h \} + \{ \nabla \widehat{\mathbf{q}}^h \} \cdot [\![\mathbf{z}^h]\!] \right) \mathrm{d}\mathbf{s} + \int_{\Gamma_{\mathcal{I}}^q} \phi^q \, [\![\widehat{\mathbf{q}}^h]\!] \cdot [\![\mathbf{z}^h]\!] \,\mathrm{d}\mathbf{s} \\
&= \int_\Omega \nabla \mathbf{u}^h \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \,.
\end{aligned} \tag{6.27}$$

Here the penalty parameter ϕ^q is defined for any edge $e_q \in \Gamma_{\mathcal{I}}^q$, and dimension $d \in \{2, 3\}$:

$$\phi^q := |e_q|^{1/(d-1)} \; \hat{\phi}^q \,,$$

and $\hat{\phi}^q > 0$ sufficiently large for the discretization to be convergent (see, e.g., [171]).

Comparing the SIPG discretization (6.27) of the strong form optimality problem (6.4) against the discrete equation (6.26), we note that the stabilization terms, as well as the fluxes on interior faces are not present in the linearized discretization (6.26). Hence, despite being consistent as defined in definition 1.1.3, the discrete optimality equation (6.26) is not stable, and cannot be expected to yield useful results.

To add the necessary terms back into (6.26), a consistent modification of the functional $\mathcal{J}_h$ is needed. Precisely, let:

$$\mathcal{R}^h(\mathbf{u}^h, \widehat{\mathbf{q}}^h) := \frac{1}{2} \int_{\Gamma_{\mathcal{I}}^q} \phi^q \, [\![\widehat{\mathbf{q}}^h]\!] \cdot [\![\widehat{\mathbf{q}}^h]\!] \,\mathrm{d}\mathbf{s} - \int_{\Gamma_{\mathcal{I}}^q} [\![\widehat{\mathbf{q}}^h]\!] \cdot \{ \nabla \widehat{\mathbf{q}}^h \} \,\mathrm{d}\mathbf{s} \,. \tag{6.28}$$

For $\mathbf{q}$, $\mathbf{q}_B \in \mathcal{H}^2(\Omega)$, consistency of the functional $\mathcal{J}_h$ is preserved. Note that the jump in $\mathbf{q}^h$ over any edge $e \in \Gamma_{\mathcal{I}} \backslash \Gamma_{\mathcal{I}}^q$ is zero. This follows from the continuity of the discrete optimal solution inside any element $\kappa^q \in \mathcal{T}_h^q$.

Remark. The residual $\mathcal{R}^h$ is directly computable from the discrete approximations $\mathbf{u}^h$ and $\mathbf{q}^h$, and it does not depend on the dual variable $\boldsymbol{\lambda}^h$. Moreover, its Fréchet derivative with respect to $\mathbf{q}^h$ along $\mathbf{z}^h \in \mathcal{Q}_h^p$ introduces suitable interior edge and penalty terms in equation (6.26), while leaving the discrete adjoint equation (6.22) unmodified:

$$\begin{aligned}
\frac{\partial \mathcal{R}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h, \mathbf{q}^h](\mathbf{z}^h) \;=\; & \int_{\Gamma_{\mathcal{I}}^q} \phi^q \, [\![\widehat{\mathbf{q}}^h]\!] \cdot [\![\mathbf{z}^h]\!] \,\mathrm{d}\mathbf{s} \\
& - \int_{\Gamma_{\mathcal{I}}^q} \left([\![\mathbf{z}^h]\!] \cdot \{ \nabla \widehat{\mathbf{q}}^h \} + [\![\widehat{\mathbf{q}}^h]\!] \cdot \{ \nabla \mathbf{z}^h \} \right) \mathrm{d}\mathbf{s} \,.
\end{aligned} \tag{6.29}$$

With this choice of $\mathcal{R}^h$, equation (6.26) reads:

$$\begin{aligned}
&\int_\Omega \nabla\widehat{\mathbf{q}}^h \cdot \nabla\mathbf{z}^h \,\mathrm{d}\mathbf{x} + \boldsymbol{\beta} \int_\Omega \widehat{\mathbf{q}}^h\, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \\
&\quad - \int_{\Gamma_{\mathcal{I}}^q} \left([\![\widehat{\mathbf{q}}^h]\!] \cdot \{\nabla\mathbf{z}^h\} + \{\nabla\widehat{\mathbf{q}}^h\} \cdot [\![\mathbf{z}^h]\!] \right) \mathrm{d}\mathbf{s} + \int_{\Gamma_{\mathcal{I}}^q} \phi^q [\![\widehat{\mathbf{q}}^h]\!] \cdot [\![\mathbf{z}^h]\!] \,\mathrm{d}\mathbf{s} \\
= &\int_\Omega \nabla\mathbf{u}^h \cdot \nabla\boldsymbol{\lambda}^h\, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Gamma \boldsymbol{\lambda}^h\, \nabla\mathbf{u}^h \cdot \vec{\mathbf{n}}\, \mathbf{z}^h \,\mathrm{d}\mathbf{s} \\
&\quad - \int_{\Gamma_{\mathcal{I}}} \left([\![\mathbf{u}^h]\!] \cdot \{\mathbf{z}^h\, \nabla\boldsymbol{\lambda}^h\} + \{\mathbf{z}^h\, \nabla\mathbf{u}^h\} \cdot [\![\boldsymbol{\lambda}^h]\!] \right) \mathrm{d}\mathbf{s} \\
&\quad - \int_\Gamma \left(\mathbf{u}^h - \mathbf{g}^h \right) \mathbf{z}^h\, \nabla\boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \,\mathrm{d}\mathbf{s} \,.
\end{aligned} \tag{6.30}$$

Let

$$\begin{aligned}
I_\Gamma^1 &:= \int_\Gamma (\mathbf{u}^h - \mathbf{g}^h)\, \nabla\boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}}\, \mathbf{z}^h \,\mathrm{d}\mathbf{s} \,, \\
I_\Gamma^2 &:= \int_\Gamma \boldsymbol{\lambda}^h\, \nabla\mathbf{u}^h \cdot \vec{\mathbf{n}}\, \mathbf{z}^h \,\mathrm{d}\mathbf{s} \,,
\end{aligned}$$

and

$$\begin{aligned}
I_{\Gamma_{\mathcal{I}}}^1 &:= \int_{\Gamma_{\mathcal{I}}} [\![\mathbf{u}^h]\!]\, \{\mathbf{z}^h\, \nabla\boldsymbol{\lambda}^h\} \,\mathrm{d}\mathbf{s} \,, \\
I_{\Gamma_{\mathcal{I}}}^2 &:= \int_{\Gamma_{\mathcal{I}}} \{\mathbf{z}^h\, \nabla\mathbf{u}^h\} \cdot [\![\boldsymbol{\lambda}^h]\!] \,\mathrm{d}\mathbf{s} \,,
\end{aligned}$$

for arbitrary $\mathbf{z}^h \in \mathcal{Q}_h^r$.

With this notation, we have the following *a priori* estimates.

Lemma 6.2.1. *Assume* $\mathbf{u}, \boldsymbol{\lambda} \in \mathcal{H}^s(\Omega)$*, and* $\mathbf{g} \in \mathcal{H}^{\widehat{s}}(\Gamma)$*, with* $\widehat{s} = s - 1/2$*. The following upper bounds hold:*

$$I_\Gamma^1 \leq \mathcal{C}(p,r)\, h^{\min(p+1,s)-3/2}\, |\!|\!| \mathbf{u} |\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)} \tag{6.31a}$$

$$I_\Gamma^2 \leq \widehat{\mathcal{C}}(p,r)\, h^{\min(p+1,s)-3/2}\, |\!|\!| \boldsymbol{\lambda} |\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)} \tag{6.31b}$$

$$I_{\Gamma_{\mathcal{I}}}^1 \leq \mathcal{C}_1(p,r)\, h^{\min(p+1,s)-3/2}\, |\!|\!| \mathbf{u} |\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)} \tag{6.31c}$$

$$I_{\Gamma_{\mathcal{I}}}^2 \leq \mathcal{C}_2(p,r)\, h^{\min(p+1,s)-3/2}\, |\!|\!| \boldsymbol{\lambda} |\!|\!|_{\mathcal{H}^s(\mathcal{T}_h)} \,. \tag{6.31d}$$

The constants $\mathcal{C}(p,r)$*,* $\widehat{\mathcal{C}}(p,r)$*,* $\mathcal{C}_1(p,r)$*, and* $\mathcal{C}_2(p,r)$ *depend only on the orders* p, r *of the polynomial bases.*

Proof. We can write

$$\begin{aligned} I_\Gamma &= \int_\Gamma \left[(\mathbf{u}^h - \mathbf{u}) + (\mathbf{u} - \mathbf{g}) + (\mathbf{g} - \mathbf{g}^h)\right] \mathbf{z}^h \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \, \mathrm{d}\mathbf{s} \\ &= \int_\Gamma (\mathbf{u}^h - \mathbf{u}) \mathbf{z}^h \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \, \mathrm{d}\mathbf{s} + \int_\Gamma (\mathbf{g} - \mathbf{g}^h) \mathbf{z}^h \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \, \mathrm{d}\mathbf{s} \\ &:= I_1 + I_2 \ . \end{aligned}$$

It follows that:

$$\begin{aligned} |I_1| &= \left| \int_\Gamma (\mathbf{u} - \mathbf{u}^h) \mathbf{z}^h (\nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}}) \, \mathrm{d}\mathbf{s} \right| \\ &\leq \sum_{e \in \Gamma} \left| \int_e h_\kappa^{-1/2} (\mathbf{u} - \mathbf{u}^h) h^{1/2} \mathbf{z}^h (\nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}}) \, \mathrm{d}\mathbf{s} \right| \\ &\leq \sum_{e \in \Gamma} c_e(r,p) \left\| h_\kappa^{-1/2} (\mathbf{u} - \mathbf{u}^h) \right\|_{\mathcal{L}^2(e)} \cdot h^{1/2} \cdot \left\| \mathbf{z}^h \right\|_{\mathcal{L}^2(e)} \cdot \left\| \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \right\|_{\mathcal{L}^2(e)} \ . \end{aligned}$$

with $c_e(r,p)$ a constant that depends only on the polynomial basis orders p and r. The trace inequalities (6.10a)–(6.10b) lead to:

$$\begin{aligned} |I_1| &\leq C(r,p) \, h^{-1/2} \left\| \mathbf{u} - \mathbf{u}^h \right\|_{\mathrm{DG}} \cdot \sum_{\kappa^q \cap \Gamma \neq \emptyset} \left\| \mathbf{z}^h \right\|_{\mathcal{L}^2(\kappa^q)} \cdot \sum_{\kappa \cap \Gamma \neq \emptyset} \left\| \nabla \boldsymbol{\lambda}^h \right\|_{\mathcal{L}^2(\kappa)} \\ &\leq \widetilde{C}(r,p) \, h^{-1/2} \left\| \mathbf{u} - \mathbf{u}^h \right\|_{\mathrm{DG}} \cdot \left\| \mathbf{z}^h \right\|_{\mathcal{L}^2(\mathcal{T}_h^q)} \cdot \left\| \nabla \boldsymbol{\lambda}^h \right\|_{\mathcal{L}^2(\mathcal{T}_h)} \\ &\leq \mathcal{C}_{I_1}(r,p) \, h^{\min(p+1,s)-3/2} \left\| \mathbf{u} \right\|_{\mathcal{H}^s(\mathcal{T}_h)} \ . \end{aligned} \tag{6.32}$$

The constant $C(r,p)$ has been chosen sufficiently large such that $\max_{e \in \Gamma} c_e(r,p) \leq C(r,p)$. We have used the fact that both $\mathbf{z}^h$, and $\nabla \boldsymbol{\lambda}^h$ are bounded up by a constant value (independent of h) on any mesh element $\kappa \in \mathcal{T}_h$, or $\kappa^q \in \mathcal{T}_h^q$. This is guaranteed by the numerical stability of the primal and dual discretizations, and by the choice of cost functionals (piecewise smooth polynomials).

The integral I_2 can be bounded using classical results from polynomial approximation theory for finite element methods. Given that the boundary data $\mathbf{g} \in \mathcal{H}^s(\Gamma)$, the following upper bound holds for the interpolation error on Γ [170]:

$$\left\| \mathbf{g} - \mathbf{g}^h \right\|_{\mathcal{L}^2(\Gamma)} \leq C_{\mathbf{g}}(p) \, h^{\min(p+1,\widehat{s})} \, |\mathbf{g}|_{\mathcal{H}^{\widehat{s}}(\Gamma)} \ , \tag{6.33}$$

with a constant C_g independent of h. The functional $|\cdot|$ is the broken Sobolev semi–norm on Γ.

The Cauchy–Schwartz inequality gives:

$$\begin{aligned}
|I_2| &\leq \left\|\mathbf{g}-\mathbf{g}^h\right\|_{\mathcal{L}^2(\Gamma)} \cdot \left\|\mathbf{z}^h\right\|_{\mathcal{L}^2(\Gamma)} \cdot \left\|\nabla\boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}}\right\|_{\mathcal{L}^2(\Gamma)} \\
&\leq C(r,p) \left\|\mathbf{g}-\mathbf{g}^h\right\|_{\mathcal{L}^2(\Gamma)} \cdot \sum_{e\in\Gamma} \left\|\mathbf{z}^h\right\|_{\mathcal{L}^2(e)} \cdot \left\|\nabla\boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}}\right\|_{\mathcal{L}^2(e)} \\
&\leq \hat{C}(r,p) \left\|\mathbf{g}-\mathbf{g}^h\right\|_{\mathcal{L}^2(\Gamma)} \cdot h^{-1} \cdot \sum_{\kappa\cap\Gamma\neq\emptyset} \left\|\mathbf{z}^h\right\|_{\mathcal{L}^2(\kappa)} \cdot \left\|\nabla\boldsymbol{\lambda}^h\right\|_{\mathcal{L}^2(\kappa)} \\
&\leq \hat{C}(r,p) \left\|\mathbf{g}-\mathbf{g}^h\right\|_{\mathcal{L}^2(\Gamma)} \cdot h^{-1} \cdot \left|\!\left|\!\left|\mathbf{z}^h\right|\!\right|\!\right|_{\mathcal{L}^2(\Gamma)} \cdot \left|\!\left|\!\left|\nabla\boldsymbol{\lambda}^h\right|\!\right|\!\right|_{\mathcal{L}^2(\mathcal{T}_h)} \\
&\leq C_{I_2}(r,p)\, h^{\min(p+1,\widehat{s})-1} \left|\mathbf{g}\right|_{\mathcal{H}^{\widehat{s}}(\Gamma)} .
\end{aligned}$$

This proves inequality (6.31a). Note that (6.31b) follows immediately from an identical argument, since $\boldsymbol{\lambda}|_\Gamma = 0$, by equation (6.3).

Consider now the integral $I^1_{\Gamma_\mathcal{I}}$. Since the jump operator $[\![\cdot]\!]$ is linear, and $[\![\mathbf{u}]\!] = 0$ (for all $\mathbf{u}\in\mathcal{U}$), we can write:

$$\begin{aligned}
|I_3| &\leq \int_{\Gamma_\mathcal{I}} \left|[\![\mathbf{u}^h-\mathbf{u}]\!]\,\{\mathbf{z}^h\,\nabla\boldsymbol{\lambda}^h\}\right| \mathrm{d}\mathbf{s} \\
&\leq \int_{\Gamma_\mathcal{I}} \left|h^{-1/2}[\![\mathbf{u}^h-\mathbf{u}]\!]\, h^{1/2}\,\{\mathbf{z}^h\,\nabla\boldsymbol{\lambda}^h\}\right| \mathrm{d}\mathbf{s} \\
&\leq \left\|h^{-1/2}[\![\mathbf{u}^h-\mathbf{u}]\!]\right\|_{\mathcal{L}^2(\Gamma_\mathcal{I})} h^{1/2} \sum_{e\in\Gamma_\mathcal{I}} \left\|\{\mathbf{z}^h\,\nabla\boldsymbol{\lambda}^h\}\right\|_{L^2(e)} \\
&\leq \left\|\mathbf{u}-\mathbf{u}^h\right\|_{\mathrm{DG}} h^{1/2} \sum_{e\in\Gamma_\mathcal{I}} \frac{1}{2} \left\|\mathbf{z}^h_+\,\nabla\boldsymbol{\lambda}^h_+ + \mathbf{z}^h_-\nabla\boldsymbol{\lambda}^h_-\right\|_{\mathcal{L}^2(e)} \\
&\leq \frac{1}{2} \left\|\mathbf{u}-\mathbf{u}^h\right\|_{\mathrm{DG}} h^{1/2} \sum_{e\in\Gamma_\mathcal{I}} \left(\left\|\mathbf{z}^h_+\,\nabla\boldsymbol{\lambda}^h_+\right\|_{L^2(e)} + \left\|\mathbf{z}^h_-\,\nabla\boldsymbol{\lambda}^h_-\right\|_{L^2(e)}\right) \\
&\leq h^{1/2} \left\|\mathbf{u}-\mathbf{u}^h\right\|_{\mathrm{DG}} \sum_{\kappa_\pm\in\mathcal{T}_h} \left(\widehat{C}_{\kappa_+}(r,p)\, h^{-1}_{\kappa_+} \left\|\mathbf{z}^h\right\|_{\mathcal{L}^2(\kappa_+)} \left\|\nabla\boldsymbol{\lambda}^h\right\|_{\mathcal{L}^2(\kappa_+)} + \right. \\
&\qquad\qquad \left. \widehat{C}_{\kappa_-}(r,p)\, h^{-1}_{\kappa_-} \left\|\mathbf{z}^h\right\|_{\mathcal{L}^2(\kappa_-)} \left\|\nabla\boldsymbol{\lambda}^h\right\|_{\mathcal{L}^2(\kappa_-)}\right) \\
&\leq C_{\max}\, h^{-1/2} \left\|\mathbf{u}-\mathbf{u}^h\right\|_{\mathrm{DG}} \left|\!\left|\!\left|\mathbf{z}^h\right|\!\right|\!\right|_{\mathcal{L}^2(\mathcal{T}_h)} \left|\!\left|\!\left|\nabla\boldsymbol{\lambda}^h\right|\!\right|\!\right|_{\mathcal{L}^2(\mathcal{T}_h)} \\
&\leq \mathcal{C}_{I_3}(p)\, h^{\min(p+1,s)-3/2} \left\|\mathbf{u}\right\|_{\mathcal{H}^s(\mathcal{T}_h)} .
\end{aligned}$$

where $\max\limits_{\kappa\in\Omega} \widehat{C}_{\kappa_\pm}(r,p) \leq C_{\max}(r,p)$. Thus (6.31c) holds. Since the SIPG discretization is dual consistent, (6.31d) follows. □

An immediate consequence of the lemma is the following corollary:

Corollary 6.2.2. *For all $p \geq 1$ and $s > 3/2$, the discrete optimality equation associated to the problem (6.20) is asymptotically consistent.*

Remark. With our specific smoothness assumptions ($s = 2$, $\widehat{s} = 3/2$), the integrals (6.31a)–(6.31d) are $\mathcal{O}(h^{1/2})$.

6.3 *A priori* convergence order

Let $\mathbf{e}_{\mathbf{q}}^h = \widehat{\mathbf{q}}^h - \widetilde{\mathbf{q}}^h$, $\mathbf{e}_{\mathbf{u}} = \mathbf{u} - \mathbf{u}^h$, and $\mathbf{e}_{\boldsymbol{\lambda}} = \boldsymbol{\lambda} - \boldsymbol{\lambda}^h$. Subtracting (6.27) from (6.30) gives the following equation for $\mathbf{e}_{\mathbf{q}}^h$:

Find $\mathbf{e}_{\mathbf{q}}^h \in \mathcal{Q}_h^p$ such that, for all $\mathbf{z}^h \in \mathcal{Q}_h^r$:

$$
\begin{aligned}
&\int_\Omega \nabla \mathbf{e}_{\mathbf{q}}^h \cdot \nabla \mathbf{z}^h \,\mathrm{d}\mathbf{x} + \boldsymbol{\beta} \int_\Omega \mathbf{e}_{\mathbf{q}}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_{\Gamma_{\mathcal{I}}^q} \left([\![\mathbf{e}_{\mathbf{q}}^h]\!] \cdot \{ \nabla \mathbf{z}^h \} + \{ \nabla \mathbf{e}_{\mathbf{q}}^h \} \cdot [\![\mathbf{z}^h]\!] \right) \mathrm{d}\mathbf{s} + \int_{\Gamma_{\mathcal{I}}^q} \phi^q \, [\![\mathbf{e}_{\mathbf{q}}^h]\!] \cdot [\![\mathbf{z}^h]\!] \,\mathrm{d}\mathbf{s} \\
&= \int_\Omega \nabla \mathbf{u}^h \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Omega \nabla \mathbf{u} \cdot \nabla \boldsymbol{\lambda} \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \\
&\quad - \int_\Gamma \boldsymbol{\lambda}^h \, \nabla \mathbf{u}^h \cdot \vec{\mathbf{n}} \, \mathbf{z}^h \,\mathrm{d}\mathbf{s} - \int_{\Gamma_{\mathcal{I}}} \left([\![\mathbf{u}^h]\!] \cdot \{ \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \} + \{ \mathbf{z}^h \, \nabla \mathbf{u}^h \} \cdot [\![\boldsymbol{\lambda}^h]\!] \right) \mathrm{d}\mathbf{s} \\
&\quad - \int_\Gamma \left(\mathbf{u}^h - \mathbf{g}^h \right) \mathbf{z}^h \, \nabla \boldsymbol{\lambda}^h \cdot \vec{\mathbf{n}} \,\mathrm{d}\mathbf{s} \\
&= \int_\Omega \nabla \mathbf{u}^h \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Omega \nabla \mathbf{u} \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} + \int_\Omega \nabla \mathbf{u} \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Omega \nabla \mathbf{u} \cdot \nabla \boldsymbol{\lambda} \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \\
&\quad - I_\Gamma^1 - I_\Gamma^2 - I_{\Gamma_{\mathcal{I}}}^1 - I_{\Gamma_{\mathcal{I}}}^2 \,. \\
&= - \int_\Omega \nabla \mathbf{e}_{\mathbf{u}} \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{e}_{\boldsymbol{\lambda}} \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \\
&\quad - I_\Gamma^1 - I_\Gamma^2 - I_{\Gamma_{\mathcal{I}}}^1 - I_{\Gamma_{\mathcal{I}}}^2 \,. \\
&= - \int_\Omega \nabla \mathbf{e}_{\mathbf{u}} \cdot \nabla \boldsymbol{\lambda}^h \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Omega \nabla \mathbf{u}^h \cdot \nabla \mathbf{e}_{\boldsymbol{\lambda}} \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} - \int_\Omega \nabla \mathbf{e}_{\mathbf{u}} \cdot \nabla \mathbf{e}_{\boldsymbol{\lambda}} \, \mathbf{z}^h \,\mathrm{d}\mathbf{x} \\
&\quad - I_\Gamma^1 - I_\Gamma^2 - I_{\Gamma_{\mathcal{I}}}^1 - I_{\Gamma_{\mathcal{I}}}^2
\end{aligned}
$$

Consider the following elliptic error equation defined over the broken Sobolev space $\mathcal{H}^s(\mathcal{T}_h)$:

$$
\begin{aligned}
-\Delta \, \mathbf{e}_{\mathbf{q}} + \boldsymbol{\beta} \, \mathbf{e}_{\mathbf{q}} &= -\nabla \mathbf{e}_{\mathbf{u}} \cdot \nabla \boldsymbol{\lambda}^h - \nabla \mathbf{u}^h \cdot \nabla \mathbf{e}_{\boldsymbol{\lambda}} - \nabla \mathbf{e}_{\mathbf{u}} \cdot \nabla \mathbf{e}_{\boldsymbol{\lambda}} \,, \; \mathbf{x} \in \Omega \,, \qquad (6.34) \\
\nabla \mathbf{e}_{\mathbf{q}} \cdot \vec{\mathbf{n}} &= 0 \,, \; \mathbf{x} \in \Gamma \,.
\end{aligned}
$$

Under our solution regularity assumptions, the Lax–Milgram theorem [171, Theorem 2.8]

guarantees the existence of a constant C independent on h, such that:

$$\begin{aligned}
\left|\!\left|\!\left|\mathbf{e_q}\right|\!\right|\!\right|_{\mathcal{H}^s(\mathcal{T}_h)} &\leq C \left\|\nabla\mathbf{e_u}\cdot\nabla\boldsymbol{\lambda}^h + \nabla\mathbf{u}^h\cdot\nabla\mathbf{e_\lambda} + \nabla\mathbf{e_u}\cdot\nabla\mathbf{e_\lambda}\right\|_{\mathcal{L}^2(\mathcal{T}_h)} \\
&\leq C \sum_{\kappa\in\mathcal{T}_h}\left|\int_{\mathcal{T}_k}\left(\nabla\mathbf{e_u}\cdot\nabla\boldsymbol{\lambda}^h + \nabla\mathbf{u}^h\cdot\nabla\mathbf{e_\lambda} + \nabla\mathbf{e_u}\cdot\nabla\mathbf{e_\lambda}\right)\,\mathrm{d}\mathbf{x}\right| \\
&\leq C \sum_{\kappa\in\mathcal{T}_h}\Big(\|\nabla\mathbf{e_u}\|_{\mathcal{L}^2(\kappa)}\,\|\nabla\boldsymbol{\lambda}^h\|_{\mathcal{L}^2(\kappa)} + \|\nabla\mathbf{e_\lambda}\|_{\mathcal{L}^2(\kappa)}\,\|\nabla\mathbf{u}^h\|_{\mathcal{L}^2(\kappa)} \\
&\qquad + \|\nabla\mathbf{e_u}\|_{\mathcal{L}^2(\kappa)}\,\|\nabla\mathbf{e_\lambda}\|_{\mathcal{L}^2(\kappa)}\Big)\ .
\end{aligned} \tag{6.35}$$

Now, by the discrete Cauchy inequality, and Theorem 6.2.1, one has:

$$\begin{aligned}
\sum_{\kappa\in\mathcal{T}_h}\|\nabla\mathbf{e_u}\|_{\mathcal{L}^2(\kappa)}\,\|\nabla\boldsymbol{\lambda}^h\|_{\mathcal{L}^2(\kappa)} &\leq \left(\sum_{\kappa\in\mathcal{T}_h}\|\nabla\mathbf{e_u}\|_{\mathcal{L}^2(\kappa)}\right)^{1/2}\left(\sum_{\kappa\in\mathcal{T}_h}\|\nabla\boldsymbol{\lambda}^h\|_{\mathcal{L}^2(\kappa)}\right)^{1/2} \\
&= \|\nabla\mathbf{e_u}\|_{\mathcal{L}^2(\mathcal{T}_h)}\cdot\|\nabla\boldsymbol{\lambda}^h\|_{\mathcal{L}^2(\mathcal{T}_h)} \\
&\leq C_{\mathbf{u}}(p)\,h^{\min(p+1,s)-1}\,\|\mathbf{u}\|_{\mathcal{H}^s(\mathcal{T}_h)}\,\|\nabla\boldsymbol{\lambda}^h\|_{\mathcal{L}^2(\mathcal{T}_h)}\ .
\end{aligned}$$

Similarly, using the convergence result for the dual problem (Corollary 6.2.1), we obtain:

$$\sum_{\kappa\in\mathcal{T}_h}\|\nabla\mathbf{e_\lambda}\|_{\mathcal{L}^2(\kappa)}\,\|\nabla\mathbf{u}^h\|_{\mathcal{L}^2(\kappa)} \leq C_{\boldsymbol{\lambda}}(p)\,h^{\min(p+1,s)-1}\,\|\boldsymbol{\lambda}\|_{\mathcal{H}^s(\mathcal{T}_h)}\,\|\nabla\mathbf{u}^h\|_{\mathcal{L}^2(\mathcal{T}_h)}\ ,$$

and

$$\sum_{\kappa\in\mathcal{T}_h}\|\nabla\mathbf{e_u}\|_{\mathcal{L}^2(\kappa)}\,\|\nabla\mathbf{e_\lambda}\|_{\mathcal{L}^2(\kappa)} \leq C_{\mathbf{u}}(p)\,C_{\boldsymbol{\lambda}}(p)\,h^{2\min(p+1,s)-2}\,\|\mathbf{u}\|_{\mathcal{H}^s(\mathcal{T}_h)}\,\|\boldsymbol{\lambda}\|_{\mathcal{H}^s(\mathcal{T}_h)}\ .$$

Substituting these upper bounds in equation (6.35), it follows that:

$$\left|\!\left|\!\left|\mathbf{e_q}\right|\!\right|\!\right|_{\mathcal{H}^s(\mathcal{T}_h)} \leq C\,h^{\min(p+1,s)-1}\left(\left|\!\left|\!\left|\mathbf{u}^h\right|\!\right|\!\right|_{\mathcal{H}^s(\mathcal{T}_h)} + \left|\!\left|\!\left|\boldsymbol{\lambda}^h\right|\!\right|\!\right|_{\mathcal{H}^s(\mathcal{T}_h)}\right)\ . \tag{6.36}$$

The unperturbed SIPG discretization of (6.34) is:

$$\begin{aligned}
&\text{Find } \overline{\mathbf{e}}^h_{\mathbf{q}}\in\mathcal{Q}^p_h \text{ such that, for all } \mathbf{z}^h\in\mathcal{Q}^r_h\text{:} \\
&\int_\Omega\nabla\overline{\mathbf{e}}^h_{\mathbf{q}}\cdot\nabla\mathbf{z}^h\,\mathrm{d}\mathbf{x} + \boldsymbol{\beta}\int_\Omega\overline{\mathbf{e}}^h_{\mathbf{q}}\,\mathbf{z}^h\,\mathrm{d}\mathbf{x} \\
&\quad - \int_{\Gamma^q_{\mathcal{I}}}\left(\left[\!\left[\overline{\mathbf{e}}^h_{\mathbf{q}}\right]\!\right]\cdot\{\nabla\mathbf{z}^h\} + \{\nabla\overline{\mathbf{e}}^h_{\mathbf{q}}\}\cdot\left[\!\left[\mathbf{z}^h\right]\!\right]\right)\,\mathrm{d}\mathbf{s} + \int_{\Gamma^q_{\mathcal{I}}}\phi^q\left[\!\left[\overline{\mathbf{e}}^h_{\mathbf{q}}\right]\!\right]\cdot\left[\!\left[\mathbf{z}^h\right]\!\right]\mathrm{d}\mathbf{s} \\
&= -\int_\Omega\nabla\mathbf{e_u}\cdot\nabla\boldsymbol{\lambda}^h\,\mathbf{z}^h\,\mathrm{d}\mathbf{x} - \int_\Omega\nabla\mathbf{u}^h\cdot\nabla\mathbf{e_\lambda}\,\mathbf{z}^h\,\mathrm{d}\mathbf{x} - \int_\Omega\nabla\mathbf{e_u}\cdot\nabla\mathbf{e_\lambda}\,\mathbf{z}^h\,\mathrm{d}\mathbf{x}\ .
\end{aligned} \tag{6.37}$$

The discrete solution $\overline{\mathbf{e}}_{\mathbf{q}}^h$ of (6.37) verifies the *a priori* error estimate given in Theorem 6.2.1, hence:

$$\begin{aligned} \|\mathbf{e}_{\mathbf{q}} - \overline{\mathbf{e}}_{\mathbf{q}}^h\|_{\mathrm{DG}} &\leq C\, h^{\min(p+1,s)-1}\, \|\!|\mathbf{e}_{\mathbf{q}}|\!\|_{\mathcal{H}^s(\mathcal{T}_h)} \\ &\leq C\, h^{2\min(p+1,s)-2} \left(\|\!|\mathbf{u}^h|\!\|_{\mathcal{H}^s(\mathcal{T}_h)} + \|\!|\boldsymbol{\lambda}^h|\!\|_{\mathcal{H}^s(\mathcal{T}_h)} \right) . \end{aligned}$$

One obtains:

$$\begin{aligned} \|\overline{\mathbf{e}}_{\mathbf{q}}^h\|_{\mathrm{DG}} &\leq \|\!|\mathbf{e}_{\mathbf{q}}|\!\|_{\mathrm{DG}} + \|\!|\mathbf{e}_{\mathbf{q}} - \overline{\mathbf{e}}_{\mathbf{q}}^h|\!\|_{\mathrm{DG}} \\ &\leq C\, h^{\min(p+1,s)-1} \left(\|\!|\mathbf{u}^h|\!\|_{\mathcal{H}^s(\mathcal{T}_h)} + \|\!|\boldsymbol{\lambda}^h|\!\|_{\mathcal{H}^s(\mathcal{T}_h)} \right) . \end{aligned}$$

The equation for the discrete optimal solution error, denoted by $\mathbf{e}_{\mathbf{q}}^h$, obtained by subtracting (6.27) from (6.30), is a perturbed SIPG discretization. The perturbation is given by the sum of the boundary and inner face integrals

$$R_I := I_\Gamma^1 + I_\Gamma^2 + I_{\Gamma_{\mathcal{I}}}^1 + I_{\Gamma_{\mathcal{I}}}^2 .$$

Subtracting the SIPG discretization of the error equation from the equation for $\mathbf{e}_{\mathbf{q}}^h$, leads to:

$$\begin{aligned} \mathcal{N}_{\mathbf{e_q}} \left(\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h,\, \mathbf{z}^h \right) &= \int_\Omega \nabla \left(\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h \right) \cdot \nabla \mathbf{z}^h \, \mathrm{d}\mathbf{x} + \boldsymbol{\beta} \int_\Omega \left(\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h \right) \mathbf{z}^h \, \mathrm{d}\mathbf{x} \\ &\quad - \int_{\Gamma_{\mathcal{I}}^q} \left([\![\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h]\!] \cdot \{ \nabla \mathbf{z}^h \} + \{ \nabla \left(\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h \right) \} \cdot [\![\mathbf{z}^h]\!] \right) \mathrm{d}\mathbf{s} \\ &\quad + \int_{\Gamma_{\mathcal{I}}^q} \phi^q \, [\![\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h]\!] \cdot [\![\mathbf{z}^h]\!] \, \mathrm{d}\mathbf{s} \\ &= -R_I . \end{aligned} \tag{6.38}$$

The bilinear form $\mathcal{N}_{\mathbf{e_q}}(\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h,\, \mathbf{z}^h) : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is of SIPG form. Thus, it is continuous and coercive [171]. The right hand side of equation (6.38) is linear in the test functionals $\mathbf{z}^h$. By the Lax–Milgram theorem applied on $\mathbb{R}^N$, and (6.31a)–(6.31d), we obtain the following a priori bound:

$$\begin{aligned} \|\!|\mathbf{e}_{\mathbf{q}}^h - \overline{\mathbf{e}}_{\mathbf{q}}^h|\!\|_{\mathcal{H}^s(\mathcal{T}_h^q)} &\leq C\, |I_\Gamma^1 + I_\Gamma^2 + I_{\Gamma_{\mathcal{I}}}^1 + I_{\Gamma_{\mathcal{I}}}^2| \\ &\leq C(r,p)\, h^{\min(p+1,s)-3/2} \left(\|\mathbf{u}\|_{\mathcal{L}^2(\mathcal{T}_h)} + \|\boldsymbol{\lambda}\|_{\mathcal{L}^2(\mathcal{T}_h)} \right) . \end{aligned} \tag{6.39}$$

6.3.1 The discrete reduced gradient

We now compute the reduced gradient $\nabla_{\mathbf{q}^h} \mathcal{J}^h$ of the discrete cost functional (6.28). The gradient $\nabla_{\mathbf{q}^h} \mathcal{J}^h$ is defined by the following identity:

$$\left\langle \nabla_{\mathbf{q}^h} \mathcal{J}^h,\, \boldsymbol{\delta}\mathbf{q}^h \right\rangle_\Omega := \mathcal{J}_{\mathbf{q}^h}^h[\mathbf{u}^h, \mathbf{q}^h](\boldsymbol{\delta}\mathbf{q}^h) + \mathcal{J}_{\mathbf{u}^h}^h[\mathbf{u}^h, \mathbf{q}^h](\boldsymbol{\delta}\mathbf{u}^h) \;,\; \forall\, \boldsymbol{\delta}\mathbf{q}^h \in \mathcal{Q}_h^r \;, \tag{6.40}$$

where

$$\boldsymbol{\delta}\mathbf{u}^h := \frac{\partial \mathbf{u}^h}{\partial \mathbf{q}^h}[\mathbf{q}^h](\boldsymbol{\delta}\mathbf{q}^h) \ \in \ \mathcal{Q}_h^r \ .$$

The discrete adjoint solution $\boldsymbol{\lambda}^h$ is a valid test function for the primal model (6.12). The Fréchet derivative of (6.12), tested with $\boldsymbol{\lambda}^h$, in the direction $(\boldsymbol{\delta}\mathbf{u}^h, \boldsymbol{\delta}\mathbf{q}^h) \in \mathcal{U}_h^p \times \mathcal{Q}_h^r$, gives the discrete tangent linear model:

$$\begin{aligned} &\frac{\partial \mathcal{N}^h}{\partial \mathbf{q}^h}[\mathbf{u}^h,\, \boldsymbol{\lambda}^h](\boldsymbol{\delta}\mathbf{q}^h) + \frac{\partial \mathcal{N}^h}{\partial \mathbf{u}^h}[\mathbf{u}^h,\, \boldsymbol{\lambda}^h](\boldsymbol{\delta}\mathbf{u}^h) \\ &\quad - \frac{\partial \mathcal{B}^h}{\partial \mathbf{q}^h}[\mathbf{g}^h,\, \boldsymbol{\lambda}^h](\boldsymbol{\delta}\mathbf{q}^h) - \frac{\partial \mathcal{B}^h}{\partial \mathbf{u}^h}[\mathbf{g}^h, \boldsymbol{\lambda}^h](\boldsymbol{\delta}\mathbf{u}^h) = 0 \ . \end{aligned} \tag{6.41}$$

From the adjoint equation (6.22) we get:

$$\mathcal{N}^{h,*}(\boldsymbol{\delta}\mathbf{u}^h, \boldsymbol{\lambda}^h) = \mathcal{J}^h_{\mathbf{u}^h}[\mathbf{u}^h, \mathbf{q}^h](\boldsymbol{\delta}\mathbf{u}^h) \ . \tag{6.42}$$

Then, using (6.42) and (6.40), and the integration by parts formula on $\mathcal{T}_h^q$,

$$\begin{aligned} \int_\Omega \nabla \mathbf{q}^h \cdot \nabla \boldsymbol{\delta}\mathbf{q}^h \,\mathrm{d}\mathbf{x} \ &= \ -\int_\Omega \Delta \mathbf{q}^h \, \boldsymbol{\delta}\mathbf{q}^h \,\mathrm{d}\mathbf{x} \\ &\quad + \int_{\Gamma^q_{\mathcal{I}} \cup \Gamma} \{\nabla \mathbf{q}^h\} \cdot [\![\boldsymbol{\delta}\mathbf{q}^h]\!] \,\mathrm{d}\mathbf{s} + \int_{\Gamma^q_{\mathcal{I}}} [\![\nabla \mathbf{q}^h]\!] \cdot \{\boldsymbol{\delta}\mathbf{q}^h\} \,\mathrm{d}\mathbf{s} \ , \end{aligned}$$

we find that:

$$\begin{aligned} \left\langle \nabla_{\mathbf{q}^h} \mathcal{J}^h,\, \boldsymbol{\delta}\mathbf{q}^h \right\rangle_\Omega \ &:= \ \int_\Omega \nabla_{\mathbf{q}^h} \mathcal{J}^h\big|_\Omega \, \boldsymbol{\delta}\mathbf{q}^h \,\mathrm{d}\mathbf{x} + \int_\Gamma \nabla_{\mathbf{q}^h} \mathcal{J}^h\big|_\Gamma \, \boldsymbol{\delta}\mathbf{q}^h \,\mathrm{d}\mathbf{s} \\ &\quad + \int_{\Gamma^q_{\mathcal{I}}} \nabla_{\mathbf{q}^h} \mathcal{J}^h\big|_{\Gamma^q_{\mathcal{I}}} \, \boldsymbol{\delta}\mathbf{q}^h \,\mathrm{d}\mathbf{s} \ . \end{aligned} \tag{6.43}$$

The cell, boundary and interior face contributions to the gradient, are computed, respectively,

through the following equations:

$$
\begin{aligned}
\int_\Omega \nabla_{\mathbf{q}^h}\mathcal{J}^h\big|_\Omega\ \boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{x} \;&:=\; -\int_\Omega \Delta\mathbf{q}^h\,\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{x}+\int_\Omega \Delta\mathbf{q}_B^h\,\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{x}\\
&\quad+\boldsymbol{\beta}\int_\Omega(\mathbf{q}^h-\mathbf{q}_B^h)\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{x}-\int_\Omega \nabla\mathbf{u}^h\cdot\nabla\boldsymbol{\lambda}^h\,\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{x}\,,\\
\int_\Gamma \nabla_{\mathbf{q}^h}\mathcal{J}^h\big|_\Gamma\ \boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{s} \;&:=\; \int_\Gamma\left(\mathbf{u}^h\,(\nabla\boldsymbol{\lambda}^h\cdot\vec{\mathbf{n}})+(\nabla\mathbf{u}^h\cdot\vec{\mathbf{n}})\,\boldsymbol{\lambda}^h\right)\,\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{s}\\
&\quad+\int_\Gamma(\nabla\mathbf{q}^h\cdot\vec{\mathbf{n}}\;-\;\nabla\mathbf{q}_B^h\cdot\vec{\mathbf{n}})\,\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{s}-\int_\Gamma \mathbf{g}^h\,\nabla\boldsymbol{\lambda}^h\cdot\vec{\mathbf{n}}\,\boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{s}\,,\\
\int_{\Gamma_{\mathcal{I}}^q} \nabla_{\mathbf{q}^h}\mathcal{J}^h\big|_{\Gamma_{\mathcal{I}}^q}\ \boldsymbol{\delta}\mathbf{q}^h\,\mathrm{d}\mathbf{s} \;&:=\; \int_{\Gamma_{\mathcal{I}}^q}[\![\nabla\mathbf{q}^h]\!]\cdot\{\boldsymbol{\delta}\mathbf{q}^h\}\,\mathrm{d}\mathbf{s}+\int_{\Gamma_{\mathcal{I}}^q}\phi\,[\![\boldsymbol{\delta}\mathbf{q}^h]\!]\cdot[\![\mathbf{q}^h]\!]\,\mathrm{d}\mathbf{s}\\
&\quad+\int_{\Gamma_{\mathcal{I}}^q}\{\nabla\mathbf{q}^h\}\cdot[\![\boldsymbol{\delta}\mathbf{q}^h]\!]\,\mathrm{d}\mathbf{s}-\int_{\Gamma_{\mathcal{I}}^q}\{\nabla\mathbf{q}_B^h\}\cdot[\![\boldsymbol{\delta}\mathbf{q}^h]\!]\,\mathrm{d}\mathbf{s}\\
&\quad+\int_{\Gamma_{\mathcal{I}}}\left([\![\mathbf{u}^h]\!]\cdot\{\boldsymbol{\delta}\mathbf{q}^h\,\nabla\boldsymbol{\lambda}^h\}+\{\boldsymbol{\delta}\mathbf{q}^h\,\nabla\mathbf{u}^h\}\cdot[\![\boldsymbol{\lambda}^h]\!]\right)\,\mathrm{d}\mathbf{s}\,.
\end{aligned}
$$

Lemma 6.3.1. *If we have convergence of the discrete primal and dual solutions, then*

$$\lim_{h\to 0}\nabla_{\mathbf{q}^h}\mathcal{J}^h=\nabla_{\mathbf{q}}\mathcal{J}\,,$$

i.e., the reduced discrete gradient is asymptotically consistent.

Proof. Use equations (6.7), (6.43), and Lemma 6.2.1. □

Remark In addition to cell and boundary components, which are present in the continuous gradient equation (6.7), the discrete gradient (6.43) is also influenced by traces on the interior element faces. Even though the contribution defined on interior faces $\nabla_{\mathbf{q}^h}\mathcal{J}^h\big|_{\Gamma_{\mathcal{I}}^q}$ vanishes in the limit of the discretization, it is nonzero for fixed $h>0$.

Computation of the discrete gradient

We now discuss the computation of the reduced gradient from an implementation perspective. The more straightforward, albeit significantly less efficient approach is the component–wise calculation of gradient entries. Let N be the total number of degrees of freedom (DoFs) for the mesh $\mathcal{T}_h^q$, and the size of the discrete solution $\mathbf{q}^h$. It follows that $N=I\times R$, where I is the total number of elements defined on $\mathcal{T}_h^q$, and the constant R denotes the number of degrees of freedom per element.

Assume the following ordering in the components of the discrete reduced gradient: the $\big((i-1)\times R+j\big)$-th entry corresponds to the j-th DoF defined in the i-th mesh element κ_i^q. Note that all the discontinuous Galerkin DoFs are defined inside the mesh elements, given that the numerical solution is discontinuous at the inter-element boundaries.

Let

$$\boldsymbol{\delta}\mathbf{q}^h(\mathbf{x}) := \left\{ \begin{array}{rcl} \boldsymbol{\chi}_j^q(\mathbf{x}) & , & \mathbf{x}\in\kappa_i^q \\ 0 & , & \mathbf{x}\notin\kappa_i^q \end{array} \right. , \tag{6.44}$$

in (6.43), and any indices $i = 1\dots I$, and $j = 1\dots R$. Then

$$\big\langle \nabla_{\mathbf{q}^h}\mathcal{J}^h,\, \boldsymbol{\delta}\mathbf{q}^h\big\rangle_\Omega = \big(\nabla_{\mathbf{q}^h}\mathcal{J}^h\big)_{(i-1)\times R+j} \ .$$

With this observation, $I\times R$ evaluations of the functionals (6.43) are needed for a complete gradient computation. For practical values of I, we need a significantly better approach.

The *sparse* structure of the test functional (6.44) can be exploited to dramatically increase the efficiency of the gradient computation. To see this, use equation (6.44) in (6.43), to get:

$$\big(\nabla_{\mathbf{q}^h}\mathcal{J}^h\big)_{(i-1)\times R+j} := \mathcal{G}_\Omega + \mathcal{G}_\Gamma + \mathcal{G}_{\Gamma_\mathcal{I}} \ , \tag{6.45}$$

where the volume and boundary integrals reduce to

$$\mathcal{G}_\Omega := \int_{\kappa_i^q} \nabla_{\mathbf{q}^h}\mathcal{J}^h\big|_\Omega \ \chi_j^q \,\mathrm{d}\mathbf{x} \ ,$$

and

$$\mathcal{G}_\Gamma := \sum_{e\in\Gamma\cap\partial\kappa_i^q} \int_e \nabla_{\mathbf{q}^h}\mathcal{J}^h\big|_\Gamma \ \chi_j^q \,\mathrm{d}\mathbf{s} \ .$$

The inner face integrals warrant further examination. Consider for brevity only the integral below (the rest of the inner face terms in (6.43) are treated similarly):

$$\begin{aligned} \int_{\Gamma_\mathcal{I}^q} [\![\nabla\mathbf{q}^h]\!]\cdot\{\boldsymbol{\delta}\mathbf{q}^h\}\,\mathrm{d}\mathbf{s} &= \sum_{e\in\Gamma_\mathcal{I}^q}\int_e [\![\nabla\mathbf{q}^h]\!]\cdot\{\boldsymbol{\delta}\mathbf{q}^h\}\,\mathrm{d}\mathbf{s} \\ &= \sum_{e\in\Gamma_\mathcal{I}^q}\frac{1}{2}\int_e \big(\nabla\mathbf{q}_+^h\cdot\vec{\mathbf{n}} - \nabla\mathbf{q}_-^h\cdot\vec{\mathbf{n}}\big)\ (\boldsymbol{\delta}\mathbf{q}_+^h + \delta\mathbf{q}_-^h)\,\mathrm{d}\mathbf{s} \ , \end{aligned}$$

where we let the edge $e = \kappa_+\cap\kappa_-$. Let $\kappa_i^q \equiv \kappa_+$. Then the integral above reduces to

$$\int_{\Gamma_\mathcal{I}^q} [\![\nabla\mathbf{q}^h]\!]\cdot\{\boldsymbol{\delta}\mathbf{q}^h\}\,\mathrm{d}\mathbf{s} = \sum_{e\in\partial\kappa_i^q\cap\Gamma_\mathcal{I}^q}\int_e \big(\nabla\mathbf{q}_+^h\cdot\vec{\mathbf{n}} - \nabla\mathbf{q}_-^h\cdot\vec{\mathbf{n}}\big)\ \boldsymbol{\delta}\mathbf{q}_+^h\,\mathrm{d}\mathbf{s} \ .$$

This locality implies that all three integrals in (6.45) can be assembled in parallel over all DoF indices $1\dots N$. Hence, the cost of a reduced gradient computation becomes comparable to that of evaluating a single gradient component on the triangulation $\mathcal{T}_h^q$ using equation (6.43).

6.4 Targeted *a posteriori* error estimation based on an error functional

We follow Becker and Vexler [110], and generalize their error estimate to cover the case where the parameter space is infinite dimensional. From this we derive a computational procedure applicable to our formulation using discrete adjoints.

Consider an error functional $E[\mathbf{q}] : \mathcal{Q} \to \mathbb{R}$. The gradient of E is defined by identification from

$$\langle \nabla_{\mathbf{q}} E, \boldsymbol{\delta q} \rangle = E_{\mathbf{q}}[\mathbf{q}](\boldsymbol{\delta q}) \ , \ \forall\, \boldsymbol{\delta q} \in \mathcal{Q} \ .$$

As in [110], we seek an error representation of the type:

$$E[\mathbf{q}] - E[\mathbf{q}^h] \approx e_h + \text{h.o.t.} \ ,$$

where the higher order terms are ignored. Note that we assume an infinite dimensional parameter space $\mathcal{Q}$. Let $\mathcal{M}$ denote the associated Lagrangian for the functional E [110]:

$$\begin{aligned} \mathcal{M}[\boldsymbol{\xi}, \boldsymbol{\sigma}] &= E[\mathbf{q}] - \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}\right](\boldsymbol{\sigma}_{\mathbf{u}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}}, \boldsymbol{\sigma}_{\mathbf{q}}) \\ &= E[\mathbf{q}] - \mathcal{L}_{\mathbf{q}}\left[\boldsymbol{\xi}\right](\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{u}}\left[\boldsymbol{\xi}\right](\boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{L}_{\boldsymbol{\lambda}}\left[\boldsymbol{\xi}\right](\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) \\ &= E[\mathbf{q}] - \mathcal{L}_{\mathbf{q}}\left[\boldsymbol{\xi}\right](\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{u}}\left[\boldsymbol{\xi}\right](\boldsymbol{\sigma}_{\mathbf{u}}) \end{aligned}$$

where the last equality holds true if $\mathbf{u} = U[\mathbf{q}]$. The fist variation of this Lagrangian is

$$\begin{aligned} \mathcal{M}_{\boldsymbol{\xi}}[\boldsymbol{\xi}, \boldsymbol{\sigma}](\boldsymbol{\delta\xi}) + \mathcal{M}_{\boldsymbol{\sigma}}[\boldsymbol{\xi}, \boldsymbol{\psi}](\boldsymbol{\delta\sigma}) &= E_{\mathbf{q}}[\mathbf{q}](\boldsymbol{\delta q}) \\ &\quad -\mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}\left[\boldsymbol{\xi}\right](\boldsymbol{\delta u},\, \boldsymbol{\delta\lambda},\, \boldsymbol{\delta q}\,;\, \boldsymbol{\sigma}_{\mathbf{u}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}}, \boldsymbol{\sigma}_{\mathbf{q}}) \\ &\quad -\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}\right](\boldsymbol{\delta\sigma}_{\mathbf{u}}, \boldsymbol{\delta\sigma}_{\boldsymbol{\lambda}}, \boldsymbol{\delta\sigma}_{\mathbf{q}}) \ . \end{aligned}$$

The last term vanishes for an optimal solution $(\mathbf{u},\, \boldsymbol{\lambda},\, \mathbf{q}) = \boldsymbol{\xi}_*$. The Lagrangian variation equation about the optimal point can be expanded as follows

$$\begin{aligned} \mathcal{M}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*, \boldsymbol{\sigma}](\boldsymbol{\delta\xi}) &= E_{\mathbf{q}}[\mathbf{q}_*](\boldsymbol{\delta q}) \\ &\quad -\mathcal{L}_{\mathbf{q},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\delta q}\,;\, \boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{q},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\delta u}\,;\, \boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{q},\boldsymbol{\lambda}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\delta\lambda}\,;\, \boldsymbol{\sigma}_{\mathbf{q}}) \\ &\quad -\mathcal{L}_{\mathbf{u},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\delta q}\,;\, \boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{L}_{\mathbf{u},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\delta u}\,;\, \boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{L}_{\mathbf{u},\boldsymbol{\lambda}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\delta\lambda}\,;\, \boldsymbol{\sigma}_{\mathbf{u}}) \\ &= \left\{E_{\mathbf{q}}[\mathbf{q}_*] - \mathcal{L}_{\mathbf{q},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{u},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}})\right\}(\boldsymbol{\delta q}) \\ &\quad -\left\{\mathcal{L}_{\mathbf{q},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) + \mathcal{L}_{\mathbf{u},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}})\right\}(\boldsymbol{\delta u}) \\ &\quad -\left\{\mathcal{L}_{\mathbf{q},\boldsymbol{\lambda}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) + \mathcal{L}_{\mathbf{u},\boldsymbol{\lambda}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}})\right\}(\boldsymbol{\delta\lambda}) \end{aligned} \tag{6.46}$$

Using (1.6), and a compact notation for the nonlinear arguments, we have that

$$\begin{aligned} \mathcal{M}_{\boldsymbol{\xi}}(\boldsymbol{\delta\xi}) + \mathcal{M}_{\boldsymbol{\sigma}}(\boldsymbol{\delta\sigma}) &= \left\{E_{\mathbf{q}}\left[\boldsymbol{\xi}_*\right] - \mathcal{J}_{\mathbf{u},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{J}_{\mathbf{q},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}})\right\}(\boldsymbol{\delta q}) \\ &\quad + \left\{\mathcal{A}_{\mathbf{q},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) + \mathcal{A}_{\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) + \mathcal{A}_{\mathbf{u},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}})\right\}(\boldsymbol{\delta q}) \\ &\quad -\left\{\mathcal{J}_{\mathbf{u},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{J}_{\mathbf{q},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}})\right\}(\boldsymbol{\delta u}) \\ &\quad -\left\{\mathcal{A}_{\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) + \mathcal{A}_{\mathbf{u},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{A}_{\mathbf{q},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}})\right\}(\boldsymbol{\delta u}) \\ &\quad +\left\{\mathcal{A}_{\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{A}_{\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}})\right\}(\boldsymbol{\delta\lambda}) \end{aligned}$$

The variation (6.46) of the Lagrangian $\mathcal{M}$ is zero at its stationary points. Note that if $(\boldsymbol{\xi}_*, \boldsymbol{\sigma}_*)$ is a stationary point of $\mathcal{M}$, then $\boldsymbol{\xi}_*$ is a stationary point of $\mathcal{L}$, since the first order optimality conditions for $\mathcal{L}$ are imposed as constraints in $\mathcal{M}$. The equation for $\boldsymbol{\delta q}$ reads

$$\begin{aligned} 0 &= E_{\mathbf{q}}\left[\boldsymbol{\xi}_*\right] - \mathcal{J}_{\mathbf{u},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{J}_{\mathbf{q},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) + \mathcal{A}_{\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) \\ &\quad + \mathcal{A}_{\mathbf{q},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) + \mathcal{A}_{\mathbf{u},\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) \,. \end{aligned} \tag{6.47}$$

The equation for $\boldsymbol{\delta\lambda}$ is the tangent linear model evaluated at the optimal solution

$$0 = \mathcal{A}_{\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{A}_{\mathbf{q}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) \quad \Leftrightarrow \quad \boldsymbol{\sigma}_{\mathbf{u}} = U'[\mathbf{q}_*]\,\boldsymbol{\sigma}_{\mathbf{q}} \,. \tag{6.48}$$

The equation for $\boldsymbol{\delta u}$ is the second order adjoint model evaluated at the optimal solution:

$$\begin{aligned} 0 &= \mathcal{J}_{\mathbf{u},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{J}_{\mathbf{q},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) \\ &\quad - \mathcal{A}_{\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) - \mathcal{A}_{\mathbf{u},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{A}_{\mathbf{q},\mathbf{u}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) \,. \end{aligned} \tag{6.49}$$

To derive the computational procedure we revisit the reduced cost function, which is equal to the reduced form of the Lagrangian (1.5) for any $\mathbf{q}$ and for any $\boldsymbol{\lambda}$:

$$\begin{aligned} j[\mathbf{q};\boldsymbol{\lambda}] &= \mathcal{L}\left[\boldsymbol{\xi}_2\right] = \mathcal{J}\left[U[\mathbf{q}],\,\mathbf{q}\right] - \mathcal{A}\left[U[\mathbf{q}],\mathbf{q}\right](\boldsymbol{\lambda}) \,, \\ &\quad \boldsymbol{\xi}_2[\mathbf{q},\boldsymbol{\lambda}] = [\mathbf{q}, U[\mathbf{q}], \boldsymbol{\lambda}] \;, \quad \forall \mathbf{q} \in \mathcal{Q} \,,\; \boldsymbol{\lambda} \in \mathcal{U} \,. \end{aligned}$$

The notation $\boldsymbol{\xi}_2$ reminds that this $\boldsymbol{\xi}$ depends on only two parameters, $\mathbf{q}$ and $\boldsymbol{\lambda}$. The reduced gradient is

$$\begin{aligned} j_{\mathbf{q}}[\mathbf{q};\boldsymbol{\lambda}](\phi) &= \mathcal{L}_{\mathbf{q}}[\boldsymbol{\xi}_2](\phi) + \mathcal{L}_{\mathbf{u}}[\boldsymbol{\xi}_2]\left(U'[\mathbf{q}]\phi\right) + \mathcal{L}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}_2]\left(\Lambda'[\mathbf{q}]\phi\right) \\ &= \mathcal{L}_{\mathbf{q}}[\boldsymbol{\xi}_2](\phi) + \mathcal{L}_{\mathbf{u}}[\boldsymbol{\xi}_2]\left(U'[\mathbf{q}]\phi\right) \,. \end{aligned} \tag{6.50}$$

The last term is the forward equation and vanishes identically since $\mathbf{u} = U[\mathbf{q}]$ in $\boldsymbol{\xi}_2$. Note that

$$j_{\mathbf{q},\boldsymbol{\lambda}}[\mathbf{q};\boldsymbol{\lambda}](\boldsymbol{\delta\lambda},\phi) = \mathcal{L}_{\mathbf{q},\boldsymbol{\lambda}}[\boldsymbol{\xi}_2](\boldsymbol{\delta\lambda},\phi) + \mathcal{L}_{\mathbf{u},\boldsymbol{\lambda}}[\boldsymbol{\xi}_2]\left(\boldsymbol{\delta\lambda}, U'[\mathbf{q}]\phi\right) \,. \tag{6.51}$$

The reduced Hessian is

$$\begin{aligned} j_{\mathbf{q},\mathbf{q}}[\mathbf{q};\boldsymbol{\lambda}](\psi,\phi) &= \mathcal{L}_{\mathbf{q},\mathbf{q}}[\boldsymbol{\xi}_2](\psi,\phi) + \mathcal{L}_{\mathbf{q},\mathbf{u}}[\boldsymbol{\xi}_2](U'[\mathbf{q}]\psi,\phi) \\ &\quad + \mathcal{L}_{\mathbf{u},\mathbf{q}}[\boldsymbol{\xi}_2]\left(\psi, U'[\mathbf{q}]\phi\right) + \mathcal{L}_{\mathbf{u},\mathbf{u}}[\boldsymbol{\xi}_2]\left(U'[\mathbf{q}]\psi, U'[\mathbf{q}]\phi\right) \\ &\quad + \mathcal{L}_{\mathbf{u}}[\boldsymbol{\xi}_2]\left(U''[\mathbf{q}](\psi,\phi)\right) \,. \end{aligned}$$

With $\boldsymbol{\lambda} = \Lambda(\mathbf{q})$ the adjoint equation $\mathcal{L}_{\mathbf{u}} = 0$ makes the last term vanish identically. Let $\boldsymbol{\xi}_1[\mathbf{q}] = (\mathbf{q}, U[\mathbf{q}], \Lambda(\mathbf{q}))$, which depends on just one parameter, $\mathbf{q}$. The reduced Hessian reads

$$\begin{aligned} j_{\mathbf{q},\mathbf{q}}[\mathbf{q}](\psi,\phi) &= \mathcal{L}_{\mathbf{q},\mathbf{q}}[\boldsymbol{\xi}_1](\psi,\phi) + \mathcal{L}_{\mathbf{q},\mathbf{u}}[\boldsymbol{\xi}_1](U'[\mathbf{q}]\psi,\phi) \\ &\quad + \mathcal{L}_{\mathbf{u},\mathbf{q}}[\boldsymbol{\xi}_1]\left(\psi, U'[\mathbf{q}]\phi\right) + \mathcal{L}_{\mathbf{u},\mathbf{u}}[\boldsymbol{\xi}_1]\left(U'[\mathbf{q}]\psi, U'[\mathbf{q}]\phi\right) \end{aligned}$$

Consider the variation of the Lagrangian $\mathcal{M}_{\boldsymbol{\xi}}$ (6.46) along the direction $\boldsymbol{\delta\xi}_0$ given by an arbitrary $\boldsymbol{\delta q}$, $\boldsymbol{\delta u} = U'[\mathbf{u}_*]\boldsymbol{\delta q}$, and $\boldsymbol{\delta\lambda} = 0$. Using (6.48) we replace $\boldsymbol{\sigma}_{\mathbf{u}} = U'[\mathbf{q}_*]\,\boldsymbol{\sigma}_{\mathbf{q}}$. Note that $\boldsymbol{\xi}_1[\mathbf{q}_*] = \boldsymbol{\xi}_*$. With these substitutions, the variation of the Lagrangian (6.46) reads

$$\begin{aligned}\mathcal{M}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*,\boldsymbol{\sigma}_*]\,(\boldsymbol{\delta\xi}_0) &= E_{\mathbf{q}}[\mathbf{q}_*]\,(\boldsymbol{\delta q}) - \mathcal{L}_{\mathbf{q},\mathbf{q}}\,[\boldsymbol{\xi}_*]\,(\boldsymbol{\delta q},\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{u},\mathbf{q}}\,[\boldsymbol{\xi}_*]\,(\boldsymbol{\delta q}, U'[\mathbf{q}_*]\,\boldsymbol{\sigma}_{\mathbf{q}})\\ &\quad -\mathcal{L}_{\mathbf{q},\mathbf{u}}\,[\boldsymbol{\xi}_*]\,(U'[\mathbf{u}_*]\boldsymbol{\delta q},\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{L}_{\mathbf{u},\mathbf{u}}\,[\boldsymbol{\xi}_*]\,(U'[\mathbf{u}_*]\boldsymbol{\delta q}, U'[\mathbf{q}_*]\,\boldsymbol{\sigma}_{\mathbf{q}})\\ &= E_{\mathbf{q}}[\mathbf{q}_*]\,(\boldsymbol{\delta q}) - j_{\mathbf{q},\mathbf{q}}[\mathbf{q}_*]\,(\boldsymbol{\delta q},\boldsymbol{\sigma}_{\mathbf{q}})\end{aligned}$$

The variation of the Lagrangian $\mathcal{M}$ (6.46) along the direction $\boldsymbol{\delta\xi}_0$ must be zero for any $\boldsymbol{\delta q}$, which leads to the following equation for the multiplier $\boldsymbol{\sigma}_{\mathbf{q}}$

$$j_{\mathbf{q},\mathbf{q}}[\mathbf{q}_*]\,(\phi,\boldsymbol{\sigma}_{\mathbf{q}}) = E_{\mathbf{q}}[\mathbf{q}_*]\,(\phi)\;,\quad \forall\phi\in\mathcal{Q}\,. \tag{6.52}$$

Equation (6.52) can be solved by using a quasi-Newton approximation of the reduced Hessian. This approximation is based on the sequence of reduced gradients obtained during the optimization. The computational procedure is given in Algorithm 6.4.1.

Algorithm 6.4.1 Error estimation using the second order adjoint solution

1: Solve equation (6.52) for $\boldsymbol{\sigma}_{\mathbf{q}}$ using a quasi-Newton approximation of $j_{\mathbf{q},\mathbf{q}}$.
2: Given $\boldsymbol{\sigma}_{\mathbf{q}}$, solve the tangent linear model (6.48) to obtain $\boldsymbol{\sigma}_{\mathbf{u}}$.
3: Given $\boldsymbol{\sigma}_{\mathbf{q}}$ and $\boldsymbol{\sigma}_{\mathbf{u}}$, solve the second order adjoint model (6.49) to obtain $\boldsymbol{\sigma}_{\boldsymbol{\lambda}}$.

We recall the error representation of Becker and Vexler [110, Theorem 1], and extend it to the infinite dimensional case as well. The error representation based on the error functional reads:

$$\begin{aligned}E[\mathbf{q}_*] - E[\mathbf{q}_*^h] &= \frac{1}{2}\boldsymbol{\rho}_{\mathbf{u}}[\boldsymbol{\xi}_*^h]\,(\Delta\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) + \frac{1}{2}\boldsymbol{\rho}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}_*^h]\,(\Delta\boldsymbol{\sigma}_{\mathbf{u}}) + \frac{1}{2}\boldsymbol{\rho}_{\mathbf{q}}[\boldsymbol{\xi}_*^h]\,(\Delta\boldsymbol{\sigma}_{\mathbf{q}})\\ &\quad +\frac{1}{2}\boldsymbol{\rho}_{\boldsymbol{\sigma}_{\mathbf{u}}}[\boldsymbol{\xi}_*^h,\boldsymbol{\sigma}_*^h]\,(\Delta\boldsymbol{\lambda}) + \frac{1}{2}\boldsymbol{\rho}_{\boldsymbol{\sigma}_{\boldsymbol{\lambda}}}[\boldsymbol{\xi}_*^h,\boldsymbol{\sigma}_*^h]\,(\Delta\mathbf{u})\\ &\quad +\frac{1}{2}\boldsymbol{\rho}_{\boldsymbol{\sigma}_{\mathbf{q}}}[\boldsymbol{\xi}_*^h,\boldsymbol{\sigma}_*^h]\,(\Delta\mathbf{q})\,.\end{aligned} \tag{6.53}$$

The residuals $\boldsymbol{\rho}$ are given by the Lagrangian derivatives evaluated at the discrete solutions:

$$\begin{aligned}\boldsymbol{\rho}_{\mathbf{u}}[\boldsymbol{\xi}_*^h]\,(\cdot) &:= -\mathcal{A}[\mathbf{u}^h,\mathbf{q}^h](\cdot)\,,\\ \boldsymbol{\rho}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}_*^h]\,(\cdot) &:= \mathcal{J}_{\mathbf{u}}[\mathbf{u}^h,\mathbf{q}^h](\cdot) - \mathcal{A}_{\mathbf{u}}[\mathbf{u}^h,\mathbf{q}^h](\cdot,\boldsymbol{\lambda}^h)\,,\\ \boldsymbol{\rho}_{\mathbf{q}}[\boldsymbol{\xi}_*^h]\,(\cdot) &:= \mathcal{J}_{\mathbf{q}}[\mathbf{u}^h,\mathbf{q}^h](\cdot) - \mathcal{A}_{\mathbf{q}}[\mathbf{u}^h,\mathbf{q}^h](\cdot,\boldsymbol{\lambda}^h)\,,\end{aligned}$$

and

$$\begin{aligned}&\boldsymbol{\rho}_{\boldsymbol{\sigma}_{\mathbf{u}}}[\boldsymbol{\xi}_*^h,\boldsymbol{\sigma}_*^h]\,(\boldsymbol{\sigma}_{\mathbf{u}}) + \boldsymbol{\rho}_{\boldsymbol{\sigma}_{\mathbf{q}}}[\boldsymbol{\xi}_*^h,\boldsymbol{\sigma}_*^h]\,(\boldsymbol{\sigma}_{\mathbf{q}}) + \boldsymbol{\rho}_{\boldsymbol{\sigma}_{\boldsymbol{\lambda}}}[\boldsymbol{\xi}_*^h,\boldsymbol{\sigma}_*^h]\,(\boldsymbol{\sigma}_{\boldsymbol{\lambda}})\\ &:= \mathcal{J}_{\mathbf{u},\mathbf{u}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{J}_{\mathbf{q},\mathbf{u}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\mathbf{q}})\\ &\quad - \mathcal{A}_{\mathbf{u}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) - \mathcal{A}_{\mathbf{u},\mathbf{u}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{A}_{\mathbf{q},\mathbf{u}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{A}_{\mathbf{u}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\mathbf{u}}) - \mathcal{A}_{\mathbf{q}}[\boldsymbol{\xi}_*^h](\boldsymbol{\sigma}_{\mathbf{q}})\,.\end{aligned}$$

The simplest way to estimate the residual weights is through local higher–order interpolation [35]. Consider, for example, the error weight

$$\Delta\boldsymbol{\sigma}_{\boldsymbol{\lambda}} := \boldsymbol{\sigma}_{\boldsymbol{\lambda}} - \mathcal{I}_h\boldsymbol{\sigma}_{\boldsymbol{\lambda}} \ .$$

Here $\mathcal{I}_h : \mathcal{U} \to \mathcal{U}_h$ denotes the projection operator onto the discrete solution space. We can approximate the weight $\Delta\boldsymbol{\sigma}_{\boldsymbol{\lambda}}$ through a patch-wise higher–order interpolation of $\boldsymbol{\sigma}^h_{\boldsymbol{\lambda}}$ onto a coarser mesh. Suppose $\boldsymbol{\sigma}^h_{\boldsymbol{\lambda}}$ is defined on a regular mesh $\mathcal{T}^q_h$ (see section 6.2 for details on the notation). The higher-order interpolant $\boldsymbol{\sigma}^H_{\boldsymbol{\lambda}}$ is defined on a coarser mesh $\mathcal{T}^q_H$ that is obtained directly from $\mathcal{T}^q_h$ through pure hierarchical coarsening (e.g., $H := 2h$). Then, for all mesh elements $\kappa \in \mathcal{T}^q_h$, we set:

$$\Delta\boldsymbol{\sigma}_{\boldsymbol{\lambda}}|_{\kappa} \approx \left(\boldsymbol{\sigma}^H_{\boldsymbol{\lambda}} - \boldsymbol{\sigma}^h_{\boldsymbol{\lambda}}\right)\big|_{\kappa} \ .$$

Similar computations are done for the other weights $\Delta\boldsymbol{\sigma}_{\mathbf{u}}$, $\Delta\boldsymbol{\sigma}_{\mathbf{q}}$, etc.

6.4.1 The dual weighted residual (DWR) method

If the error functional is the optimization cost function, $E[\mathbf{q}] = \mathcal{J}[\mathbf{u}, \mathbf{q}]$, then a simpler dual-weighted error estimation is available, cf. Rannacher and Vexler [35].

The continuous and discrete optimality conditions read:

$$\begin{aligned} &\mathcal{L}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*](\boldsymbol{\psi}) = 0 \ , \ \forall\, \boldsymbol{\psi} \in \mathcal{X} \\ &\mathcal{L}^h_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_*\right](\boldsymbol{\psi}^h) = 0 \ , \ \forall\, \boldsymbol{\psi}^h \in \mathcal{X}^h \ . \end{aligned}$$

The error criterion is the difference in cost function values

$$\begin{aligned} \mathcal{J}[\mathbf{q}_*, \mathbf{u}_*] - \mathcal{J}^h\left[\mathbf{q}^h_*, \mathbf{u}^h_*\right] &= \mathcal{L}[\boldsymbol{\xi}_*] - \mathcal{L}^h\left[\boldsymbol{\xi}^h_*\right] \\ &= \left(\mathcal{L}[\boldsymbol{\xi}_*] - \mathcal{L}\left[\boldsymbol{\xi}^h_*\right]\right) + \left(\mathcal{L}\left[\boldsymbol{\xi}^h_*\right] - \mathcal{L}^h\left[\boldsymbol{\xi}^h_*\right]\right) \end{aligned}$$

Let $e := \boldsymbol{\xi}_* - \boldsymbol{\xi}^h_*$. According to [35]:

$$\begin{aligned} \mathcal{L}[\boldsymbol{\xi}_*] - \mathcal{L}\left[\boldsymbol{\xi}^h_*\right] &= \int_0^1 \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_* + se\right](e)\, ds \\ &\qquad -\frac{1}{2}\left\{\underbrace{\mathcal{L}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*](e)}_{=0} + \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_*\right](e)\right\} + \frac{1}{2}\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_*\right](e) \\ &\approx \frac{1}{2}\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_*\right](e) \\ &= \frac{1}{2}\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_*\right]\left(\boldsymbol{\xi}_* - \xi^h\right) \ , \quad \forall \xi^h \in \mathcal{U}^h \times \mathcal{U}^h \times \mathcal{Q}^h \\ &= \frac{1}{2}\mathcal{L}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}^h_*](\boldsymbol{\lambda}_* - \lambda^h) + \frac{1}{2}\mathcal{L}_{\mathbf{u}}[\boldsymbol{\xi}^h_*](\mathbf{u}_* - u^h) + \frac{1}{2}\mathcal{L}_{\mathbf{q}}[\boldsymbol{\xi}^h_*](\mathbf{q}_* - q^h) \\ &= \frac{1}{2}\boldsymbol{\rho}_{\mathbf{u}}[\boldsymbol{\xi}^h_*](\Delta\boldsymbol{\lambda}) + \frac{1}{2}\boldsymbol{\rho}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}^h_*](\Delta\mathbf{u}) + \frac{1}{2}\boldsymbol{\rho}_{\mathbf{q}}[\boldsymbol{\xi}^h_*](\Delta\mathbf{q}) \ . \end{aligned}$$

Here the residuals are the corresponding Lagrangian derivatives, e.g.,

$$\boldsymbol{\rho}_{\mathbf{u}}[\boldsymbol{\xi}_*^h]\,(\phi) = \mathcal{L}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}_*^h]\,(\phi)\,, \quad \boldsymbol{\rho}_{\mathbf{u}}^h[\boldsymbol{\xi}_*^h]\,(\phi) = \mathcal{L}_{\boldsymbol{\lambda}}^h[\boldsymbol{\xi}_*^h]\,(\phi)\,, \quad \text{etc.}$$

These residuals are defined on each element. The residual weights are defined as in equation (6.53).

The remaining piece $\mathcal{L}\left[\boldsymbol{\xi}_*^h\right] - \mathcal{L}^h\left[\boldsymbol{\xi}_*^h\right]$ consists of the additional discrete terms added to the cost function and to the discretized operators, and evaluated at the discrete solution.

Let the desired tolerance be $|\mathcal{J} - \mathcal{J}^h| \leq \text{TOL} \cdot \mathcal{J}^h$. In a well balanced grid each element brings about the same error contribution $\overline{e} = \text{TOL} \cdot \mathcal{J}^h \cdot N_{\text{elem}}^{-1}$ to the total error (by the error equi-distribution principle). A local error indicator e_k is computed on each element $\mathcal{T}_k$, $k = 1, \cdots, N_{\text{elem}}$, by the scalar products of local residuals and local solution differences. The element is flagged for refinement if $e_k \geq f_1\,\overline{e}$, and for coarsening if $e_k \leq f_2\,\overline{e}$. The safety factors can have values such as $f_1 = 5$ and $f_2 = 0.2$.

6.4.2 Norm error

Similar to [172], we derive error bounds on the norm of the difference. The following assumptions are made. There exist $\zeta, \gamma > 0$ such that

$$\begin{aligned}
&\left\|\mathbf{q}_* - \mathbf{q}_*^h\right\|_{\mathcal{Q}} \leq \zeta\,; \\
&j_{\mathbf{q},\mathbf{q}}[\mathbf{q}]\,(\phi,\phi) \geq \gamma\,\|\phi\|_{\mathcal{Q}}^2\,; \quad \forall \phi \in \mathcal{Q}\,; \quad \forall \mathbf{q} \in \mathcal{Q}\,,\ \|\mathbf{q} - \mathbf{q}_*\|_{\mathcal{Q}} \leq \zeta\,.
\end{aligned}$$

Then the following results are established

$$\frac{\gamma}{2}\,\left\|\mathbf{q}_* - \mathbf{q}_*^h\right\|_{\mathcal{Q}}^2 \leq j(\mathbf{q}_*^h) - j(\mathbf{q}_*)\,.$$

and

$$\begin{aligned}
\gamma\,\left\|\mathbf{q}_* - \mathbf{q}_*^h\right\|_{\mathcal{Q}}^2 \;\leq\; & \boldsymbol{\rho}_{\mathbf{u}}[\boldsymbol{\xi}_*^h]\,\left(\boldsymbol{\lambda}_* - \Lambda[\mathbf{q}_*^h] - \lambda^h\right) \\
& + \boldsymbol{\rho}_{\boldsymbol{\lambda}}[\boldsymbol{\xi}_*^h]\,\left(\mathbf{u}_* - U[\mathbf{q}_*^h] - u^h\right) + \boldsymbol{\rho}_{\mathbf{q}}[\boldsymbol{\xi}_*^h]\,\left(\mathbf{q}_* - q^h\right)\,,
\end{aligned}$$

for any discrete variables λ^h, u^h, q^h from the respective discrete spaces. $U[\mathbf{q}_*^h]$ and $\Lambda[\mathbf{q}_*^h]$ are the solutions of the continuous forward and adjoint problems corresponding to the optimal discrete parameter. An auxiliary optimization is performed on a finer grid. The first iteration stars with an interpolated value of $\mathbf{q}_*^h$ and produces $U[\mathbf{q}_*^h]$ and $\Lambda[\mathbf{q}_*^h]$. After a number of iterations the auxiliary optimization problem produces approximations of $\mathbf{u}_*$ and $\boldsymbol{\lambda}_*$. The projection out of the discrete space is performed as above.

6.5 More on error estimation via an error functional

We follow Becker and Vexler [110], and seek an error representation of the type:

$$E[\mathbf{q}] - E[\mathbf{q}^h] \approx e_h + \text{h.o.t.} ,$$

where the higher order terms are ignored. Consider now a perturbation to the optimality conditions, in the form of a small residual $\boldsymbol{\rho}$. The perturbed optimality conditions have the solution $\boldsymbol{\xi}_*^{\boldsymbol{\rho}}$

$$\left(\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*^{\boldsymbol{\rho}}\right] + \boldsymbol{\rho}\right)(\boldsymbol{\psi}) = 0 , \quad \forall\, \boldsymbol{\psi} \in \mathcal{X}.$$

For the unperturbed case $\boldsymbol{\rho} = 0$ we obtain the solution $\boldsymbol{\xi}_*^0 = \boldsymbol{\xi}_*$. Let $\mathcal{M}^{\boldsymbol{\rho}}$ denote the associated Lagrangian for the functional E, corresponding to the perturbed optimality conditions [110]:

$$\mathcal{M}^{\boldsymbol{\rho}}[\boldsymbol{\xi}, \boldsymbol{\sigma}] \;=\; E[\boldsymbol{\xi}] - \left(\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}\right] + \boldsymbol{\rho}\right)(\boldsymbol{\sigma}) .$$

Let $(\boldsymbol{\xi}_*, \boldsymbol{\sigma})$ be a stationary point of $\mathcal{M}^0$ for the unperturbed case $\boldsymbol{\rho} = 0$. This means that

$$\mathcal{M}^0_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*, \boldsymbol{\sigma}](\boldsymbol{\psi}) \;=\; E_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*](\boldsymbol{\psi}) - \mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*\right](\boldsymbol{\psi}, \boldsymbol{\sigma}) , \quad \forall \boldsymbol{\psi} . \tag{6.54}$$

Consider now the residual $\boldsymbol{\rho}$ for which $\boldsymbol{\xi}_*^{\boldsymbol{\rho}} = \boldsymbol{\xi}_*^h$:

$$\boldsymbol{\rho} = \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*\right] - \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*^h\right] = -\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*^h\right] \quad \Rightarrow \quad \left(\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*^h\right] + \boldsymbol{\rho}\right)(\psi) = 0 , \quad \forall \psi .$$

The error in the cost functional reads:

$$\begin{aligned}
E[\boldsymbol{\xi}_*] - E[\boldsymbol{\xi}_*^h] &= \mathcal{M}^0[\boldsymbol{\xi}_*, \boldsymbol{\sigma}] - \mathcal{M}^{\boldsymbol{\rho}}[\boldsymbol{\xi}_*^h, \boldsymbol{\sigma}] \\
&= E[\boldsymbol{\xi}_*] - E[\boldsymbol{\xi}_*^h] - \left(\mathcal{L}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*] - \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*^h\right] - \boldsymbol{\rho}\right)(\boldsymbol{\sigma}) \\
&= E_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*](\boldsymbol{\xi}_* - \boldsymbol{\xi}_*^h) - \mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_*](\boldsymbol{\xi}_* - \boldsymbol{\xi}_*^h, \boldsymbol{\sigma}) + \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle + h.o.t. \\
&= \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle + h.o.t. ,
\end{aligned}$$

where the first two terms disappear due to the stationarity of $\mathcal{M}^0$ condition (6.54).

Combining the equations we are lead to the following error estimate:

$$\begin{aligned}
E[\boldsymbol{\xi}_*^h] - E[\boldsymbol{\xi}_*] &= \mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}_*^h\right](\boldsymbol{\sigma}) + h.o.t. \\
&= \mathcal{A}[\mathbf{u}^h, \mathbf{q}^h](\Delta\boldsymbol{\sigma}_{\mathbf{u}}) \\
&\quad + \mathcal{J}_{\mathbf{u}}[\mathbf{u}^h, \mathbf{q}^h](\Delta\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) - \mathcal{A}_{\mathbf{u}}[\mathbf{u}^h, \mathbf{q}^h](\Delta\boldsymbol{\sigma}_{\boldsymbol{\lambda}}, \boldsymbol{\lambda}^h) \\
&\quad + \mathcal{J}_{\mathbf{q}}[\mathbf{u}^h, \mathbf{q}^h](\Delta\boldsymbol{\sigma}_{\mathbf{q}}) - \mathcal{A}_{\mathbf{q}}[\mathbf{u}^h, \mathbf{q}^h](\Delta\boldsymbol{\sigma}_{\mathbf{q}}, \boldsymbol{\lambda}^h) + h.o.t.
\end{aligned}$$

Remark. We note the following:

1. All the residuals in the above equation are for the continuous operators.

2. By performing the integration by parts in reverse order, we arrive at the strong form (high derivatives of the polynomials inside each element) dot product with the $\boldsymbol{\sigma}$.

3. Each residual can be written as a sum of residuals within each element. Similarly, each element-wise residual can be computed in strong form, by differentiating the polynomials, to arrive at polynomial residuals.

A more accurate estimator is obtained as follows. Define

$$\begin{aligned} e &= \boldsymbol{\xi}_*^h - \boldsymbol{\xi}_* \\ M(s) &= E[\boldsymbol{\xi}_* + se] - \left(\mathcal{L}_{\boldsymbol{\xi}}[\boldsymbol{\xi}_* + se] + s\boldsymbol{\rho}\right)(\boldsymbol{\sigma}) \\ M(1) - M(0) &= \int_0^1 M'(s)\, ds \end{aligned}$$

Therefore we have that

$$\begin{aligned} \mathcal{M}^{\boldsymbol{\rho}}[\boldsymbol{\xi}_*^h, \boldsymbol{\sigma}] - \mathcal{M}^0[\boldsymbol{\xi}_*, \boldsymbol{\sigma}] &= \int_0^1 \left(E_{\boldsymbol{\xi}}[\boldsymbol{\xi}_* + se](e) - \mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_* + se](e, \boldsymbol{\sigma}) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle\right) ds \\ &= -\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle + \frac{1}{2}\left(E_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*](e) - \mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_*](e, \boldsymbol{\sigma})\right) \\ &\quad + \frac{1}{2}\left(E_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*^h](e) - \mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_*^h](e, \boldsymbol{\sigma})\right) + R(e, e, e) \\ &= -\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle + \frac{1}{2}\left(E_{\boldsymbol{\xi}}[\boldsymbol{\xi}_*^h](e) - \mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_*^h](e, \boldsymbol{\sigma})\right) + R(e, e, e) \end{aligned}$$

where the residual of the trapezoidal integration method is of order three. Using the fact that $\boldsymbol{\xi}_*^h = \boldsymbol{\xi}_* + e$, and expanding in Taylor series, leads to:

$$\mathcal{M}^{\boldsymbol{\rho}}[\boldsymbol{\xi}_*^h, \boldsymbol{\sigma}] - \mathcal{M}^0[\boldsymbol{\xi}_*, \boldsymbol{\sigma}] = -\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle + \frac{1}{2} E_{\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_*](e, e) - \frac{1}{2}\mathcal{L}_{\boldsymbol{\xi},\boldsymbol{\xi},\boldsymbol{\xi}}[\boldsymbol{\xi}_*](e, e, \boldsymbol{\sigma}) + h.o.t.$$

where the higher order terms are at least of order three. This analysis reveals the structure of the second order error terms. It also shows that the additional terms considered by Becker are the second order terms. These second order terms are the residual of the optimality equation (6.54) for the optimal discrete solution. This residual is weighed against the error e.

Consider how this applies to an elliptic equation described by

$$\mathcal{A}[\mathbf{q}, \mathbf{u}](\phi) = a[\mathbf{q}](\mathbf{u}, \phi) - \ell(\phi)$$

where a is bilinear, and ℓ is linear. The KKT conditions (1.2a), (1.2b), and (1.2c) read

$$a\left[\mathbf{q}_*\right](\mathbf{u}_*, \boldsymbol{\psi}_{\boldsymbol{\lambda}}) = \ell(\boldsymbol{\psi}_{\boldsymbol{\lambda}}) \ , \ \forall\, \boldsymbol{\psi}_{\boldsymbol{\lambda}} \in \mathcal{U} \ , \tag{6.55a}$$

$$a\left[\mathbf{q}_*\right](\boldsymbol{\psi}_{\mathbf{u}}, \boldsymbol{\lambda}_*) = \mathcal{J}_{\mathbf{u}}\left[\mathbf{u}_*, \mathbf{q}_*\right](\boldsymbol{\psi}_{\mathbf{u}}) \ , \ \forall\, \boldsymbol{\psi}_{\mathbf{u}} \in \mathcal{U} \ , \tag{6.55b}$$

$$a_{\mathbf{q}}\left[\mathbf{q}_*\right](\boldsymbol{\psi}_{\mathbf{q}}, \mathbf{u}_*, \boldsymbol{\lambda}_*) = \mathcal{J}_{\mathbf{q}}\left[\mathbf{u}_*, \mathbf{q}_*\right](\boldsymbol{\psi}_{\mathbf{q}}) \ , \ \forall\, \boldsymbol{\psi}_{\mathbf{q}} \in \mathcal{Q} \ . \tag{6.55c}$$

The error Lagrangian is

$$\begin{aligned}\mathcal{M}^{\boldsymbol{\rho}}[\boldsymbol{\xi},\boldsymbol{\sigma}] &= E[\boldsymbol{\xi}] - a\left[\mathbf{q}\right](\mathbf{u},\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) + \ell(\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) \\ &\quad -a\left[\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{u}},\boldsymbol{\lambda}) + \mathcal{J}_{\mathbf{u}}\left[\mathbf{u},\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{u}}) \\ &\quad -a_{\mathbf{q}}\left[\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{q}},\mathbf{u},\boldsymbol{\lambda}) + \mathcal{J}_{\mathbf{q}}\left[\mathbf{u},\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{q}})\end{aligned}$$

Therefore

$$\begin{aligned}\mathcal{M}^0_{\boldsymbol{\xi}}[\boldsymbol{\xi},\boldsymbol{\sigma}](\boldsymbol{\psi}) &= E_{\mathbf{q}}[\boldsymbol{\xi}](\boldsymbol{\psi}_{\mathbf{q}}) + \mathcal{J}_{\mathbf{q},\mathbf{q}}\left[\mathbf{u},\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{q}},\boldsymbol{\sigma}_{\mathbf{q}}) + \mathcal{J}_{\mathbf{u},\mathbf{q}}\left[\mathbf{u},\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{q}},\boldsymbol{\sigma}_{\mathbf{u}}) \\ &\quad -a_{\mathbf{q}}\left[\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{q}},\mathbf{u},\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) - a_{\mathbf{q}}\left[\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{q}},\boldsymbol{\sigma}_{\mathbf{u}},\boldsymbol{\lambda}) - a_{\mathbf{q},\mathbf{q}}\left[\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{q}},\boldsymbol{\sigma}_{\mathbf{q}},\mathbf{u},\boldsymbol{\lambda}) \\ &\quad +E_{\mathbf{u}}[\boldsymbol{\xi}](\boldsymbol{\psi}_{\mathbf{u}}) + \mathcal{J}_{\mathbf{u},\mathbf{u}}\left[\mathbf{u},\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{u}},\boldsymbol{\sigma}_{\mathbf{u}}) + \mathcal{J}_{\mathbf{q},\mathbf{u}}\left[\mathbf{u},\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{u}},\boldsymbol{\sigma}_{\mathbf{q}}) \\ &\quad -a\left[\mathbf{q}\right](\boldsymbol{\psi}_{\mathbf{u}},\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) - a_{\mathbf{q}}\left[\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{q}},\boldsymbol{\psi}_{\mathbf{u}},\boldsymbol{\lambda}) \\ &\quad -\left\{a\left[\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{u}},\boldsymbol{\phi}_{\boldsymbol{\lambda}}) + a_{\mathbf{q}}\left[\mathbf{q}\right](\boldsymbol{\sigma}_{\mathbf{q}},\mathbf{u},\boldsymbol{\phi}_{\boldsymbol{\lambda}})\right\}\end{aligned}$$

The residual is

$$\begin{aligned}\boldsymbol{\rho}(\boldsymbol{\sigma}) &= -\mathcal{L}_{\boldsymbol{\xi}}\left[\boldsymbol{\xi}^h_*\right](\boldsymbol{\sigma}) \qquad \{\boldsymbol{\sigma}_{\boldsymbol{\lambda}} \in \mathcal{U}\,, \boldsymbol{\sigma}_{\mathbf{u}} \in \mathcal{U}\,, \boldsymbol{\sigma}_{\mathbf{q}} \in \mathcal{Q}\} \\ &= a\left[\mathbf{q}^h_*\right](\mathbf{u}^h_*,\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) - \ell(\boldsymbol{\sigma}_{\boldsymbol{\lambda}}) \quad \{\equiv \boldsymbol{\rho}_{\mathbf{u}}(\boldsymbol{\sigma}_{\boldsymbol{\lambda}})\} \\ &\quad +a\left[\mathbf{q}^h_*\right](\boldsymbol{\sigma}_{\mathbf{u}},\boldsymbol{\lambda}^h_*) - \mathcal{J}_{\mathbf{u}}\left[\mathbf{u}^h_*,\mathbf{q}^h_*\right](\boldsymbol{\sigma}_{\mathbf{u}}) \quad \{\equiv \boldsymbol{\rho}_{\mathbf{u}}(\boldsymbol{\sigma}_{\mathbf{u}})\} \\ &\quad +a_{\mathbf{q}}\left[\mathbf{q}^h_*\right](\boldsymbol{\sigma}_{\mathbf{q}},\mathbf{u}^h_*,\boldsymbol{\lambda}^h_*) - \mathcal{J}_{\mathbf{q}}\left[\mathbf{u}^h_*,\mathbf{q}^h_*\right](\boldsymbol{\sigma}_{\mathbf{q}}) \quad \{\equiv \boldsymbol{\rho}_{\mathbf{q}}(\boldsymbol{\sigma}_{\mathbf{q}})\}\ .\end{aligned}$$

6.6 Numerical results for the coefficient identification problem

Let $\mathcal{H}$ be the identity operator, $\mathbf{o} := \mathcal{H}\mathbf{u}_*$, $\boldsymbol{\beta} = 100$, and $\Omega := [0,\,1] \times [0,\,1]$. The first numerical test makes use of

$$\begin{aligned}\text{Test A:} \qquad \mathbf{q}_B(\mathbf{x}) &:= 1 + x^2 + y^2\,, \\ \mathbf{u}_*(\mathbf{x}) &:= 10\,\exp\left(-\frac{(10x-5)^2}{4}\right)\exp\left(-\frac{(10y-5)^2}{4}\right)\,,\end{aligned} \tag{6.56}$$

and the initial guess

$$\mathbf{q}_0(\mathbf{x}) := 1 + 7x + 7y\,,\ \mathbf{x} \in \Omega\,. \tag{6.57}$$

The second set of experiments uses the following functions:

$$\begin{aligned}\text{Test B:} \qquad \mathbf{q}_B(\mathbf{x}) &:= 5 + \sin\left(\frac{\pi x}{2}\right)\cos(2\pi y)\,, \\ \mathbf{u}_*(\mathbf{x}) &:= 10\,\exp\left(-\frac{(10x-5)^2}{4}\right)\exp\left(-\frac{(10y-5)^2}{4}\right)\,,\end{aligned} \tag{6.58}$$

and a constant initial guess

$$\mathbf{q}_0 := 1 \ . \tag{6.59}$$

For both experiments we choose

$$\mathbf{f}(\mathbf{x}) \ := \ -\nabla \cdot (\mathbf{q}_B \, \nabla \mathbf{u}_*)(\mathbf{x}) \ , \ \forall \, \mathbf{x} \in \Omega \ .$$

Then, $\mathbf{q}_* = \mathbf{q}_B$, and $\mathcal{J}(\mathbf{u}_*, \mathbf{q}_*) = 0$. The discrete solver uses the `deal.II` library [163] for finite element computations. The spaces $\mathcal{U}_h^p$ and $\mathcal{Q}_h^r$ are spanned by the second order Lagrange basis functions ($p = r = 2$). Moreover, $\mathcal{T}_h \equiv \mathcal{T}_h^q$. Note that for the first set (6.56)–(6.57), the exact optimal solution $\mathbf{q}_*$ can be represented exactly by the polynomial basis spanning $\mathcal{Q}_h^r$.

6.6.1 Consistency of the discrete gradient

An important step in adjoint code development is validation of the discrete gradient (6.43). The classic approach to adjoint code validation is through the numerical verification of a truncated Taylor expansion [71]. For smooth $\mathcal{J}_h$, and small perturbations $\boldsymbol{\varepsilon}\, \boldsymbol{\delta}\mathbf{q}^h$ around a reference state $\mathbf{q}^h$, the Taylor theorem gives:

$$\mathcal{J}^h(\mathbf{q}^h + \boldsymbol{\varepsilon}\, \boldsymbol{\delta}\mathbf{q}^h) = \mathcal{J}^h(\mathbf{q}^h) + \boldsymbol{\varepsilon} \, \left\langle \nabla_{\mathbf{q}^h} \mathcal{J}^h, \, \boldsymbol{\delta}\mathbf{q}^h \right\rangle_\Omega + \mathcal{O}(\boldsymbol{\varepsilon}^2) \ .$$

We verify numerically the following limit for small values of ε:

$$\lim_{\boldsymbol{\varepsilon} \to 0} G^h(\boldsymbol{\varepsilon}) := \lim_{\boldsymbol{\varepsilon} \to 0} \frac{\mathcal{J}^h(\mathbf{q}^h + \boldsymbol{\varepsilon}\, \boldsymbol{\delta}\mathbf{q}^h) - \mathcal{J}^h(\mathbf{q}^h)}{\boldsymbol{\varepsilon} \, \left\langle \nabla_{\mathbf{q}^h} \mathcal{J}^h, \, \boldsymbol{\delta}\mathbf{q}^h \right\rangle_\Omega} = 1 \ . \tag{6.60}$$

As shown in figure 6.1, the discrete reduced gradient is found to be numerically consistent. For $\boldsymbol{\varepsilon} < 10^{-11}$, truncation errors start to degrade the quality of the finite difference approximation (6.60).

6.6.2 The numerical behavior of the discrete optimality condition

We now consider the solution of equation (6.26). Figure 6.2(a) shows the decrease with $h \sim \mathrm{DoF}^{-1/2}$ of the integrals I_Γ^1, I_Γ^2, $I_{\Gamma_\mathcal{I}}^1$, and $I_{\Gamma_\mathcal{I}}^2$ in (6.31a)–(6.31d). For $p = r = 2$, and the $\mathcal{C}^\infty(\Omega)$ exact solutions (6.56)–(6.57), the integrals vanish asymptotically at least as fast as $h^{3/2}$, verifying the analytical bounds. Similar results are reported in figure 6.2(b) for the problem (6.58)–(6.59).

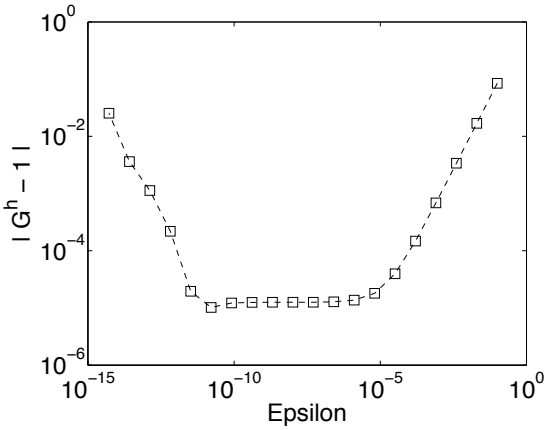


Figure 6.1: Asymptotic consistency test for the discrete gradient $\nabla_{\mathbf{q}^h}\mathcal{J}^h$ using the first order Taylor approximation (6.60), and the functions (6.56)–(6.57).

6.6.3 Multigrid optimization and solution convergence

The multigrid optimization procedure is described in Algorithm 6.6.1. The subscript k indicates the current multigrid iteration. On each mesh level, we keep iterating until there is a sufficient reduction in the cost functional $\mathcal{J}_h$. A practical stopping criterion for the multigrid iterations can be based on the value of the target functional $\mathcal{J}^h$ at the current solution, or some reduced gradient norm.

The algorithm requires the constant relative and absolute tolerance vectors ATOL and RTOL, which decide if there has been sufficient decrease in the cost functional value on the current mesh level. In practice, one may also want to limit the number of optimization iterations on a given mesh, to prevent excessively long run times.

Algorithm 6.4.1 is used for the error estimation at step 9 of Algorithm 6.6.1. The mesh adaptation is performed at step 12, which get executed once per multigrid iteration. The exact choice of error functional is problem dependent. We will choose E to be a weighted average of the optimal solution over Ω:

$$E(\mathbf{x}) = \int_{\Omega} w\,\mathbf{q}\,\mathrm{d}\mathbf{x}\ . \tag{6.61}$$

The weight function $w \in \mathcal{H}^2(\Omega)$ is positive *a.e.* on Ω.

Figure 6.3 shows the convergence of multigrid optimization algorithm on several mesh levels. We plot both the decrease in the cost functional, and in the optimal solution error for the problem (6.56)–(6.57). For this experiment the mesh refinement process is guided by a simple element–wise error estimator based on the scaled gradient $\nabla\mathbf{q}^h$. The order of convergence is consistent with the theoretical expectations: the optimal solution shows cubic convergence

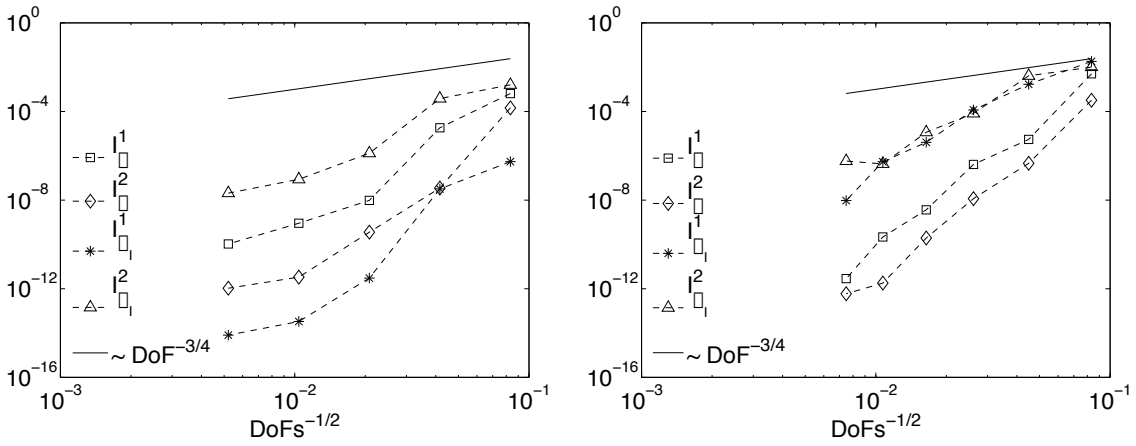


Figure 6.2: Asymptotic behavior of the integrals (6.31a)–(6.31d) for test A (left), and test B (right).

with the mesh size h (for $p = r = 2$). We chose ATOL $= 10^{-10}$, and RTOL $= 10^{-2}$ on all iteration levels.

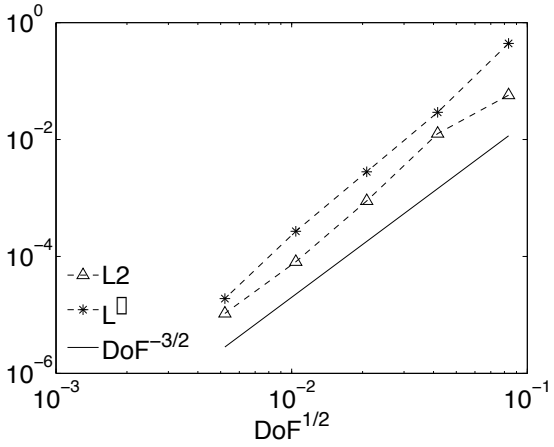


Figure 6.3: Convergence of the multigrid optimization algorithm: test A. The optimal solution error is plotted versus $h \sim \text{DoF}^{-1/2}$. The errors correspond to the optimal solutions on each mesh level.

Remark. The meshes $\mathcal{T}_h^q$ and $\mathcal{T}_h$ may also be locally coarsened in step 12 of Algorithm 6.6.1. The coarsening operation is analogous to that of refinement. Care must be taken to maintain the mesh nesting property for $\mathcal{T}_h^q$ and $\mathcal{T}_h$.

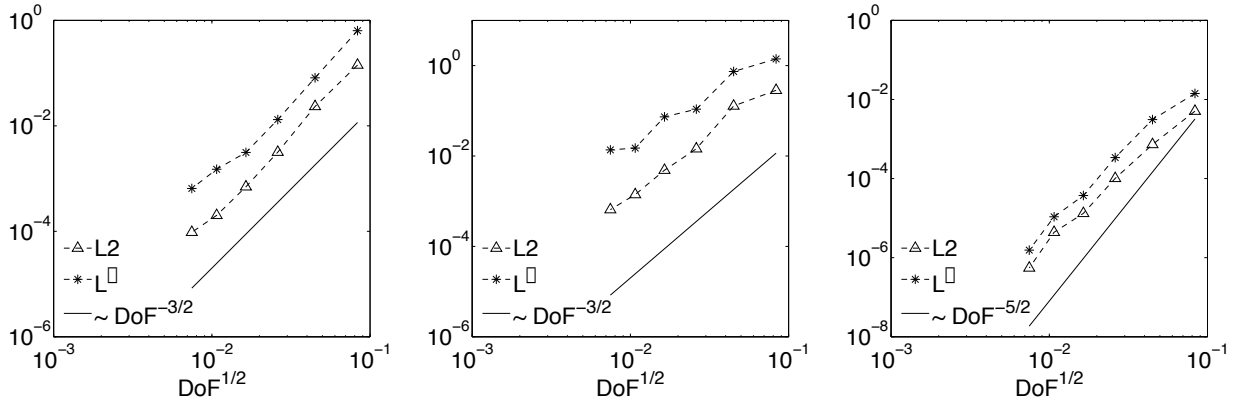


Figure 6.4: Convergence of the discrete optimal (left), primal (center), and dual (right) solutions for test B. The errors correspond to the converged solutions on each mesh level, and are plotted versus $h \sim \mathrm{DoF}^{-1/2}$.

Algorithm 6.6.1 Multigrid optimization with error–driven AMR

Require: ATOL[maxit], RTOL[maxit].

1: Define the initial guess $\mathbf{q}^h$ on $(\mathcal{T}_h^q)_0$, and calculate $\mathbf{u}^h$, $\boldsymbol{\lambda}^h$ on $(\mathcal{T}_h)_0$.
2: Define the error functional $E[\mathbf{q}]$.
For example, $E[\mathbf{q}] := \int_\Omega w(\mathbf{x})\mathbf{q}(\mathbf{x})\,\mathrm{d}\mathbf{x}$, for some positive weight function w.
3: **for** $k = 0$ to maxit $- 1$ **do**
4: Solve the primal model on $(\mathcal{T}_h)_k$ to get $\mathcal{J}_0^h = \mathcal{J}^h(\mathbf{u}^h, \mathbf{q}^h)$.
5: **while** $\mathcal{J}^h/\mathcal{J}_0^h >$ RTOL$[k]$ **and** $\mathcal{J}^h >$ ATOL$[k]$ **do**
6: Run the optimization iterations on $(\mathcal{T}_h)_k$ and $(\mathcal{T}_h^q)_k$, and update the current approximation $\mathbf{q}^h$, and cost functional $\mathcal{J}^h$.
7: Construct approximation to the reduced Hessian $\nabla^2_{\mathbf{q}^h}\mathcal{J}^h$ from the optimization iterates $\mathbf{q}^h$ and gradients $\nabla_{\mathbf{q}^h}\mathcal{J}^h$.
8: **end while**
9: Calculate the element-wise error indicators for all $\kappa^q \in \mathcal{T}_h^q$ using Algorithm 6.4.1.
10: Flag the cells with highest estimated error on $(\mathcal{T}_h^q)_k$ for refinement.
11: For any element $\kappa^q \in (\mathcal{T}_h^q)_k$ flagged for refinement, flag for refinement all elements $\kappa \in (\mathcal{T}_h)_k$ such that $\kappa \subseteq \kappa^q$. This is required to maintain the mesh nesting property.
12: Refine the triangulations to obtain $(\mathcal{T}_h)_{k+1}$ and $(\mathcal{T}_h^q)_{k+1}$.
13: Transfer the current optimal solution approximation $\mathbf{q}^h$ to the new parameter mesh $(\mathcal{T}_h^q)_{k+1}$.
14: **end for**

Figure 6.5 illustrates the sequence of meshes generated by algorithm 6.6.1 with the numerical test B. The observed numerical orders of convergence for $\mathbf{q}^h$, $\mathbf{u}^h$, and $\boldsymbol{\lambda}^h$, are consistent with

the theoretical bounds. Note that the perturbation integrals (6.31a)–(6.31d) - shown in figure 6.6 - are sufficiently small to not affect the convergence of the optimization process, see figure 6.7. However, if the optimal solution error is reduced further, the perturbations may impact the convergence process, as indicated by the a priori error bound (6.39).

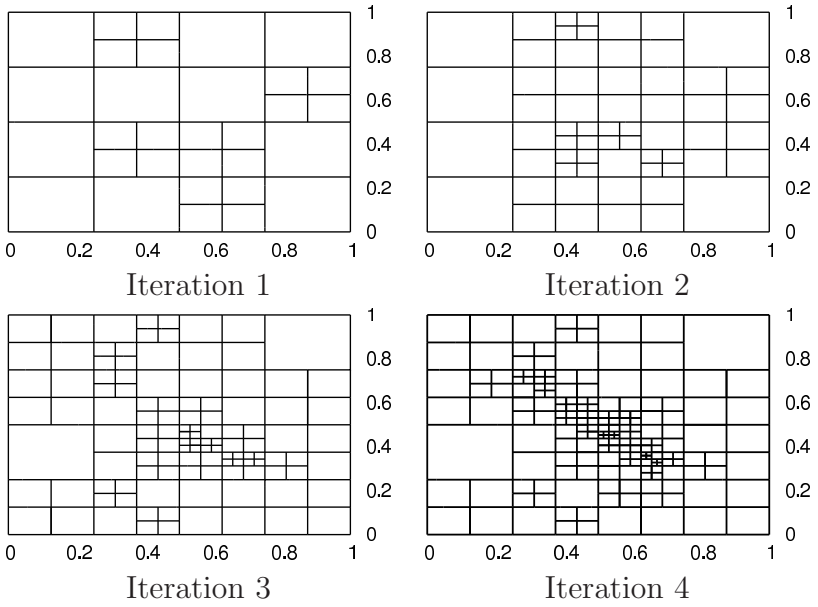


Figure 6.5: Optimization meshes generated by algorithm 6.6.1 on numerical test B.

Conclusions

This chapter considers consistency and mesh adaptivity for discrete parameter estimation problems. The discrete, or *discretize – first* approach is very attractive in practice, because the dual and optimality equations in the KKT set, as well as the reduced gradient of the target functional, can be generated with low effort by automatic differentiation. However, the discrete approach introduces several complications in the inverse solution algorithm. A consistency analysis of the full set of KKT equations is needed before convergence of the discrete optimal solution can be established. Investigating consistency and stability of the discrete KKT system is one of the main objectives of this chapter. While the discretization of the primal equation is *a priori* assumed to be convergent, the consistency of the dual and optimality equations does not automatically follow. We extend the dual consistency framework given in [28] to cover the third equation in the KKT set, namely, the discrete optimality condition. While consistency and stability may not immediately hold for this equation, the analysis shows that they can be restored through suitable consistent modifications of the discrete target functional. An *a priori* error analysis is performed on the modified optimality optimality equation. The theoretical results are supported by the numerical experiments

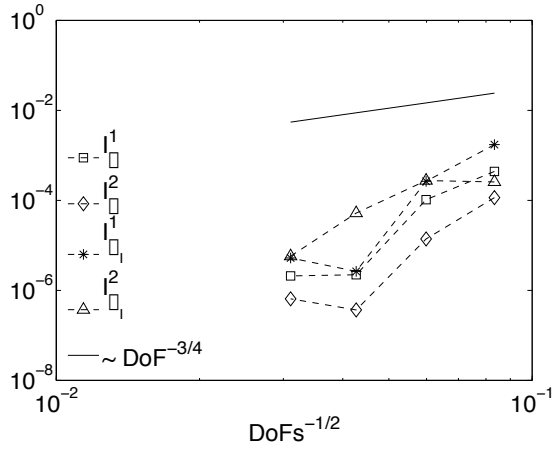


Figure 6.6: Asymptotic behavior of the integrals (6.31a)–(6.31d) for the *a posteriori* numerical experiments with test B.

with a symmetric DG discretization of the primal problem. Discontinuous Galerkin is chosen as the discretization method because of its amenability to parallel computations, and h/p-adaptivity.

A crucial ingredient in any adaptive algorithm is the error estimation step, that guides the mesh refinement process. Previous results in error estimation are either of limited practical use (because of their dependence of problem–dependent unknown constants), or rely on the assumptions that the control parameter space is finite dimensional. We explain the construction of a practical error estimator for parameter estimation problems, in the more general case of infinite dimensional controls. The estimation process is based on dual–weighted residuals of first and second order sensitivity equations. The Hessian of the reduced cost functional is replaced by a low–order BFGS approximation. The use of a BFGS Hessian, obtained at very low cost from the optimization algorithm, keeps the computational overhead of the *a posteriori* error estimation procedure sufficiently low to make it feasible in practice.

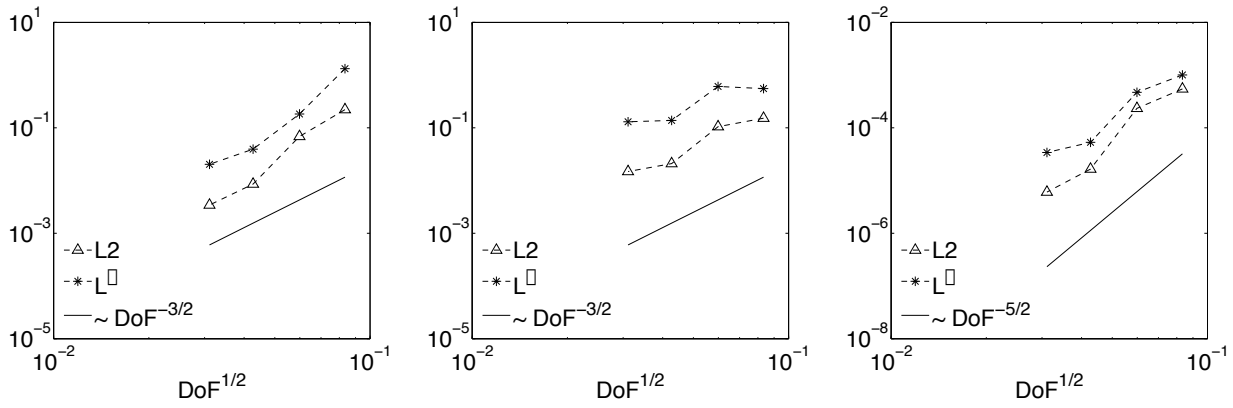


Figure 6.7: Solution convergence for the multigrid optimization algorithm 6.6.1 using the *a posteriori* estimation procedure 6.4.1: optimal solution (left), primal solution (center), and dual solution convergence (right).

Chapter 7

Summary and outlook

The discrete duality framework proposed in this dissertation opens the possibility to perform inverse studies using exclusively the discrete adjoint method, throughout the entire inversion process. This purely discrete approach is very attractive in practice due to the fact that the discrete first and second order optimality systems can be generated with low effort using automatic differentiation. We analyze in detail the properties of discrete adjoints for the main ingredients of state–of–the–art adaptive algorithms: space and time mesh refinements, solution limiters, grid transfer operators, and error analysis and estimation. Our framework enables the construction of consistent and stable discretizations of inverse problems that benefit from all the above mentioned adaptive features. Our prototypical discretization methods are nodal discontinuous Galerkin for space domains, and Runge–Kutta quadratures for the time dimension. This particular choice of discretizations allows for maximum efficiency, and versatility in h/p-adaptive algorithms. The finite volume method is also examined as an alternative to DG.

Time and space refinement are essential components of any large–scale inversion algorithm. Spatial meshes can be locally refined or coarsened to accurately capture the solution features in regions of interest. Similarly, adaptive time stepping allows for maximum efficiency in the inverse solution algorithm, by varying the time steps according to the local temporal error of the constraint equations. However, adaptivity introduces important challenges in the inversion process. While the primal discretization may *a priori* be considered to be convergent, numerical consistency and stability of the discrete dual or optimality equation variables are not guaranteed. We give two extensions of the dual consistency concept explained by other authors. First, we define dual consistency for the well–known space–time adaptive Runge–Kutta discontinuous Galerkin discretizations. The dual consistency definition for the spatial discretization cannot be readily extended to the time dimension. However, a careful analysis of the weak form Runge–Kutta discrete adjoints, reveals that primal RK–DG discretizations that are adjoint consistent in space, are also dual consistent in time. Numerical accuracy order tests, for a two–dimensional data assimilation scenario, confirm the theoretical order

results for the discrete space–time dual problem.

The second extension of the consistency framework addresses the third equation in the discrete KKT set, namely, the optimality condition. We introduce the concept of consistency for the discrete optimality equation, by extending the consistency definitions for the primal and dual discretizations. In this framework, consistency and stability of the dual and optimality discretizations imply the convergence of the discrete optimal solution to its analytical counterpart. The discrete optimality equation does not inherit the consistency automatically from the primal discretization. However, our analysis indicates that consistency and stability issues with the discrete optimality equation may be resolved by consistent suitable modifications of the target functional for the primal problem. The theoretical analysis is supported by numerical results for a prototypical elliptic inverse problem.

Other essential components of adaptive inverse solution are error analysis and estimation for local mesh coarsening and refinement. We consider a general elliptic problem. After suitable consistent modifications to the target functional, the discrete optimality equation is subject to an *a priori* error analysis that establishes consistency and stability. We extend previous results on error estimation for parameter identification problems, to cover the more general case of infinite dimensional controls. The implementation of the error adaptation algorithm relies on the discrete first and second order sensitivity conditions. For computational efficiency, we recommend that second order derivative information (the Hessian of the target functional), be replaced with a quasi–Newton approximation that is provided virtually for free as a by–product of the optimization process. The second order optimality conditions can be generated from the first order KKT set by automatic differentiation.

Future work will extend the *a priori* KKT error analysis to more complex operators. With the continuous improvements in state of the art computing systems, the solution of large scale inverse problems can benefit from the increasing computing power through parallelization of existing numerical algorithms. The degree of transfer in the parallelization properties of the primal discretization, to the discrete dual and optimality equations, warrants further investigation.

The discrete inversion framework described in detail in this dissertation is the first effort at a completely discrete solution algorithm for adaptive inverse problems. The rules and guidelines given herein can be applied to models of high complexity. The use of targeted automatic differentiation will allow practitioners to significantly reduce development time for the numerical solvers, with minimal post–processing required to restore accuracy in the linearized equations. *A priori* error analysis arguments similar the ones given in this work may be used to establish consistency of linearized sensitivity equations for general nonlinear problems. Adaptive mesh refinement can be guided by computationally cheap *a posteriori* error estimates, to achieve both high accuracy in the optimal solution (the end goal in any inversion algorithm), and locally capture fine scale phenomena, without compromising run time efficiency.

Bibliography

[1] Randall J. Leveque. *Finite volume methods for hyperbolic methods*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, UK, 2002.

[2] Alan C. Hindmarsh and Radu Serban. *User documentation for CVODES v 2.5.0*. Lawrence Livermore National Laboratory, Livermore, CA, USA, 2006.

[3] Shengtai Li and Linda Petzold. Description of DASPKADJOINT: An adjoint sensitivity solver for differential-algebraic equations. Technical Report TRCS99-28, University of California at Santa-Barbara, Santa Barbara, CA, USA, 2002.

[4] Peter Eberhard and Christian Bischof. Automatic differentiation of numerical integration algorithms. *Mathematics of Computation*, 68(226):717–731, 1999.

[5] Simon R Arridge and John C Schotland. Optical tomography: forward and inverse problems. *Inverse Problems*, 25(12):123010, 2009.

[6] S L Cotter, M Dashti, J C Robinson, and A M Stuart. Bayesian inverse problems for functions and applications to fluid mechanics. *Inverse Problems*, 25(11):115008, 2009.

[7] W. E. Heinz, F. Christoph, K. Philipp, J. Lu, S. Müller, and P. Schuster. Inverse problems in systems biology. *Inverse Problems*, 25(12):123014, 2009.

[8] Sixun Huang, Jie Xiang, Huadong Du, and Xiaoqun Cao. Inverse problems in atmospheric science and their application. *Journal of Physics: Conference Series*, 12(1):45, 2005.

[9] William Carlisle Thacker. Oceanographic inverse problems. *Phys. D*, 60(1-4):16–37, 1992.

[10] W W Symes. The seismic reflection inverse problem. *Inverse Problems*, 25(12):123008, 2009.

[11] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, Philadelphia, PA, USA, 2nd edition, 2005.

[12] V. Carey, D. Estep, A. Johansson, M. Larson, and S. Tavener. Blockwise adaptivity for time dependent problems based on coarse scale adjoint solutions. *SIAM Journal on Scientific Computing*, 32(4):2121–2145, 2010.

[13] Wolfgang Bangerth. A framework for the adaptive finite element solution of large-scale inverse problems. *SIAM J. Sci. Comput.*, 30(6):2965–2989, 2008.

[14] Wolfgang Bangerth and Amit Joshi. Adaptive finite element methods for the solution of inverse problems in optical tomography. *Inverse Problems*, 24(3):034011 (22pp), 2008.

[15] Shengtai Li and Linda Petzold. Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement. *J. Comput. Phys.*, 198(1):310–325, 2004.

[16] F. Fang, C.C. Pain, M.D. Piggott, G.J. Gorman, and A.J.H. Goddard. An adaptive mesh adjoint data assimilation method applied to free surface flows. *International Journal for Numerical Methods in Fluids*, 47(8-9):995–1001, 2005.

[17] F. Fang, M.D. Piggott, C.C. Pain, G.J. Gorman, and A.J.H. Goddard. An adaptive mesh adjoint data assimilation method. *Ocean Modelling*, 15(1-2):39–55, 2006. The Third International Workshop on Unstructured Mesh Numerical Modelling of Coastal, Shelf and Ocean Flows.

[18] F. Fang, C. C. Pain, M. D. Piggott, G. J. Gorman, and A. J. H. Goddard. Data assimilation of three-dimensional free surface flow using an adaptive adjoint method. part 1: Adjoint model formulation. Technical report, Imperial College London, London, UK, 2007.

[19] F. Fang, C. C. Pain, M.D. Piggott, G. J. Gorman, and A. J. H. Goddard. Data assimilation of three-dimensional free surface flow using an adaptive adjoint method. part 2: Applications. Technical report, Imperial College London, London, UK, 2007.

[20] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, 2nd edition, 2006.

[21] J. L. Lions. *Optimal control of systems governed by partial differential equations*. Springer Verlag, New York, NY, USA, 1971.

[22] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer Verlag, 2007.

[23] Andreas Griewank. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, Philadelphia, PA, USA, 2000.

[24] M. Alexe and A. Sandu. Space-time adaptive solution of inverse problems with the discrete adjoint method. Technical Report TR-10-14, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2010.

[25] Roland Becker and Boris Vexler. A posteriori error estimation for finite element discretization of parameter identification problems. *Numer. Math.*, 96(3):435–459, 2003.

[26] R. M. Lewis and S. G. Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM J. Sci. Comput.*, 26:1811–1837, June 2005.

[27] Max D. Gunzburger. *Perspectives in Flow Control and Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[28] Ralf Hartmann. Adjoint consistency analysis of discontinuous galerkin discretizations. *SIAM J. Numer. Anal.*, 45(6):2671–2696, 2007.

[29] Zheng Liu and Adrian Sandu. On the properties of discrete adjoints of numerical methods for the advection equation. *International Journal for Numerical Methods in Fluids*, 56(7):769–803, 2007.

[30] Radu Serban and Alan C. Hindmarsh. CVODES: the sensitivity-enabled ODE solver in SUNDIALS. Technical Report UCRL-JP-200037, Lawrence Livermore National Laboratory, Livermore, CA, USA, 2003.

[31] Shengtai Li and Linda Petzold. Design of new DASPK for sensitivity analysis. Technical Report TRCS99-28, University of California at Santa-Barbara, Santa Barbara, CA, USA, 1999.

[32] J. Lu. *An a Posteriori Error Control Framework for Adaptive Precision Optimization Using the Discontinuous Galerkin Finite Element Method*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, 2005.

[33] William W. Hager. Runge-Kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282, 2000.

[34] Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

[35] R. Rannacher and B. Vexler. Adaptive finite element discretization in PDE-based optimization. *GAMM-Mitteilungen*, 33(2):177–193, 2010.

[36] R. Bermejo and J. Carpio. A space-time adaptive finite element algorithm based on dual weighted residual methodology for parabolic equations. *SIAM J. Sci. Comput.*, 31:3324–3355, August 2009.

[37] R. Hartmann and P. Houston. Adaptive discontinuous galerkin finite element methods for the compressible euler equations. *J. Comput. Phys.*, 183:508–532, December 2002.

[38] R Hartmann and P. Houston. Adaptive discontinuous galerkin finite element methods for nonlinear hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 24:979–1004, March 2002.

[39] E. H. Georgoulis, E. Hall, and P. Houston. Discontinuous galerkin methods on hp-anisotropic meshes ii: a posteriori error analysis and adaptivity. *Appl. Numer. Math.*, 59:2179–2194, September 2009.

[40] T. Grätsch and K.-J. Bathe. Review: A posteriori error estimation techniques in practical finite element analysis. *Comput. Struct.*, 83:235–265, January 2005.

[41] S. Micheletti and S. Perotto. Output functional control for nonlinear equations driven by anisotropic mesh adaption: The navier-stokes equations. *SIAM J. Sci. Comput.*, 30:2817–2854, October 2008.

[42] T. Feng, M. Gulliksson, and W. Liu. Adaptive finite element methods for the identification of elastic constants. *J. Sci. Comput.*, 26:217–235, February 2006.

[43] T. Feng, N. Yan, and W. Liu. Adaptive finite element methods for the identification of distributed parameters in elliptic equation. *Advances in Computational Mathematics*, 29:27–53, 2008. 10.1007/s10444-007-9035-6.

[44] Randall J. Leveque. *Finite volume methods for hyperbolic methods*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, UK, 2002.

[45] A. Saltelli, E. M. Scott, and K. Chan. *Sensitivity analysis*, volume 535 of *Wiley Series in Probability and Analysis*. Wiley, Chichester, UK, 2000.

[46] Yang Cao, Shengtai Li, Linda Petzold, and Radu Serban. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint dae system and its numerical solution. *SIAM J. Sci. Comput.*, 24(3):1076–1089, 2002.

[47] Yang Cao, Shengtai Li, and Linda Petzold. Adjoint sensitivity analysis for differential-algebraic equations: Algorithms and software. *Journal of Computational and Applied Mathematics*, 149(1):171–191, 2002.

[48] D. G. Cacuci. Sensitivity theory for nonlinear systems. I: Nonlinear functional analysis approach. *J. Math. Phys.*, 22:2794–2802, 1981.

[49] D. G. Cacuci. Sensitivity theory for nonlinear systems. II: Extensions to additional classes of responses. *J. Math. Phys.*, 22:2803–2812, 1981.

[50] M. B. Giles and N. Pierce. Adjoint equations in cfd: duality, boundary conditions and solution behavior. In *Proceedings of the 13th Computational Fluid Dynamics Conference*, number AIAA Paper 97-1850, 1997.

[51] M. B. Giles and N. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65:393–415, 2000. 10.1023/A:1011430410075.

[52] K. Atkinson and W. Han. *Theoretical numerical analysis: A functional analysis framework*. Number 39 in Texts in applied mathematics. Springer, New York, NY, USA, 2nd edition, 2005.

[53] Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch. OpenAD/F: A modular, open-source tool for automatic differentiation of Fortran codes. *ACM Transactions on Mathematical Software*, 34(4):18:1–18:36, 2008.

[54] Laurent Hascöet and Valérie Pascual. TAPENADE 2.1 User's guide. Technical Report 0300, INRIA, Sophia Antipolis, France, 2004.

[55] Andreas Griewank, David Juedes, and Jean Utke. Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131–167, 1996.

[56] C. H. Bischof, L. Roh, and A. J. Mauer-Oats. ADIC: an extensible automatic differentiation tool for ANSI-C. *Software: Practice & Experience*, 27(12):1427–1456, 1997.

[57] Ralf Giering and Thomas Kaminski. Applying TAF to generate efficient derivative code of Fortran 77-95 programs. *Proceedings in Applied Mathematics and Mechanics*, 2(1):54–57, 2003.

[58] Ralf Giering. *Tangent linear and Adjoint Model Compiler, Users manual 1.4*, 1999.

[59] Michael A. Heroux and James M. Willenbring. Trilinos Users Guide. Technical Report SAND2003-2952, Sandia National Laboratories, 2003.

[60] Shaun A. Forth. An efficient overloaded implementation of forward mode automatic differentiation in MATLAB. *ACM Trans. Math. Softw.*, 32(2):195–222, 2006.

[61] Axel Dürrbaum, Willy Klier, and Hubert Hahn. Comparison of automatic and symbolic differentiation in mathematical modeling and computer simulation of rigid-body systems. *Multibody System Dynamics*, 7:331–355, 2002. 10.1023/A:1015523018029.

[62] V. Heuveline and A. Walther. Online checkpointing for parallel adjoint computation in PDEs: Application to goal-oriented adaptivity and flow control. In W. Nagel, W. Walter, and W. Lehner, editors, *Euro-Par 2006 Parallel Processing*, volume 4128 of *Lecture Notes in Computer Science*, pages 689–699. Springer Verlag, Germany, 2006.

[63] P. Stumm and A. Walther. Multistage approaches for optimal offline checkpointing. *SIAM Journal on Scientific Computing*, 31(3):1946–1967, 2009.

[64] P. Stumm and A. Walther. New algorithms for optimal online checkpointing. *SIAM J. Sci. Comput.*, 32:836–854, March 2010.

[65] N. K. Yamaleev, B. Diskin, and E. J. Nielsen. Local-in-time adjoint-based method for design optimization of unsteady flows. *Journal of Computational Physics*, In Press, Corrected Proof:–, 2010.

[66] L. Wang, D. J. Mavriplis, and W. K. Anderson. Unsteady discrete adjoint formulation for high-order discontinuous galerkin discretizations in time dependent flows. In *AIAA Aerospace Sciences Meeting*, number AIAA Paper 2010-0367, 2009.

[67] C. Bischof, G. Corliss, and A. Griewank. Structured second-and higher-order derivatives through univariate taylor series. *Optimization methods and software*, 2(3–4):211–232, 1993.

[68] A. Griewank, J. Utke, and A. Walther. Evaluating higher derivative tensors by forward propagation of univariate taylor series. *Mathematics of computation*, (69):1117–1130, 2000.

[69] D. B. Özyurt and P. I. Barton. Cheap second order directional derivatives of stiff ODE embedded functionals. *SIAM Journal on Scientific Computing*, 26(5):1725–1743, 2005.

[70] F. X. LeDimet, I. M. Navon, and D.N. Daescu. Second order information in data assimilation. *Monthly Weather Review*, 130(3):629–648, 2002.

[71] Zhi Wang, I. M. Navon, F. X. LeDimet, and X. Zou. The second order adjoint analysis: Theory and applications. *Meteorology and Atmospheric Physics*, 50(1-3):3–20, 1992.

[72] Adrian Sandu and Lin Zhang. Discrete second order adjoints in atmospheric chemical transport modeling. *Journal of Computational Physics*, 227(12):5949–5983, 2008.

[73] M. Alexe, A. Cioaca, and A. Sandu. Obtaining and using second order derivative information in the solution of large scale inverse problems. In *Proceedings of the 2010 Spring Simulation Multiconference*, SpringSim '10, pages 85:1–85:8, New York, NY, USA, 2010. ACM.

[74] M. Shichitake and M. Kawahara. Optimal control applied to water flow using second order adjoint method. *Int. J. Comput. Fluid Dyn.*, 22:351–365, June 2008.

[75] Z. Wang, K. Droegemeier, and L. White. The adjoint newton algorithm for large-scale unconstrained optimization in meteorology applications. *Comput. Optim. Appl.*, 10:283–320, July 1998.

[76] X. Ye, P. Li, and F. Y. Liu. Exact time-domain second-order adjoint-sensitivity computation for linear circuit analysis and optimization. *Trans. Cir. Sys. Part I*, 57:236–248, January 2010.

[77] L. V. Kantorovich and G. P. Akilov. *Functional Analysis in Normed Spaces*, volume 46 of *International Series of Monographs in Pure and Applied Mathematics*. Pergamon Press, 1964.

[78] W. K Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. Technical report, 1997.

[79] Eyal Arian and Manuel D. Salas. Admitting the inadmissible: Adjoint formulation for incomplete cost functionals in aerodynamic optimization. Technical report, 1997.

[80] Eyal Arian and Manuel D. Salas. *Adjoint formulation using auxiliary boundary equations (ABEs) demonstrated on the Stokes equations*, volume 515 of *Lecture Notes in Physics*, pages 355–360. Springer Berlin / Heidelberg, 1998.

[81] Alain Sei and William Symes. A note on consistency and adjointness for numerical schemes. Technical Report 95527, University of Minnesota, 1995.

[82] Z. Sirkes and E. Tziperman. Finite difference of adjoint or adjoint of finite difference? *Monthly Weather Review*, 125(12):3373–3378, 1997.

[83] G. F. Duivesteijn, H. Bijl, B. Koren, and E. H. van Brummelen. On the adjoint solution of the quasi-1D Euler equations: the effect of boundary conditions and the numerical flux function. *International Journal for Numerical Methods in Fluids*, 47(8-9):987–993, 2005.

[84] M. B. Giles. Discrete adjoint approximations with shocks. In T. Hou and E. Tadmor, editors, *Hyperbolic Problems: Theory, Numerics, Applications*. Springer-Verlag, 2003.

[85] Mike Giles and Stefan Ulbrich. Convergence of linearized and adjoint approximations for discontinuous solutions of conservation laws. part 1: Linearized approximations and linearized output functionals. *SIAM Journal on Numerical Analysis*, 48(3):882–904, 2010.

[86] Mike Giles and Stefan Ulbrich. Convergence of linearized and adjoint approximations for discontinuous solutions of conservation laws. part 2: Adjoint approximations and extensions. *SIAM Journal on Numerical Analysis*, 48(3):905–921, 2010.

[87] A. C. Marta, C. A. Mader, J. R. R. A. Martins, E. Van der Weide, and J. J. Alonso. A methodology for the development of discrete adjoint solvers using automatic differentiation tools. *Int. J. Comput. Fluid Dyn.*, 21(9-10):307–327, 2007.

[88] E. Simon. *Variational data assimilation for nested models*. PhD thesis, Universite Joseph Fourier - Grenoble I, 2007.

[89] E. Simon, L. Debreu, and E. Blayo. 4d variational data assimilation for locally nested models : complementary theoretical aspects and application to a 2d shallow water model. *Submitted*, 2008.

[90] L. Debreu, E. Simon, and E. Blayo. 4d variational data assimilation for locally nested models: optimality system and preliminary numerical experiments. *Submitted*, 2008.

[91] F. Fang, C.C. Pain, I.M. Navon, G.J. Gorman, M.D. Piggott, P.A. Allison, P.E. Farrell, and A.J.H. Goddard. A POD reduced order unstructured mesh ocean modelling method for moderate reynolds number flows. *Ocean Modelling*, 28(1-3):127–136, 2009.

[92] F. Fang, C.C. Pain, M.D. Piggott, G.J. Gorman, P.A. Allison, and A.J.H. Goddard. A POD goal-oriented error measure for mesh optimisation. *International Journal for Numerical Methods in Fluids*, (Early View), 2009.

[93] F. Fang, C.C. Pain, I. M. Navon, G.J. Gorman, M.D. Piggott, and P.A. Allison. The independent set perturbation adjoint method: A new method of differentiating mesh based fluids models. *(Submitted for publication to International Journal for Numerical Methods in Fluids)*, 2009.

[94] Adrian Sandu and Philipp Miehe. Forward, tangent linear, and adjoint Runge-Kutta methods in KPP–2.2 for efficient chemical kinetic simulations. Technical Report TR-06-17, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2006.

[95] Andrea Walther. Automatic differentiation of explicit Runge-Kutta methods for optimal control. *Computational Optimization and Applications*, 36(1):83–108, 2007.

[96] Adrian Sandu. On consistency properties of discrete adjoint linear multistep methods. Technical Report TR-07-40, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2007.

[97] Mihai Alexe and Adrian Sandu. Forward and adjoint sensitivity analysis with continuous explicit runge–kutta schemes. *Applied Mathematics and Computation*, 208(2):328–346, 2009.

[98] Kenneth Eriksson, Don Estep, Peter Hansbo, and Claes Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, 4(-1):105–158, 1995.

[99] Eric J. Nielsen, James Lu, Michael A. Park, and David L. Darmofal. An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids. *Computers & Fluids*, 33(9):1131–1155, 2004.

[100] P. W. Power, C. C. Pain, M. D. Piggott, F. Fang, G. J. Gorman, A. P. Umpleby, A. J. H. Goddard, and I. M. Navon. Adjoint a posteriori error measures for anisotropic mesh optimisation. *Comput. Math. Appl.*, 52(8-9):1213–1242, 2006.

[101] David A. Venditti and David L. Darmofal. Adjoint error estimation and grid adaptation for functional outputs: application to quasi-one-dimensional flow. *J. Comput. Phys.*, 164(1):204–227, 2000.

[102] David A. Venditti and David L. Darmofal. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. *J. Comput. Phys.*, 176(1):40–69, 2002.

[103] David A. Venditti and David L. Darmofal. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *J. Comput. Phys.*, 187(1):22–46, 2003.

[104] Q. Wang, P. Moin, and G. Iaccarino. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM Journal on Scientific Computing*, 31(4):2549–2567, 2009.

[105] J. D. Muller and M. B. Giles. Solution adaptive mesh refinement using adjoint error analysis. In *15th AIAA Computational Fluid Dynamics Conference*, number AIAA-2001-2550. AIAA, 2001.

[106] Yang Cao and Linda Petzold. A posteriori error estimation and global error control for ordinary differential equations by the adjoint method. *SIAM J. Sci. Comput.*, 26(2):359–374, 2005.

[107] M. B. Giles and E. Suli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. *Acta Numerica*, 11:145–236, 2002.

[108] Niles A. Pierce and Michael B. Giles. Adjoint recovery of superconvergent functionals from pde approximations. *SIAM Rev.*, 42(2):247–264, 2000.

[109] Niles A. Pierce and Michael B. Giles. Adjoint and defect error bounding and correction for functional estimates. *J. Comput. Phys.*, 200(2):769–794, 2004.

[110] R. Becker and B. Vexler. A posteriori error estimation for finite element discretization of parameter identification problems. *Numer. Math.*, 96:435–459, January 2004.

[111] R. Becker and B. Vexler. Mesh refinement and numerical sensitivity analysis for parameter calibration of partial differential equations. *J. Comput. Phys.*, 206:95–110, June 2005.

[112] P. Wesseling. *Principles of computational fluid dynamics*, volume 29 of *Springer Series in Computational Mathematics*. Springer–Verlag, Berlin, Germany, 2009.

[113] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time dependent problems*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, UK, 2007.

[114] Robert E. Harris and Z.J. Wang. High-order adaptive quadrature-free spectral volume method on unstructured grids. *Computers and Fluids*, 38(10):2006–2025, 2009.

[115] Z. J. Wang. Spectral (finite) volume method for conservation laws on unstructured grids. basic formulation: Basic formulation. *Journal of Computational Physics*, 178(1):210–251, 2002.

[116] Z. J. Wang and Yen Liu. Spectral (finite) volume method for conservation laws on unstructured grids: Ii. extension to two-dimensional scalar equation. *Journal of Computational Physics*, 179(2):665–697, 2002.

[117] W. Hundsdorfer, B. Koren, M. van Loon, and J. G. Verwer. A positive finite-difference advection scheme. *J. Comput. Phys.*, 117(1):35–46, 1995.

[118] Z. Liu and A. Sandu. On the properties of discrete adjoints of numerical methods for the advection equation. *International Journal for Numerical Methods in Fluids*, 56(7):769–803, 2008.

[119] Zheng Liu and Adrian Sandu. *Analysis of discrete adjoints for upwind numerical schemes*, volume 3515 of *Lecture Notes in Computer Science*, pages 829–836. Springer Berlin / Heidelberg, 2005.

[120] Bram van Leer. Towards the ultimate conservative difference scheme: IV. A new approach to numerical convection. *Journal of Computational Physics*, 23:276–299, 1977.

[121] Mihai Alexe and Adrian Sandu. On the discrete adjoints of adaptive time stepping algorithms. *J. Comput. Appl. Math.*, 233(4):1005–1020, 2009.

[122] Adrian Sandu. On the properties of runge-kutta discrete adjoints. In *International Conference on Computational Science (4)*, pages 550–557, 2006.

[123] T. S. Baker, J. R. Dormand, J. P. Gilmore, and P. J. Prince. Continuous approximation with embedded Runge-Kutta methods. *Applied Numerical Mathematics*, 22(1-3):51–62, 1996.

[124] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations: Nonstiff Problems*, volume I of *Computational Mathematics*. Springer-Verlag, 1993.

[125] P. W. Sharp and J. H. Verner. Generation of high-order interpolants for explicit Runge-Kutta pairs. *ACM Transactions on Mathematical Software*, 24(1):13–29, 1998.

[126] A. C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005.

[127] V. Damian, Adrian Sandu, M. Damian, F. Potra, and G. R. Carmichael. The kinetic preprocessor KPP - a software environment for solving chemical kinetics. *Computers & Chemical Engineering*, 26:1567–1579, 2002.

[128] Dacian N. Daescu, Adrian Sandu, and Gregory R. Carmichael. Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: II – numerical validation and applications. *Atmospheric Environment*, 37(36):5097–5114, 2003.

[129] Adrian Sandu, Dacian Daescu, and Gregory R. Carmichael. Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: Part I – theory and software tools. *Atmospheric Environment*, 37(36):5083–5096, 2003.

[130] J. H. Verner. Differentiable interpolants for high-order Runge-Kutta methods. *SIAM J. Numer. Anal.*, 30(5):1446–1466, 1993.

[131] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations: Stiff and Differential-Algebraic Problems*, volume II of *Computational Mathematics*. Springer-Verlag, 1994.

[132] Alexander Ostermann. Continuous extensions of Rosenbrock-type methods. *Computing*, 44(1):59–68, 1990.

[133] Ernst Hairer and Alexander Ostermann. Dense output for extrapolation methods. *Numerische mathematik*, 58(1):419–439, 1990.

[134] J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[135] P. J. Prince and J. R. Dormand. High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 7:67–76, 1981.

[136] J. R. Dormand, M. A. Lockyer, N. E. McGorrigan, and P. J. Prince. Global error estimation with Runge-Kutta triples. *Computers & Mathematics with Applications*, 18(9):835–846, 1989.

[137] Alistair Adcroft, Jean-Michel Campin, Patrick Heimbach, Chris Hill, and John Marshall. *MIT General Circulation Model User's Manual*. Massachusetts Institute of Technology, Boston, MA, USA, 2007.

[138] Alan C. Hindmarsh and Radu Serban. *Example Programs for CVODES v.2.5.0*. Lawrence Livermore National Laboratory, Livermore, CA, USA, Nov. 2006.

[139] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[140] Thomas Beck. Automatic differentiation of iterative processes. In *ICCAM'92: Proceedings of the fifth international conference on Computational and applied mathematics*, pages 109–118, Amsterdam, The Netherlands, 1994. Elsevier Science Publishers B. V.

[141] Andreas Griewank, Christian H. Bischof, George F. Corliss, Alan Carle, and Karen Williamson. Derivative convergence for iterative equation solvers. *Optimization Methods and Software*, 2:321–355, 1993.

[142] A. Sandu, D. Daescu, G. R. Carmichael, and T. Chai. Adjoint sensitivity analysis of regional air quality models. *Journal of Computational Physics*, 204(1):222–252, 2005.

[143] Ralf Giering and Thomas Kaminski. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software*, 24(4):437–474, 1998.

[144] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, 18:1–22, January 1997.

[145] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 28(125):145–162, 1974.

[146] T. A. Oliver and D. L. Darmofal. Analysis of dual consistency for discontinuous Galerkin discretizations of source terms. *SIAM Journal on Numerical Analysis*, 47(5):3507–3525, 2009.

[147] Wolfgang Bangerth. *Adaptive Finite Element Methods for the Identification of Distributed Parameters in Partial Differential Equations*. PhD thesis, University of Heidelberg, 2002.

[148] Wolfgang Bangerth, Amit Joshi, and Eva M. Sevick-Muraca. Inverse biomedical imaging using separately adapted meshes for parameters and forward model variables. In *Proceedings of the IEEE International Symposium on Biomedical Imaging, Arlington, VA, 2007*, pages 1368–1371. IEEE, 2007.

[149] L. Wang and D. J. Mavriplis. Adjoint based *h-p* adaptive discontinuous Galerkin methods for aerospace applications. In *AIAA Aerospace Sciences Meeting*, number AIAA Paper 2009-0952, 2009.

[150] R Serban, S. Li, and L. R. Petzold. Adaptive algorithms for optimal control of time-dependent partial differential-algebraic equation systems. *Int. J. Numer. Meth. Eng.*, (57):1457–1469, 2003.

[151] Jayandran Palaniappan, Robert B. Haber, and Robert L. Jerrard. A spacetime discontinuous galerkin method for scalar conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 193(33-35):3607–3631, 2004.

[152] J. J. Sudirham, J. J. W. van der Vegt, and R. M. J. van Damme. Space-time discontinuous galerkin method for advection-diffusion problems on time-dependent domains. *Appl. Numer. Math.*, 56(12):1491–1518, 2006.

[153] M. B. Giles, M. G. Larson, J. M. Levenstam, and E. Suli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. Technical Report 97/06, Oxford University Computing Lab, Numerical Analysis Group, Oxford, UK, 1997.

[154] K. Harriman, D. Gavaghan, and E. Süli. The importance of adjoint consistency in the approximation of linear functionals using the discontinuous galerkin finite element method. Technical Report 04/18, Oxford University Computing Lab, Numerical Analysis Group, Oxford, UK, 2004.

[155] Sigal Gottlieb and Lee-Ad J. Gottlieb. Strong stability preserving properties of Runge–Kutta time discretization methods for linear constant coefficient operators. *J. Sci. Comput.*, 18(1):83–109, 2003.

[156] E. Haber, S Heldmann, and U. Ascher. Adaptive finite volume method for distributed non-smooth parameter identification. *Inverse Problems*, 23(4):1659, 2007.

[157] Y. Liu, C. W. Shu, E. Tadmor, and M. Zhang. Non-oscillatory hierarchical reconstruction for central and finite volume schemes. *Communications in Computational Physics*, 2(5):933–963, 2007.

[158] R. Hartmann. Error estimation and adjoint based refinement for an adjoint consistent dg discretisation of the compressible euler equations. *Int. J. Comput. Sci. Math.*, 1(2-4):207–220, 2007.

[159] Ralf Hartmann and Paul Houston. An optimal order interior penalty discontinuous galerkin discretization of the compressible navier-stokes equations. *Journal of Computational Physics*, 227(22):9670–9685, 2008.

[160] Jari Kaipio and Erkki Somersalo. Statistical inverse problems: Discretization, model reduction and inverse crimes. *Journal of Computational and Applied Mathematics*, 198(2):493–504, 2007. Applied Computational Inverse Problems.

[161] libLBFGS: a library of limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), 2009.

[162] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[163] W. Bangerth, R. Hartmann, and G. Kanschat. `deal.II` *Differential Equations Analysis Library, Technical Reference*. `http://www.dealii.org`.

[164] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge–Kutta schemes. *Math. Comput.*, 67(221):73–85, 1998.

[165] P. Grisvard. *Elliptic problems in nonsmooth domains*, volume 24 of *Monographs and studies in mathematics*. Pitman Advanced Pub. Program, Boston, MA, USA, 1985.

[166] M. Renardy and R. C. Rogers. *An Introduction to Partial Differential Equations*, volume 13 of *Texts in Applied Mathematics*. Springer Verlag, 2nd edition, 2004.

[167] P. C. Hansen. *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics, Philadephia, PA, 2010.

[168] K. Kunisch and J. Zou. Iterative choices of regularization parameters in linear inverse problems. *Inverse Problems*, 14(5):1247, 1998.

[169] J. Xie and J. Zou. An improved model function method for choosing regularization parameters in linear inverse problems. *Inverse Problems*, 18(3):631, 2002.

[170] P. G. Ciarlet. *The finite element method for elliptic problems*. North Holland Publishing Company, New York, NY, USA, 2002.

[171] B. Riviere. *Discontinuous Galerkin Methods For Solving Elliptic And parabolic Equations: Theory and Implementation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[172] R. Becker. Estimating the control error in discretized PDE-constrained optimization. *J. Numer. Math.*, 14:163–185, September 2006.