An Adjoint Method using Fully Implicit Runge-Kutta Schemes for Optimization of Flow Problems

Michael Franco*, Per-Olof Persson†, Will Pazner‡, and Matthew J. Zahr§

University of California, Berkeley, Berkeley, CA 94720-3840, U.S.A

Lawrence Livermore National Laboratory, Livermore, CA 94550-9234, U.S.A.

University of Notre Dame, Notre Dame, IN 46556-5637, U.S.A

The fully discrete adjoint equations and corresponding adjoint method are derived for unsteady PDE-constrained optimization problems. Specifically, we consider conservation laws on deforming domains that are temporally discretized by high-order fully implicit Runge-Kutta (IRK) schemes. Through a change of variables, the linear systems arising in the primal and dual problem are transformed, leading to computationally cheaper systems that compare competitively with those derived from diagonally implicit Runge-Kutta (DIRK) schemes. Quantities of interest that take the form of space-time integrals are discretized in a solver-consistent manner. Our fully discrete, IRK adjoint method is used to compute exact gradients of quantities of interest with respect to the optimization parameters. These quantities of interest and their gradients are used for gradient-based PDE-constrained optimization. Our implementation of this IRK adjoint method is tested by computing the energetically optimal trajectory of a 2D airfoil in flow governed by the compressible Navier-Stokes equations. We also analyze the parallel performance of our IRK adjoint method and the DIRK adjoint method, showing that our implementation is computationally comparable.

I. Introduction

Optimization problems constrained by partial differential equations (PDEs) commonly arise in computational fluid dynamics (CFD), particularly in the context of structural design or control systems. When the problems have inherently transient dynamics or objective functions that do not reach a steady state, unsteady analysis is required. In particular, these time-dependent PDE-constrained optimization problems have the form:

*Graduate Student, Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720-3840. Email: michaelfranco@berkeley.edu. AIAA Student Member.

†Associate Professor, Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720-3840. Email: persson@berkeley.edu. AIAA Senior Member.

‡Sidney Fernbach Postdoctoral Fellow, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94550-9234. Email: pazner1@llnl.gov. AIAA Member.

§Assistant Professor, Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556-5637. Email: mzahr@nd.edu. AIAA Member.

$$
\begin{aligned}
\min_{\boldsymbol{U},\boldsymbol{\mu}} \quad & \int_0^T \int_{\partial\Omega} f_0(\boldsymbol{U},\boldsymbol{\mu},t)\ dS\ dt \\
s.t. \quad & \int_0^T \int_{\partial\Omega} f_1(\boldsymbol{U},\boldsymbol{\mu},t)\ dS\ dt \leq 0 \\
& \frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}, \nabla \boldsymbol{U}) = 0 \text{ in } \Omega(\boldsymbol{\mu}, t),
\end{aligned}
\tag{1}
$$

where the last constraint corresponds to a system of conservation laws in a moving domain $\Omega(\boldsymbol{\mu}, t)$ with solution $\boldsymbol{U}$ and optimization parameters $\boldsymbol{\mu}$, and the objective and constraint functions take the form of space-time integrals of quantities of interest f_0 and f_1 over the surface of the body $\partial\Omega$. In the context of CFD, this last equation is typically the Navier-Stokes equations.

If derivatives of these functions are either computationally infeasible or not available, derivative-free methods[1] may be used for such design-based optimization problems. While these widely applicable methods handle non-smooth objectives, they often suffer from slow convergence rates, if they converge at all.[2] In this work, we only consider problems with smooth objectives and constraints, allowing use of gradient-based techniques that display much higher convergence rates. A practical implementation of gradient-based methods requires highly accurate computation of quantities of interest (QoI) because these values drive the optimization trajectory.[3]

To minimize computational errors, we develop a globally high-order numerical discretization of our conservation laws. For a given level of accuracy, high-order methods require fewer degrees of freedom.[4] Moreover, we choose to efficiently compute derivatives of optimization functionals with an adjoint method. The principal advantage of the adjoint approach is that the computational cost is independent of the number of design variables. Therefore, a sensitivity analysis for hundreds of optimization variables can be performed at a cost roughly equivalent to solving the governing equations twice.[5] While an adjoint method in the context of PDEs can be derived at the continuous, semi-discrete, or fully discrete level, we choose to consider the fully discrete adjoint method as it ensures discrete consistency of computed gradients with discretization errors.[6] In the context of gradient-based optimization, discrete consistency ensures black-box optimizers maximally benefit from our high-order methods.[2] Otherwise, specialized optimization algorithms must be employed to handle gradient inexactness.[7] A key property of adjoint-consistent discretizations is that they possess optimal convergence rates in the L^2-norm and in QoI.[8]

Previous work on unsteady adjoints has mostly considered temporal discretization by Backward Differentiation Formulas.[5,9] However, apart from requiring special techniques for initialization, these schemes are limited to second-order accuracy if A-stability is required.[10] Previous work by Zahr et al.[3] derived a fully discrete adjoint method using the diagonally implicit Runge-Kutta (DIRK) schemes common in many engineering applications. There exist high-order, A-stable (and even L-stable) DIRK schemes; however, these methods have a low stage order, often resulting in order reduction when applied to stiff problems.[11] Indeed, in Section IV.D, we find stability issues with a fifth-order DIRK scheme. Therefore, we turn to fully implicit Runge-Kutta (IRK) schemes, which can be high order, L-stable, and have high stage order. Because each time step requires solving a large, coupled system of equations, these IRK schemes have often been viewed as computationally prohibitive.[12] But recent work by Pazner and Persson[13] showed that a large class of IRK schemes can be transformed into a computationally cheaper system of equations. This new formulation of the Radau IIA IRK schemes compares favorably with equal order DIRKs in terms of accuracy, GMRES iterations, and compute time.

In this work, we derive a fully discrete adjoint method corresponding to this new formulation of IRK schemes. High-order discretization of integrated QoI will be done in a solver-consistent manner in Section II. That is, spatial integrals will be evaluated by integrating the Galerkin shape functions used for the spatial discretization, and temporal integrals will be evaluated via the transformed Radau IIA IRK scheme for the temporal discretization. This ensures the discretization order of QoI exactly matches the PDE discretization. Then the fully discrete adjoint method will be derived in Section III. Finally, we will present results analyzing the accuracy, convergence rate, and scalability of our implementation of this IRK adjoint method in Section IV. Our method will be applied to the CFD problem of determining the energetically optimal trajectory of a 2D airfoil immersed in viscous flow, showing that it is competitive with the DIRK adjoint method from Ref. 3.

II. Governing Equations and Discretization

Consider a general system of conservation laws, defined on a parameterized, deforming domain, $\Omega(\boldsymbol{\mu}, t)$, written at the continuous level as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}}(\boldsymbol{U}, \nabla \boldsymbol{U}) = 0, \tag{2}$$

where the physical flux is decomposed into an inviscid and a viscous part $\boldsymbol{\mathcal{F}}(\boldsymbol{U}, \nabla \boldsymbol{U}) = \boldsymbol{\mathcal{F}}_i(\boldsymbol{U}) + \boldsymbol{\mathcal{F}}_v(\boldsymbol{U}, \nabla \boldsymbol{U})$, $\boldsymbol{U}(\boldsymbol{x}, \boldsymbol{\mu}, t)$ is the solution of the system of conservation laws, $t \in [0, T]$ represents time, and $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$ is a vector of parameters. Let N_{sd} be the number of spatial dimensions and N_c be the number of components of the vector of state variables.

The conservation law on a deforming domain is transformed into a conservation law on a fixed reference domain through the introduction of a time-dependent mapping between the physical and reference domains, resulting in an Arbitrary Lagrangian-Eulerian (ALE) description of the governing equations. For an in-depth derivation of the ALE formulation that satisfies the Geometric Conservation Law (GCL), we refer to Refs. 14 or 3. The result is a system of PDEs: a mapped conservation law and a GCL augmentation both of the form seen in (2).

A. Spatial Discretization

These PDEs are discretized using a method of lines approach. Namely, we spatially discretize (2) by the compact discontinuous Galerkin (DG) method,[15] a sparser variation of the local DG method.[16] After spatial discretization, we have the following system of ordinary differential equations (ODE):

$$\mathbb{M} \frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{\mu}, t), \tag{3}$$

where $\boldsymbol{u}(\boldsymbol{\mu}, t) \in \mathbb{R}^{N_u}$ is our semi-discrete approximation to $\boldsymbol{U}$, and N_u is the number of degrees of freedom from our spatial discretization. A convenient property of this DG-ALE scheme is that all computations are performed on the reference domain, which is *independent* of time and parameter. Therefore, the mass matrix of the ODEs in (3) is time-, solution-, and parameter-independent.

$$\frac{\partial \mathbb{M}}{\partial t} = \frac{\partial \mathbb{M}}{\partial \boldsymbol{u}} = \frac{\partial \mathbb{M}}{\partial \boldsymbol{\mu}} = \boldsymbol{0}$$

In subsequent sections of this article, we will assume that the mass matrix satisfies this property, significantly simplifying Jacobian calculations.

Equation 3 is the system of ODEs from which we will derive our adjoint method. So while this section introduces a DG-ALE spatial discretization to arrive at this system of ODEs, we may instead arrive at this system from a different spatial discretization. A high-order spatial discretization merely provides a stable framework within which we can now focus on high-order temporal discretization so as to yield a globally high-order numerical method.

B. Temporal Discretization

Continuing the method of lines approach, the time interval $[0, T]$ is discretized into a sequence of $N_t + 1$ time values $\left\{t^{(n)}\right\}_{n=0}^{N_t}$, with the n^{th} time step $\Delta t^{(n)} := t^{(n+1)} - t^{(n)}$. A general s-stage Runge-Kutta method will advance the solution from a known $\boldsymbol{u}^{(n)} \approx \boldsymbol{u}(t^{(n)})$ to $\boldsymbol{u}^{(n+1)} \approx \boldsymbol{u}(t^{(n+1)})$ with the update:

$$\begin{aligned} \mathbb{M}\boldsymbol{k}_i^{(n)} &= \boldsymbol{f}\left(\boldsymbol{u}^{(n)} + \Delta t^{(n)} \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)} c_i\right) \qquad i = 1, \ldots, s \\ \boldsymbol{u}^{(n+1)} &= \boldsymbol{u}^{(n)} + \Delta t^{(n)} \sum_{i=1}^{s} b_i \boldsymbol{k}_i^{(n)}. \end{aligned} \tag{4}$$

The coefficients a_{ij}, b_j, and c_i concisely express any Runge-Kutta scheme; therefore we compare the IRK schemes of interest using the Butcher tableau in Table 1b with the DIRK schemes in Table 1a. The IRK schemes that we will explore further enjoy high accuracy and favorable stability properties, but have a dense Butcher matrix $\boldsymbol{A}$. Thus, computing the stages $\boldsymbol{k}_i^{(n)}$ requires the solution of one large nonlinear system of equations of size $s \times N_u$.

$$\begin{array}{c|cccc} c_1 & a_{11} & & & \\ c_2 & a_{21} & a_{22} & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

(a) s-stage DIRK schemes

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

(b) s-stage IRK schemes

Table 1: Butcher tableaux for Runge-Kutta schemes

The Radau IIA IRK schemes are L-stable with order $2s-1$ and are fully constructed in Ref. 17. This article will exclusively consider Radau IIA IRK schemes because they possess two properties the following derivations require: $\boldsymbol{A}$ is invertible, and $\boldsymbol{b}^T\boldsymbol{A}^{-1} = \boldsymbol{e}_s = (0,\ldots,0,1)$. The Butcher tableaux for these schemes are given in Appendix A.

The nonlinear system (4) that arises from these schemes is solved by Newton's method, an iterative process that requires the solution of a linear system of equations at every step. These large, sparse, linear systems are typically solved by an iterative solver such as GMRES, which approximates the exact solve by a sequence of $\mathcal{O}(s^2)$ matrix-vector multiplications.

Pazner and Persson[13] showed that the cost of this matrix-vector multiplication could be reduced to $\mathcal{O}(s)$ by simply deriving Newton's method for the Runge-Kutta stage updates $\boldsymbol{w}_i^{(n)}$ rather than the stages $\boldsymbol{k}_i^{(n)}$ themselves. We define

$$\boldsymbol{w}_i^{(n)} = \sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j^{(n)}$$

as the i^{th} stage update. Moreover, we stack all the updates into a single vector $\boldsymbol{W}^{(n)}$, all the stages into a single vector $\boldsymbol{K}^{(n)}$, s copies of the known solution $\boldsymbol{u}^{(n)}$ into a single vector $\boldsymbol{U}^{(n)}$, and the right hand side function $\boldsymbol{f}$ applied component-wise to each of the s inputs into the vector $\boldsymbol{F}$. Then we can rewrite (4) as

$$\left(\mathbb{I}_s \otimes \mathbb{M}\right)\boldsymbol{K}^{(n)} = \boldsymbol{F}\left(\boldsymbol{U}^{(n)} + \Delta t^{(n)}\boldsymbol{W}^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)}\boldsymbol{c}\right) \tag{5}$$

utilizing Kronecker product notation. In the case that the Butcher matrix $\boldsymbol{A}$ is invertible (such as for all Radau IIA schemes), this is equivalent to

$$\left(\boldsymbol{A}^{-1} \otimes \mathbb{M}\right)\boldsymbol{W}^{(n)} = \boldsymbol{F}\left(\boldsymbol{U}^{(n)} + \Delta t^{(n)}\boldsymbol{W}^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)}\boldsymbol{c}\right) \tag{6}$$

utilizing the Kronecker relation

$$\left(\boldsymbol{A}^{-1} \otimes \mathbb{M}\right)\boldsymbol{W}^{(n)} = \left(\boldsymbol{A}^{-1} \otimes \mathbb{M}\right)\left(\boldsymbol{A} \otimes \mathbb{I}_n\right)\boldsymbol{K}^{(n)} = \left(\mathbb{I}_s \otimes \mathbb{M}\right)\boldsymbol{K}^{(n)}.$$

Moreover, the new solution $\boldsymbol{u}^{(n+1)}$ is updated as

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + \Delta t^{(n)}\left(\boldsymbol{b}^T\boldsymbol{A}^{-1} \otimes \mathbb{I}_n\right)\boldsymbol{W}^{(n)} = \boldsymbol{u}^{(n)} + \Delta t^{(n)}\boldsymbol{w}_s^{(n)} \tag{7}$$

because, in the case of Radau IIA schemes, $\boldsymbol{b}^T\boldsymbol{A}^{-1} = \boldsymbol{e}_s$. Using Newton's method on this transformed system of equations (6) requires repeatedly solving the linear system:

$$\left(\boldsymbol{A}^{-1} \otimes \mathbb{M} - \Delta t^{(n)}\begin{bmatrix} \mathbb{J}_1 & 0 & \ldots & 0 \\ 0 & \mathbb{J}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbb{J}_s \end{bmatrix}\right)\Delta\boldsymbol{W}^{(n)} = \tilde{\boldsymbol{R}}^{(n)}. \tag{8}$$

The residual vectors $\boldsymbol{r}_i^{(n)} = f - \mathbb{M}\boldsymbol{k}_i^{(n)}$ are stacked into a single vector $\tilde{\boldsymbol{R}}^{(n)}$, and $\mathbb{J}_i$ is the Jacobian matrix of $\boldsymbol{f}$ for the i^{th} input vector. This resulting matrix is an $s \times s$ block matrix such that *computing a matrix-vector product only requires s matrix-vector multiplications with the mass matrix and s matrix-vector multiplications with a Jacobian matrix.* Therefore the sparse matrix-vector products scale as $\mathcal{O}(s)$, a fundamental improvement over the untransformed matrix which scales as $\mathcal{O}(s^2)$. Moreover, memory usage is optimal, at only s times that of a DIRK scheme.

C. Solver-Consistent Quantities of Interest

Given our high-order numerical method for solving the PDE (2), we now return to the CFD optimization problem. In particular, we must determine how to compute the QoI—space-time integrals over the surface of the body $\partial\Omega$—in a solver-consistent manner. Discretization of QoI will introduce an additional discretization error, separate from that in the approximation of $\boldsymbol{u}$. To ensure neither discretization error dominates, thereby lowering the global order of the scheme, it is necessary to equate the discretization orders of these QoI and the PDE solution itself.

These time-dependent QoI take the form:

$$F(\boldsymbol{U}, \boldsymbol{\mu}, t) := \int_0^t \int_{\partial\Omega} f(\boldsymbol{U}, \boldsymbol{\mu}, \tau)\ dS\ d\tau, \tag{9}$$

where F can correspond to either the constraints or objective function for our optimization problem (1). Define f_h as the approximation of $\int_{\partial\Omega} f(\boldsymbol{U}, \boldsymbol{\mu}, t)\ dS$ using the DG shape functions from the spatial discretization of the governing equations. Alternative spatial discretizations of these governing equations will yield different solver-consistent approximations to this integral. Therefore, our solver-consistent approximation to (9) becomes

$$F_h(\boldsymbol{u}, \boldsymbol{\mu}, t) = \int_0^t f_h(\boldsymbol{u}, \boldsymbol{\mu}, \tau)\ d\tau,$$

or, equivalently

$$\dot{F}_h(\boldsymbol{u}, \boldsymbol{\mu}, t) = f_h(\boldsymbol{u}, \boldsymbol{\mu}, t). \tag{10}$$

The QoI semi-discrete formulation (10) can be combined with the semi-discrete governing equations (3) to yield the augmented system:

$$\begin{bmatrix} \mathbb{M} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{u}} \\ \dot{F}_h \end{bmatrix} = \begin{bmatrix} \boldsymbol{r} \\ f_h \end{bmatrix}. \tag{11}$$

Temporally integrating this system with the Radau IIA schemes derived in Section II.B yields the fully discrete equations:

$$\begin{aligned} \boldsymbol{u}^{(n+1)} &= \boldsymbol{u}^{(n)} + \Delta t^{(n)} \boldsymbol{w}_s^{(n)} \\ F_h^{(n+1)} &= F_h^{(n)} + \Delta t^{(n)} \sum_{i=1}^{s} b_i f_h \left(\boldsymbol{u}^{(n)} + \Delta t^{(n)} \boldsymbol{w}_i^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)} c_i \right) \\ \left(\boldsymbol{A}^{-1} \otimes \mathbb{M} \right) \boldsymbol{W}^{(n)} &= \boldsymbol{F} \left(\boldsymbol{U}^{(n)} + \Delta t^{(n)} \boldsymbol{W}^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)} \boldsymbol{c} \right). \end{aligned} \tag{12}$$

In (12), $n = 0, \ldots, N_t - 1$, and we have reused the vector-stacking notation from Section II.B to define $\boldsymbol{W}^{(n)}$, $\boldsymbol{F}$, and $\boldsymbol{U}^{(n)}$. After solving these equations, we can compute the solver-consistent QoI by evaluating the functional F at the final time $t = T$ as

$$F(\boldsymbol{U}, \boldsymbol{\mu}, T) \approx F_h^{(N_t)} = F_h \left(\boldsymbol{u}^{(0)}, \ldots, \boldsymbol{u}^{(N_t)}, \boldsymbol{W}^{(0)}, \ldots, \boldsymbol{W}^{(N_t-1)}, \boldsymbol{\mu} \right). \tag{13}$$

III. Fully Discrete, Time-Dependent Adjoint Equations

Now we turn to the derivation of the adjoint method for this solver-consistent QoI, F, from (13). Specifically, we need to compute its total derivative,

$$\frac{dF}{d\boldsymbol{\mu}} = \frac{\partial F}{\partial \boldsymbol{\mu}} + \sum_{n=0}^{N_t} \frac{\partial F}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \sum_{n=0}^{N_t-1} \sum_{i=1}^{s} \frac{\partial F}{\partial \boldsymbol{w}_i^{(n)}} \frac{\partial \boldsymbol{w}_i^{(n)}}{\partial \boldsymbol{\mu}}, \tag{14}$$

in a way that does not depend on the sensitivities of the state variables $\frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}}, \frac{\partial \boldsymbol{w}_i^{(n)}}{\partial \boldsymbol{\mu}}$, because these sensitivities are intractable to compute when N_μ is large. Once we provide a computationally feasible expression for this derivative, we can utilize any desired gradient-based, black-box optimization solver to optimize our CFD problem. We will discuss this further in Section III.B.

A. Derivation

We derive an alternative expression for the total derivative (14) by introducing the adjoint equations for a functional F and its corresponding dual variables. The adjoint equation will allow a reconstruction of the total derivative of F independent of these state sensitivities.

First, we define the residuals to (12). That is, for $n = 0, \ldots, N_t - 1$,

$$\begin{aligned} 0 = \boldsymbol{r}^{(0)} &:= \boldsymbol{u}^{(0)} - \boldsymbol{u}_0(\boldsymbol{\mu}) \\ 0 = \boldsymbol{r}^{(n+1)} &:= \boldsymbol{u}^{(n+1)} - \boldsymbol{u}^{(n)} - \Delta t^{(n)} \boldsymbol{w}_s^{(n)} \\ 0 = \boldsymbol{R}^{(n)} &:= \left(\boldsymbol{A}^{-1} \otimes \mathbb{M}\right) \boldsymbol{W}^{(n)} - \boldsymbol{F}\left(\boldsymbol{U}^{(n)} + \Delta t^{(n)} \boldsymbol{W}^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)} \boldsymbol{c}\right). \end{aligned} \tag{15}$$

Now we can differentiate each of these equations with respect to the parameters $\boldsymbol{\mu}$ to get the fully discrete sensitivity equations:

$$\begin{aligned} 0 &= \frac{\partial \boldsymbol{r}^{(0)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(0)}}{\partial \boldsymbol{u}^{(0)}} \frac{\partial \boldsymbol{u}^{(0)}}{\partial \boldsymbol{\mu}} \\ 0 &= \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{u}^{(n+1)}} \frac{\partial \boldsymbol{u}^{(n+1)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{w}_s^{(n)}} \frac{\partial \boldsymbol{w}_s^{(n)}}{\partial \boldsymbol{\mu}} \\ 0 &= \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \sum_{j=1}^{s} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{w}_j^{(n)}} \frac{\partial \boldsymbol{w}_j^{(n)}}{\partial \boldsymbol{\mu}}. \end{aligned} \tag{16}$$

We introduce the dual variables $\boldsymbol{\lambda}^{(0)}, \boldsymbol{\lambda}^{(n)}, \boldsymbol{\omega}_i^{(n)} \in \mathbb{R}^{N_u}$. As in Section II.B, we stack all s $\boldsymbol{\omega}_i^{(n)}$ for a given time step into a single vector $\boldsymbol{\Omega}^{(n)} \in \mathbb{R}^{sN_u}$. For any value of these dual variables, multiplying the sensitivity equations by these duals will still yield the $\mathbf{0}$ vector. Therefore, we can subtract all the sensitivity equations (16), multiplied by our dual variables, from (14) to yield:

$$\begin{aligned} \frac{dF}{d\boldsymbol{\mu}} = \frac{\partial F}{\partial \boldsymbol{\mu}} &+ \sum_{n=0}^{N_t} \frac{\partial F}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \sum_{n=0}^{N_t-1} \sum_{i=1}^{s} \frac{\partial F}{\partial \boldsymbol{w}_i^{(n)}} \frac{\partial \boldsymbol{w}_i^{(n)}}{\partial \boldsymbol{\mu}} - \boldsymbol{\lambda}^{(0)^T} \left(\frac{\partial \boldsymbol{r}^{(0)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(0)}}{\partial \boldsymbol{u}^{(0)}} \frac{\partial \boldsymbol{u}^{(0)}}{\partial \boldsymbol{\mu}} \right) \\ &- \sum_{n=0}^{N_t-1} \boldsymbol{\lambda}^{(n+1)^T} \left(\frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{u}^{(n+1)}} \frac{\partial \boldsymbol{u}^{(n+1)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{w}_s^{(n)}} \frac{\partial \boldsymbol{w}_s^{(n)}}{\partial \boldsymbol{\mu}} \right) \\ &- \sum_{n=0}^{N_t-1} \boldsymbol{\Omega}^{(n)^T} \left(\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{\mu}} + \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \sum_{j=1}^{s} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{w}_j^{(n)}} \frac{\partial \boldsymbol{w}_j^{(n)}}{\partial \boldsymbol{\mu}} \right). \end{aligned} \tag{17}$$

Now, rearranging the terms to isolate the state sensitivities, we have:

$$\begin{aligned}\frac{dF}{d\boldsymbol{\mu}} = \frac{\partial F}{\partial \boldsymbol{\mu}} &- \sum_{n=0}^{N_t} \boldsymbol{\lambda}^{(n)^T} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{\mu}} - \sum_{n=0}^{N_t-1} \boldsymbol{\Omega}^{(n)^T} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{\mu}} + \left(\frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}} - \boldsymbol{\lambda}^{(N_t)^T} \frac{\partial \boldsymbol{r}^{(N_t)}}{\partial \boldsymbol{u}^{(N_t)}} \right) \frac{\partial \boldsymbol{u}^{(N_t)}}{\partial \boldsymbol{\mu}} \\ &+ \sum_{n=0}^{N_t-1} \left(\frac{\partial F}{\partial \boldsymbol{u}^{(n)}} - \boldsymbol{\lambda}^{(n+1)^T} \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{u}^{(n)}} - \boldsymbol{\lambda}^{(n)^T} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} - \boldsymbol{\Omega}^{(n)^T} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \right) \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} \\ &+ \sum_{n=0}^{N_t-1} \left(\frac{\partial F}{\partial \boldsymbol{w}_s^{(n)}} - \boldsymbol{\lambda}^{(n+1)^T} \frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{w}_s^{(n)}} - \boldsymbol{\Omega}^{(n)^T} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{w}_s^{(n)}} \right) \frac{\partial \boldsymbol{w}_s^{(n)}}{\partial \boldsymbol{\mu}} \\ &+ \sum_{n=0}^{N_t-1} \sum_{i=1}^{s-1} \left(\frac{\partial F}{\partial \boldsymbol{w}_i^{(n)}} - \boldsymbol{\Omega}^{(n)^T} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{w}_i^{(n)}} \right) \frac{\partial \boldsymbol{w}_i^{(n)}}{\partial \boldsymbol{\mu}}.\end{aligned} \tag{18}$$

Equation 18 holds true for all dual variables. Thus, to simplify our computation of the total derivative, we only consider dual variables such that the terms in parentheses are identically zero. This yields the system of equations:

$$\begin{aligned}\left(\frac{\partial \boldsymbol{r}^{(N_t)}}{\partial \boldsymbol{u}^{(N_t)}} \right)^T \boldsymbol{\lambda}^{(N_t)} &= \left(\frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}} \right)^T \\ \left(\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \right)^T \boldsymbol{\lambda}^{(n)} &= \left(\frac{\partial F}{\partial \boldsymbol{u}^{(n)}} \right)^T - \left(\frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{u}^{(n)}} \right)^T \boldsymbol{\lambda}^{(n+1)} - \left(\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \right)^T \boldsymbol{\Omega}^{(n)} \\ \left(\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{W}^{(n)}} \right)^T \boldsymbol{\Omega}^{(n)} &= \left(\frac{\partial F}{\partial \boldsymbol{W}^{(n)}} \right)^T - \boldsymbol{e}_s \otimes \left(\frac{\partial \boldsymbol{r}^{(n+1)}}{\partial \boldsymbol{w}_s^{(n)}} \right)^T \boldsymbol{\lambda}^{(n+1)}.\end{aligned} \tag{19}$$

Analytically evaluating the sensitivities of our residuals from their definitions in (15), we rewrite (19) as

$$\begin{aligned}\boldsymbol{\lambda}^{(N_t)} &= \left(\frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}} \right)^T \\ \boldsymbol{\lambda}^{(n)} &= \boldsymbol{\lambda}^{(n+1)} + \left(\frac{\partial F}{\partial \boldsymbol{u}^{(n)}} \right)^T + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}} \left(\boldsymbol{U}^{(n)} + \Delta t^{(n)} \boldsymbol{W}^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)} \boldsymbol{c} \right)^T \boldsymbol{\Omega}^{(n)} \\ &\left(\boldsymbol{A}^{-1} \otimes \mathbb{M} - \Delta t^{(n)} \begin{bmatrix} \mathbb{J}_1 & 0 & \dots & 0 \\ 0 & \mathbb{J}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbb{J}_s \end{bmatrix} \right)^T \boldsymbol{\Omega}^{(n)} = \left(\frac{\partial F}{\partial \boldsymbol{W}^{(n)}} \right)^T + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \Delta t^{(n)} \boldsymbol{\lambda}^{(n+1)} \end{bmatrix}.\end{aligned} \tag{20}$$

Equation 20 is the fully discrete adjoint equation, a dual system of equations to our primal problem (12). Note that solving this system for $\boldsymbol{\lambda}^{(n)}$ and $\boldsymbol{\Omega}^{(n)}$ is computationally similar to solving the primal problem for $\boldsymbol{u}^{(n)}$ and $\boldsymbol{W}^{(n)}$. This is because computing either the Runge-Kutta stage updates or the dual variables at each time step requires solving a $sN_u \times sN_u$ system of equations (which due to its convenient block structure can be performed as a sequence of $\mathcal{O}(s)$ matrix-vector multiplications) before updating the solution to the next time step. However, while the discrete primal problem is nonlinear, the discrete adjoint problem is linear and therefore only requires a single solution of a linear system. Lastly, we note that while the discrete primal problem steps forwards in time from $n = 0$ to $n = N_t$, the dual problem steps backwards in time from $n = N_t$ to $n = 0$ with a well-defined "initial condition" at $n = N_t$.

After solving this dual problem for our variables $\boldsymbol{\lambda}^{(n)}, \boldsymbol{\Omega}^{(n)}$, we return to (18) and simplify the expression

for the total derivative:

$$\begin{aligned}\frac{dF}{d\boldsymbol{\mu}} &= \frac{\partial F}{\partial \boldsymbol{\mu}} - \sum_{n=0}^{N_t} \boldsymbol{\lambda}^{(n)^T} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{\mu}} - \sum_{n=0}^{N_t-1} \boldsymbol{\Omega}^{(n)^T} \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{\mu}} \\ &= \frac{\partial F}{\partial \boldsymbol{\mu}} + \boldsymbol{\lambda}^{(0)^T} \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{\mu}} + \sum_{n=0}^{N_t-1} \boldsymbol{\Omega}^{(n)^T} \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\mu}} \left(\boldsymbol{U}^{(n)} + \Delta t^{(n)} \boldsymbol{W}^{(n)}, \boldsymbol{\mu}, t^{(n)} + \Delta t^{(n)} \boldsymbol{c} \right).\end{aligned} \tag{21}$$

Of critical importance, we note that (21) does not depend on state sensitivities, save for $\frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{\mu}}$. This one term does not destroy the efficiency of our adjoint method for two reasons: (1) only matrix-vector products with this term are required, and (2) the initial condition is either known analytically or is the solution of some nonlinear system of equations. Zahr and Persson[3] showed that at worst case this matrix-vector product can be computed at the cost of one linear solve of size $N_u \times N_u$ and one inner product of size $\mathcal{O}(N_\mu)$.

B. Fully Discrete Framework

With the ability to evaluate a QoI using (12) and compute its derivative using (21), we now return to our CFD optimization problem (1). Using any gradient-based optimization solver, such as a quasi-Newton method,[18] we can iteratively improve an approximation to the local minimum of this problem until we are within a desired tolerance. But why use a fully discrete framework over the more traditional, continuous adjoint method?

Firstly, the fully discrete framework ensures that the QoI evaluation and derivative computations are solver-consistent. This property is important because practical convergence rates, and even many theoretical convergence bounds, depend on consistent gradients of optimization functionals.

Secondly, when Runge-Kutta schemes are used for the temporal discretization, the fully discrete framework avoids any complications with asymmetrical stage nodes. For a general Runge-Kutta scheme, the stage nodes $\boldsymbol{c}$ are not symmetric about 1/2. During a primal solve, the solution $\boldsymbol{u}$ is computed at each time step and at $t^{(n)} + \Delta t^{(n)} c_i$ for $i = 1, \ldots, s$. But using the same Runge-Kutta scheme during the backward solve of the semi-discrete or continuous adjoint equations, the solution $\boldsymbol{\lambda}$ is computed at each time step and at $t^{(n+1)} - \Delta t^{(n)} c_i$ for $i = 1, \ldots, s$. In general, this is a different set of times, so the primal solutions must be interpolated to these new stage nodes. Beyond complicating any implementations, this degrades accuracy at the intermediate stages. One major benefit of the IRK schemes we are using in this work is their high stage order. *So using a semi-discrete or continuous adjoint method with IRK schemes would be knowingly decreasing accuracy at the exact place where the accuracy of these IRK schemes is most coveted.* Happily, this issue does not arise in the fully discrete setting as only terms computed during the primal solve appear in the adjoint equations, by construction.

Lastly, the fully discrete adjoint equation applied to these IRK schemes yields a convenient linear system of equations. Compare the matrix in (20) with that in (8). Because their structure and sparsity pattern are identical, we can reuse solvers and preconditioners developed for the primal equations when solving the adjoint equations. We confirm that their effectiveness transfers to the adjoint equations in Section IV.D. There is no guarantee that such a convenient property arises when discretizing the semi-discrete or continuous adjoint method.

IV. Numerical Results

A. Energetically Optimal Trajectory of 2D Airfoil in Compressible, Viscous Flow

In this section, we apply the high-order, PDE-constrained optimization framework developed in this article to find the energetically optimal trajectory of an airfoil immersed in an isentropic, viscous flow. The governing equations are the 2D compressible, isentropic Navier-Stokes equations:

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) &= 0 \\
\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot \left(\rho \boldsymbol{u} \otimes \boldsymbol{u}^T + p \boldsymbol{I}_{N_{sd}} - \boldsymbol{\tau}\right) &= 0 \\
\frac{\partial (\rho E)}{\partial t} + \nabla \cdot ((\rho E + p)\boldsymbol{u} + \boldsymbol{q} - \boldsymbol{\tau}\boldsymbol{u}) &= 0.
\end{aligned} \tag{22}
$$

where ρ is the fluid density, $\boldsymbol{u}$ is the fluid velocity, E is the total energy, p is the pressure, and $\boldsymbol{I}_{N_{sd}}$ is the $N_{sd} \times N_{sd}$ identity tensor. The viscous stress tensor $\boldsymbol{\tau}$ and heat flux $\boldsymbol{q}$ are given by

$$
\boldsymbol{\tau} := \mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T - \frac{2}{3} (\nabla \cdot \boldsymbol{u}) \boldsymbol{I}_{N_{sd}} \right), \qquad \boldsymbol{q} := -\frac{\mu}{Pr} \nabla \left(E + \frac{p}{\rho} - \frac{1}{2} \|\boldsymbol{u}\|_2^2 \right),
$$

where μ is the viscosity coefficient and $Pr = 0.72$ is the Prandtl number, which we assume to be constant. For an ideal gas, the pressure satisfies $p = (\gamma - 1)\rho \left(E - \frac{1}{2} \|\boldsymbol{u}\|_2^2 \right)$, where $\gamma = 1.4$ is the adiabatic gas constant. Furthermore, the entropy of the system is assumed constant, so the flow is adiabatic and reversible. Thus, for a perfect gas, the entropy is defined as $s = p/\rho^\gamma$. Favorably, this equation allows us to relate pressure and density, thereby making the energy equation in (22) redundant. This reduces the number of components of the PDE from $N_{sd} + 2$ to $N_{sd} + 1$, where we recall that $N_{sd} = 2$ is the number of spatial dimensions.

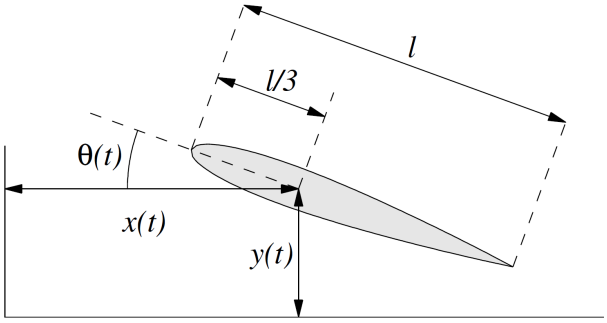

Figure 1: Airfoil specifications

The goal of this optimization problem is to find a trajectory for the airfoil that minimizes energy, subject to a specified thrust constraint. The airfoil is immersed in a flow with Reynolds number 1000 and Mach number 0.1. These parameters were chosen to balance approximate incompressibility and well-conditioned equations. Ω is taken to be a circular domain with radius 100 centered at $(0, 0)$. The airfoil's leading edge is located at the origin. The 2D NACA0012 airfoil, as in Figure 1, has chord length $l = 1$, zero-thickness trailing edge, and a kinematic motion parameterized with a single Fourier mode. Mathematically, we write:

$$
\begin{aligned}
x(\boldsymbol{\mu}, t) &= A_x \sin(\omega_x t + \phi_x) + c_x \\
y(\boldsymbol{\mu}, t) &= A_y \sin(\omega_y t + \phi_y) + c_y \\
\theta(\boldsymbol{\mu}, t) &= A_\theta \sin(\omega_\theta t + \phi_\theta) + c_\theta.
\end{aligned} \tag{23}
$$

For a trajectory to be valid, we must satisfy the thrust constraint,

$$
T(\boldsymbol{U}, \boldsymbol{\mu}) := \int_0^T \int_{\partial\Omega} \boldsymbol{f}(\boldsymbol{U}, \boldsymbol{\mu}, t) \cdot \boldsymbol{e}_1 \, dS \, dt \geq T_0. \tag{24}
$$

Note that $\partial\Omega$ refers to the surface of the airfoil, $\boldsymbol{f}(\boldsymbol{U}, \boldsymbol{u}, t)$ is the instantaneous force that the fluid exerts on the airfoil, $T = 5$ is the dimensionless period of the flow, and $T_0 = 0.7$ is the minimum allowed thrust over a period. The total work done on the airfoil by the fluid, W, can be computed as

$$
W(\boldsymbol{U}, \boldsymbol{\mu}) := \int_0^T \int_{\partial\Omega} \boldsymbol{f}(\boldsymbol{U}, \boldsymbol{\mu}, t) \cdot \dot{\boldsymbol{x}} \, dS \, dt, \tag{25}
$$

where $\dot{\boldsymbol{x}}$ is the velocity of a point on the surface of the airfoil. Therefore, the formal optimization problem we wish to solve is

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{\mu}} \quad & W(\boldsymbol{U},\boldsymbol{\mu}) \\ s.t. \quad & T(\boldsymbol{U},\boldsymbol{\mu}) \geq T_0 \\ & \frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}, \nabla \boldsymbol{U}) = 0 \text{ in } \Omega(\boldsymbol{\mu}, t). \end{aligned} \tag{26}$$

B. Adjoint Gradient Verification

Before considering this optimization problem, we first verify the accuracy of the proposed adjoint method against a more traditional finite difference approach. The finite difference approximation to the gradient requires solving the fully discrete equations (12) at perturbations to a reference parameter configuration. We consider the reference parameters $\boldsymbol{\mu}$:

$$\begin{array}{lll} c_x = 0 & c_y = 0 & c_\theta = 0 \\ A_x = -5/4 & A_y = 3/2 & A_\theta = \pi/8 \\ \omega_x = 2\pi/5 & \omega_y = 2\pi/5 & \omega_\theta = \pi/2 \\ \phi_x = 0 & \phi_y = 0 & \phi_\theta = 0 \end{array}$$

Figure 2 shows the relative error between the gradients computed by our adjoint method and the second-order finite difference approximation across a sweep of finite difference step sizes using IEEE 754 double-precision floating-point numbers. A relative error of 3.4×10^{-10} is observed for a step size of 5.0×10^{-6}. The slope of the last eight data points is exactly 2.0, showing the convergence rate of the finite difference approximation to the adjoint-based gradient. As expected, after the step size becomes too small, the error increases with decreasing step size because of the trade-off between finite difference accuracy and roundoff error.

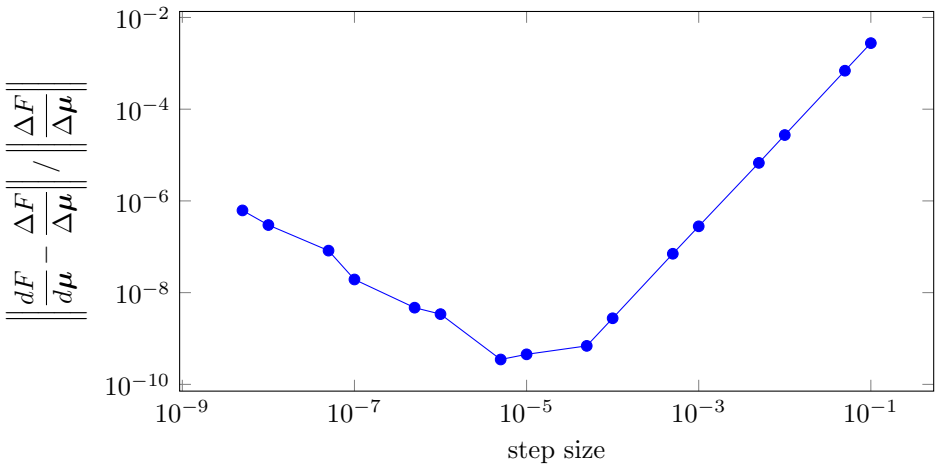

Figure 2: Verification of the adjoint-based gradient with second-order centered finite difference approximation. The adjoint gradient matches the finite difference approximation to about 10 digits of accuracy in double precision before roundoff errors degrade the accuracy.

C. Optimal Trajectory

With the adjoint gradient verified, we now return to the optimization problem (26). For this and all further results, we discretize the domain around the airfoil with a mesh containing 971 triangles that is refined around the airfoil, in particular at the leading and trailing edges. We apply our DG-ALE formulation with polynomial order $p = 3$ to get a system of ODEs with 38,840 degrees of freedom. All of our IRK schemes use $\Delta t^{(n)} = 5 \times 10^{-2}$, a reasonably large value that exceeds the CFL condition of most explicit time integrators.

We use a limited memory quasi-Newton method to solve sequential quadratic programming approximations to our nonlinear optimization problem, implemented in the SNOPT software package.[19]

Figure 3 shows the initial guess for the optimization problem, a purely vertical oscillation, and the achieved optimal trajectory using the Radau23 adjoint method. The initial guess does not satisfy the thrust constraint with $W = 1.8$; however, the optimal trajectory generates the required thrust with $W = 9.8 \times 10^{-2}$. Indeed, at the optimal solution, the total work required to perform this periodic flapping motion is more than an order of magnitude smaller than at the initial guess.

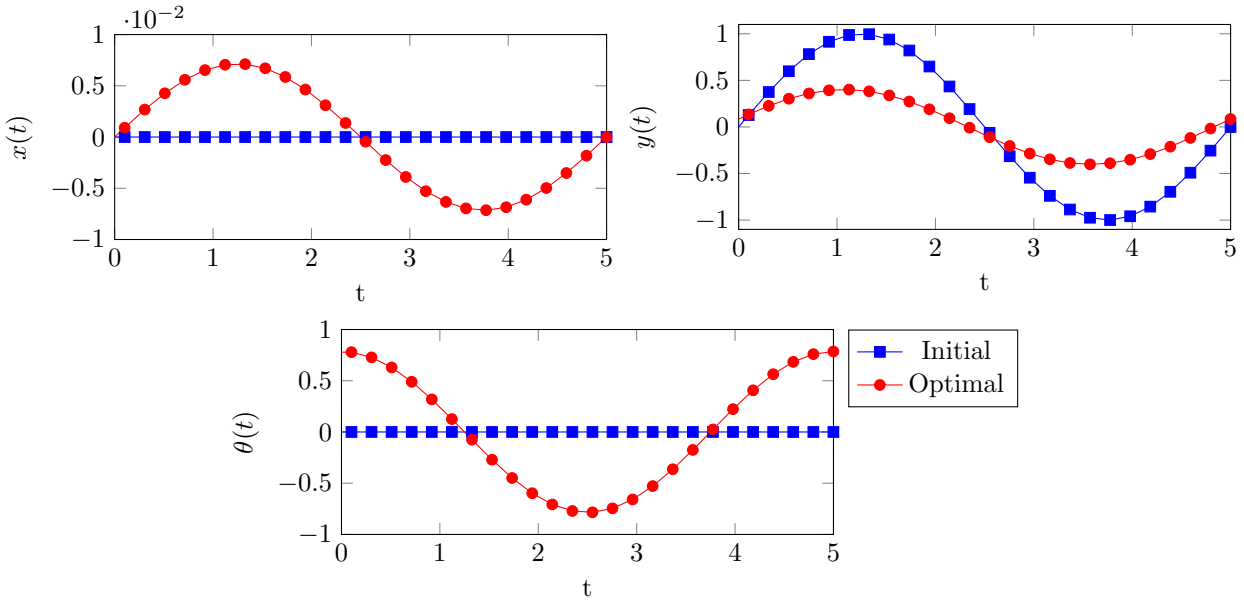

Figure 3: Initial guess and energetically optimal $x(t)$, $y(t)$, and $\theta(t)$ trajectories over one period $T = 5$

Figures 4 and 5 show vorticity snapshots created by the initial guess and energetically optimal trajectories over one period. Snapshots are taken at $t_1 = 0$, $t_2 = T/4 = 1.25$, $t_3 = T/2 = 2.5$, and $t_4 = 3T/4 = 3.75$ of the third oscillation period. Figure 4 shows flow separation off the leading edge and vortex shedding, implying a large amount of work is required to complete each oscillation. Figure 5 shows that the optimal trajectory produces less flow separation and reduced amounts of shedding, consequently reducing the total work required.

Figure 6 contains a convergence study of three time integrators for our numerical optimizer. We consider the 1-stage, first-order Radau11 scheme (Backward Euler), 2-stage, third-order Radau23 scheme, and the 3-stage, fifth-order Radau35 scheme. The y-axis is relative error in total work $|W - W^*|$, where W^* is the final total work computed when the optimizer converges to a solution. Using the Radau11, Radau23, and Radau35 schemes, the optimizer computes exactly 17, 15, and 14 iterations, respectively. We note that these iteration counts are approximately the same and quite low, highlighting a benefit of computing high-order gradients with discrete consistency for our black-box optimizer. While the curves appears to struggle after reaching 3 digits of accuracy, the overall convergence is still superlinear, with estimated convergence rates of 1.4, 1.3, and 1.8, respectively.

D. Performance Analysis

In this section, we compare the performance of our IRK adjoint method with the previously derived DIRK adjoint method from Ref. 3. Specifically, we compare DIRK33, an L-stable, 3-stage, third-order DIRK scheme,[12] to the equal-order Radau23 scheme, and SDIRK55, an L-stable, 5-stage, fifth-order singly DIRK scheme,[20] to the equal-order Radau35 scheme[a]. Appendix A contains the Butcher tableaux for these schemes.

[a]We initially tested ESDIRK65,[21] a 6-stage, fifth-order DIRK scheme that has an explicit first step. However, we found stability issues using our large Δt. ESDIRK65 was only stable after decreasing Δt by two orders of magnitude.

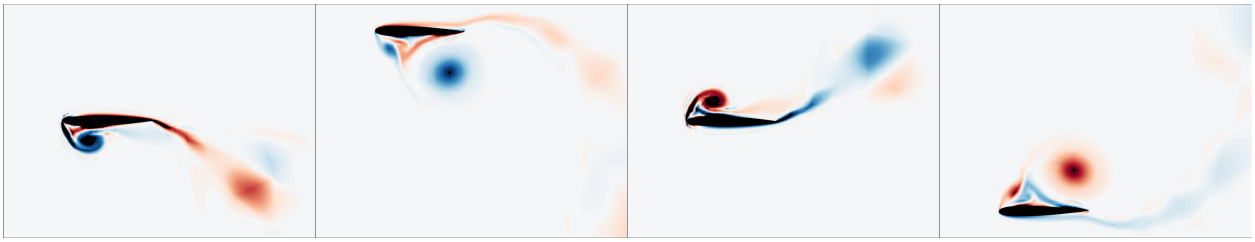

Figure 4: Vorticity snapshots of initial trajectory at $t = 0$, $T/4$, $T/2$, and $3T/4$

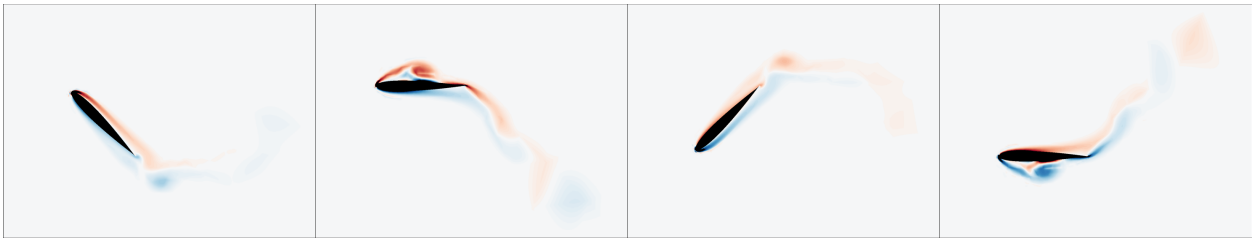

Figure 5: Vorticity snapshots of optimal trajectory at $t = 0$, $T/4$, $T/2$, and $3T/4$

The DIRK systems are preconditioned with an ILU(0) factorization, and the IRK systems are preconditioned with the stage-uncoupled ILU(0) factorization introduced in Ref. 13. These performance tests set the tolerance for Newton's method to 10^{-8} and the relative, preconditioned tolerance of GMRES to 5×10^{-4}.

From Figure 6, we know that the number of iterations our optimizer requires to converge is approximately the same for all IRK schemes. Therefore, we analyze the overall performance of our adjoint method by instead measuring the performance of one optimization iteration. From Section II.B and Section III.A, we know that the cost of these methods is dominated by the matrix-vector multiplications. Therefore, we compute the number of $N_u \times N_u$ matrix-vector products across all time steps of one representative optimization iteration—N_t forward time steps of the primal problem and N_t backward time steps of the dual problem. We refer to this quantity as the number of *equivalent multiplications* and use it as a machine- and optimizer-independent measure of performance. For DIRK schemes, the equivalent multiplications per time step is equal to the number of GMRES iterations per time step. For IRK schemes, each multiplication by the large block matrix from (8) essentially consists of s matrix-vector products of size $N_u \times N_u$. So the equivalent multiplications per time step is equal to s times the number of GMRES iterations per time step.

Figure 7 plots the equivalent multiplications for our Runge-Kutta schemes against the number of distributed-memory compute processes. First, we notice that the higher order schemes cost more than the lower order schemes, but scale at the same rate. Therefore, if temporal accuracy is important, these high-order schemes should out-perform the low-order schemes, no matter the computing scale at which the problem is run.

Furthermore, we can make direct comparisons between the performance of DIRK and IRK adjoint methods. The Radau23 scheme outperforms DIRK33 by roughly 10%, and the improvement slightly increases as the number of processes increases. Likewise, the SDIRK55 scheme outperforms Radau35 by roughly 15%, although the improvement decreases as the number of processes increases. Figure 7 also confirms that the preconditioner initially derived for the forward, linear system (8) also works well for the linear adjoint system (20).

By construction, the ILU(0) preconditioners we use are local operators with no inter-process communication; all inter-element contributions that require communication between processes are ignored. So the preconditioners should perform worse as the domain is partitioned among more processes. Indeed, as the number of partitions approaches the number of mesh elements, the preconditioner reduces to a block Jacobi preconditioner.[22] As expected, Figure 7 shows that the equivalent multiplications increases for all schemes as the number of domain partitions increases.

These considerations apply to both the DIRK and IRK linear systems; however, the stage-uncoupled ILU(0)

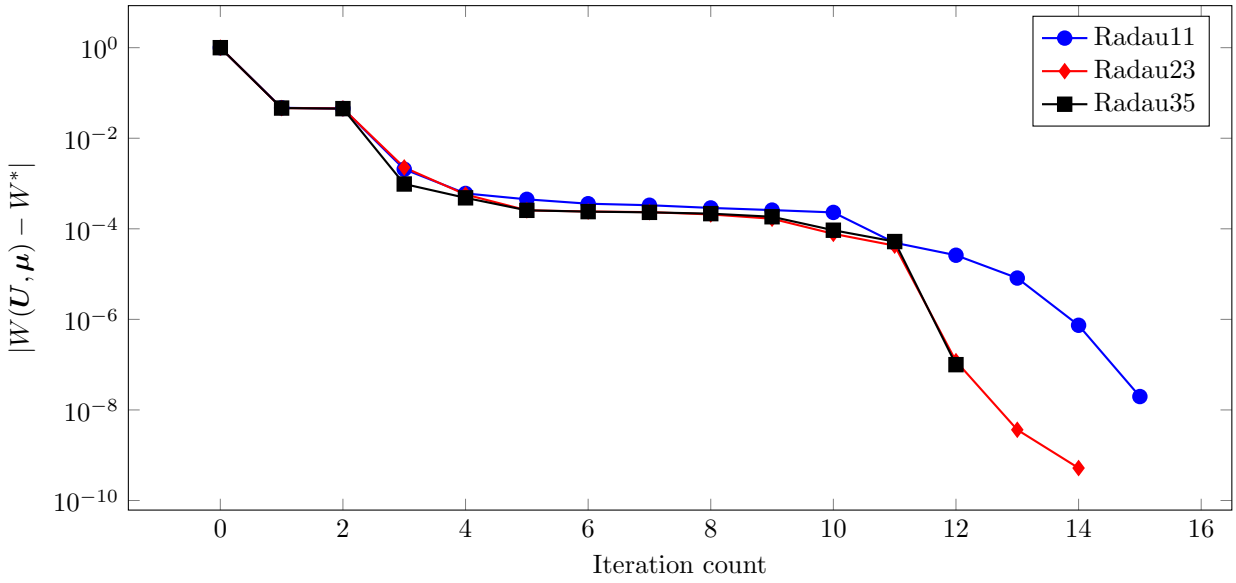

Figure 6: Convergence of computed work for the optimization problem (26) solved with IRK schemes with temporal order 1, 3, and 5. W^* is taken to be the work computed at the final iteration for each of the schemes. Radau11, Radau23, and Radau35 reach the desired 10^{-9} tolerance after 15, 14, and 12 iterations, respectively. To confirm this convergence, the optimizer computes 17, 15, and 14 iterations.

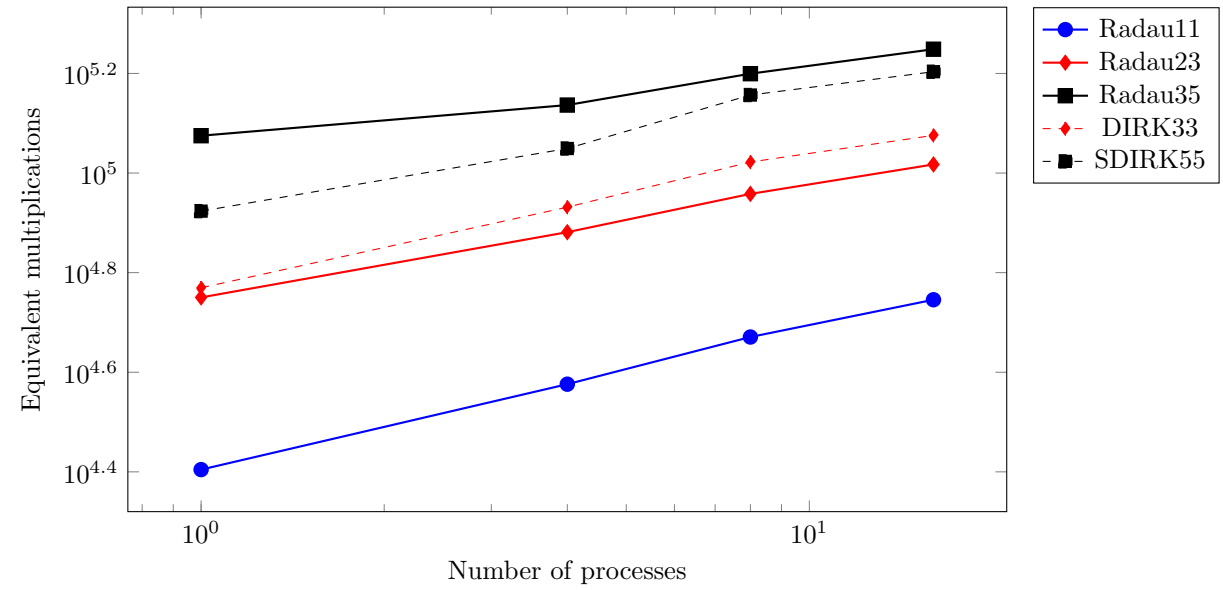

Figure 7: Comparison of equivalent multiplications between IRK (solid) and DIRK (dashed) schemes for one representative optimization iteration. Radau23 outperforms DIRK33 by roughly 10%. SDIRK55 outperforms Radau35 by roughly 15%. Each equivalent multiplication is one $N_u \times N_u$ matrix-vector multiplication.

preconditioner for our IRK schemes can be modified to handle a domain that is decomposed into a factor of s fewer partitions. Instead of assigning each process to one partition, we can instead assign s processes to one partition. Each process will then be responsible for one stage of the IRK scheme. With this setup, the assembly of Jacobian matrices will be local to each set of s processes, and the precomputation and application of our uncoupled ILU(0) preconditioner will be local to each process. We expect that this stage-parallel solver should have the benefit of translating the Radau IIA results in Figure 7 to the right roughly by a factor of s. While we have not implemented this parallel-in-time solver, analyzing its benefits to our fully discrete IRK adjoint method would be an interesting topic for future research.

V. Conclusions

This article derived the fully discrete adjoint equations to create a globally high-order numerical method for computing gradients for an optimization problem. Fully implicit Runge-Kutta (IRK) schemes were used for temporal integration so that our method benefits from high-order stability and high stage order. By transforming the formulation of these IRK schemes, the linear systems that dominate the cost of our IRK adjoint method are computationally competitive with those formed from diagonally implicit Runge-Kutta (DIRK) schemes. We detailed the computation of solver-consistent quantities of interest and their exact gradients for solving PDE-constrained optimization problems. This fully discrete IRK adjoint method has the benefit of matching the order of the discretization errors of the optimization inputs and PDE solution. In contrast to semi-discrete or continuous adjoint methods, this property ensures that we maintain intermediate stage accuracy—exactly where the high stage order of IRK schemes is most coveted.

In the context of gradient-based optimization, our implementation of the IRK adjoint method was used to solve an optimization problem constrained by the isentropic, compressible Navier-Stokes equations. This work verified the accuracy of these gradients, and rapid convergence to the optimal solution was noted for several IRK schemes. Moreover, we showed that the parallel performance of the IRK adjoint methods was comparable to the DIRK adjoint methods of the same order. We expect that the development and application of a stage-parallel solver to our adjoint method will further highlight its computational benefits.

Appendix A Runge-Kutta Butcher Tableaux

1	1
	1

Backward Euler (Radau11) scheme

$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

Radau23 scheme

$\frac{2}{5}-\frac{\sqrt{6}}{10}$	$\frac{11}{45}-\frac{7\sqrt{6}}{360}$	$\frac{37}{225}-\frac{169\sqrt{6}}{1800}$	$-\frac{2}{225}+\frac{\sqrt{6}}{75}$
$\frac{2}{5}+\frac{\sqrt{6}}{10}$	$\frac{37}{225}-\frac{169\sqrt{6}}{1800}$	$\frac{11}{45}-\frac{7\sqrt{6}}{360}$	$-\frac{2}{225}-\frac{\sqrt{6}}{75}$
1	$\frac{4}{9}-\frac{\sqrt{6}}{36}$	$\frac{4}{9}+\frac{\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{4}{9}-\frac{\sqrt{6}}{36}$	$\frac{4}{9}+\frac{\sqrt{6}}{36}$	$\frac{1}{9}$

Radau35 scheme

α	α	0	0
$\frac{1+\alpha}{2}$	$\frac{1-\alpha}{2}$	α	0
1	b_1	b_2	α
	b_1	b_2	α

DIRK33 scheme

$$\alpha = 1 + \frac{\sqrt{6}}{2}\sin\left(\frac{1}{3}\arctan\left(\frac{\sqrt{2}}{4}\right)\right) - \frac{\sqrt{2}}{2}\cos\left(\frac{1}{3}\arctan\left(\frac{\sqrt{2}}{4}\right)\right)$$

$$b_1 = -\frac{1}{4}\left(6\alpha^2 - 16\alpha + 1\right)$$

$$b_2 = \frac{1}{4}\left(6\alpha^2 - 20\alpha + 5\right)$$

.
.

$\frac{4024571134387}{14474071345096}$	$\frac{4024571134387}{14474071345096}$	0	0	0	0
$\frac{5555633399575}{5431021154178}$	$\frac{9365021263232}{12572342979331}$	$\frac{4024571134387}{14474071345096}$	0	0	0
$\frac{5255299487392}{12852514622453}$	$\frac{2144716224527}{9320917548702}$	$\frac{-397905335951}{4008788611757}$	$\frac{4024571134387}{14474071345096}$	0	0
$\frac{3}{20}$	$\frac{-291541413000}{6267936762551}$	$\frac{226761949132}{4473940808273}$	$\frac{-1282248297070}{9697416712681}$	$\frac{4024571134387}{14474071345096}$	0
$\frac{10449500210709}{14474071345096}$	$\frac{-2481679516057}{4626464057815}$	$\frac{-197112422687}{6604378783090}$	$\frac{3952887910906}{9713059315593}$	$\frac{4906835613583}{8134926921134}$	$\frac{4024571134387}{14474071345096}$
	$\frac{-2522702558582}{12162329469185}$	$\frac{1018267903655}{12907234417901}$	$\frac{4542392826351}{13702606430957}$	$\frac{5001116467727}{12224457745473}$	$\frac{1509636094297}{3891594770934}$

SDIRK55 scheme

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1752814. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was also performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Moreover, this work was also supported by the National Aeronautics and Space Administration (NASA) under grant number NNX16AP15A, and by the AFOSR Computational Mathematics program under grant number FA9550-15-1-0010. Additionally, this work was supported in part by the Luis W. Alvarez Postdoctoral Fellowship (MZ), by the Director, Office of Science, Office of Advanced Scientific Computing Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231 (MZ, PP).

References

[1]Rajnarayan, D., Haas, A., and Kroo, I., "A multifidelity gradient-free optimization method and application to aerodynamic design," *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2008, AIAA-2008-6020.

[2]Nocedal, J. and Wright, S. J., *Numerical optimization*, Springer Series in Operations Research, Springer-Verlag, New York, 1999.

[3]Zahr, M. J. and Persson, P.-O., "An adjoint method for a high-order discretization of deforming domain conservation laws for optimization of flow problems," *Journal of Computational Physics*, Vol. 326, 2016, pp. 516–543.

[4]Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., et al., "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, pp. 811–845.

[5]Nielsen, E. J., Diskin, B., and Yamaleev, N. K., "Discrete adjoint-based design optimization of unsteady turbulent flows on dynamic unstructured grids," *AIAA journal*, Vol. 48, No. 6, 2010, pp. 1195–1206.

[6]Mader, C. A., RA Martins, J., Alonso, J. J., and Der Weide, E. V., "ADjoint: An approach for the rapid development of discrete adjoint solvers," *AIAA journal*, Vol. 46, No. 4, 2008, pp. 863–873.

[7]Heinkenschloss, M. and Vicente, L. N., "Analysis of inexact trust-region SQP algorithms," *SIAM Journal on Optimization*, Vol. 12, No. 2, 2002, pp. 283–302.

[8]Harriman, K., Gavaghan, D., and Suli, E., "The importance of adjoint consistency in the approximation of linear functionals using the discontinuous Galerkin finite element method," Tech. rep., Unspecified, 2004.

[9]Mani, K. and Mavriplis, D. J., "Unsteady discrete adjoint formulation for two-dimensional flow problems with deforming meshes," *AIAA journal*, Vol. 46, No. 6, 2008, pp. 1351–1364.

[10]Zahr, M. J. and Persson, P.-O., "Performance tuning of Newton-GMRES methods for discontinuous Galerkin discretizations of the Navier-Stokes equations," *21st AIAA Computational Fluid Dynamics Conference*, 2013, AIAA-2013-2685.

[11]Frank, R., Schneid, J., and Ueberhuber, C. W., "Order results for implicit Runge–Kutta methods applied to stiff systems," *SIAM journal on numerical analysis*, Vol. 22, No. 3, 1985, pp. 515–534.

[12]Alexander, R., "Diagonally implicit Runge–Kutta methods for stiff ODEs," *SIAM Journal on Numerical Analysis*, Vol. 14, No. 6, 1977, pp. 1006–1021.

[13]Pazner, W. and Persson, P.-O., "Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations," *Journal of Computational Physics*, Vol. 335, 2017, pp. 700–717.

[14]Wang, L. and Persson, P.-O., "High-order Discontinuous Galerkin Simulations on Moving Domains using ALE Formulations and Local Remeshing and Projections," *53rd AIAA Aerospace Sciences Meeting*, 2015, AIAA-2015-0820.

[15]Peraire, J. and Persson, P.-O., "The compact discontinuous Galerkin (CDG) method for elliptic problems," *SIAM Journal on Scientific Computing*, Vol. 30, No. 4, 2008, pp. 1806–1824.

[16]Cockburn, B. and Shu, C.-W., "The local discontinuous Galerkin method for time-dependent convection-diffusion systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, pp. 2440–2463.

[17]Wanner, G. and Hairer, E., "Solving ordinary differential equations II," *Stiff and Differential-Algebraic Problems*, 1991.

[18]Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 23, No. 4, 1997, pp. 550–560.

[19]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131.

[20]Kennedy, C. A. and Carpenter, M. H., "Diagonally implicit runge-kutta methods for ordinary differential equations. a review," Tech. rep., NASA, 2016.

[21]Boom, P. D. and Zingg, D. W., "High-order implicit temporal integration for unsteady compressible fluid flow simulation," *21st AIAA Computational Fluid Dynamics Conference*, 2013, AIAA-2013-2831.

[22]Persson, P.-O., "Scalable parallel Newton-Krylov solvers for discontinuous Galerkin discretizations," *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, 2009, AIAA-2009-606.