AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-0577

Implicit Runge-Kutta Schemes for Segregated Solutions of Unsteady Flow and Adjoint Equations

Lean Fang* and Ping He†
Iowa State University, Ames, IA 50011

Many aerospace engineering design problems require efficient solutions of unsteady flow and adjoint equations. Implicit Runge–Kutta (IRK) is an efficient temporal discretization scheme for unsteady flow solutions because it allows high order accuracy and relaxed Courant numbers. However, existing IRK studies mostly focused on fully coupled solvers for compressible flow. In this paper, we develop an IRK-PIMPLE method using the two-stage Gauss scheme for segregated flow solvers, along with the Gauss IRK-adjoint formulation for efficient gradient computations. We modify the standard iterative PIMPLE method with block Gauss-Seidel sweeps to work with the two-stage Gauss scheme, and we also simplify the Gauss IRK-adjoint formulation to an easy-to-evaluate form. We validate the proposed IRK-PIMPLE method by simulating unsteady flow over a ramp geometry and comparing the resulting flow field against the second-order backward scheme reference. We implement the Gauss IRK-adjoint for the scalar transport problem, and the adjoint derivative agrees reasonably well with the finite difference reference. The proposed IRK-PIMPLE and its corresponding adjoint solver have the potential to significantly reduce the computation cost for time-resolved unsteady optimization using segregated flow solvers.

I. Introduction

In many aerospace design problems, one needs to consider time-resolved unsteady flow characteristics that can not be assumed to be steady-state or periodic. For example, during the transition from vertical take-off to horizontal cruise for urban air mobility aircraft, one is interested in the temporal evolution of aircraft aerodynamics. For hypersonic reentry vehicles, heat would accumulate throughout the entire flight path; therefore, a steady-state or periodic assumption may not be applicable. Gradient-based optimization has been widely used for large-scale design problems in aerospace engineering that involve lots of design variables. The adjoint method is usually chosen for gradient computation because its computational cost is independent of the number of design variables [1, 2].

While the use of gradient-based optimization and the adjoint method has been a common practice for steady-state aerodynamic optimization problems, it remains a major challenge for unsteady problems due to the high computational cost. With the commonly used time-marching method and a finite difference-based temporal scheme, the primal solution process for an unsteady problem marches forward through possibly tens of thousands of time steps, which is primarily limited by the Courant–Friedrichs–Lewy CFL number for numerical stability. Once the time-marching primal solution is done, the adjoint equations need to be repeatedly solved in reverse order from the last time instance so that the gradient can then be computed. Therefore, the computational cost for conducting an unsteady optimization can become prohibitive due to the large number of time steps. In previous work [3, 4], we developed fixed-point and Krylov-based adjoint solvers for the PIMPLE algorithm that allows for a relaxed CFL number (>10) so that fewer time steps can be used without endangering stability. However, using a relatively large time step for unsteady flow simulations may compromise the temporal discretization accuracy.

One approach to alleviating the high computational cost for unsteady optimization while maintaining temporal accuracy is to use the time-spectral method [5, 6]. The time-spectral method can be advantageous in terms of computational cost because it has spectral convergence, which means that it can achieve a high level of accuracy

*PhD Student, Department of Aerospace Engineering, AIAA Student Member.
†Assistant Professor, Department of Aerospace Engineering, AIAA Senior Member; Email: phe@iastate.edu.

with only a few or a dozen modes or collocation points, contrary to the time-marching method that would require tens of thousands of time steps. For aerodynamic design optimization problems that involve periodic flow, He et al. [7] developed a Fourier-based time-spectral method to solve both the primal flow problem and the adjoint equations. Then, He and Martins [8] proposed a hybrid time-spectral method that uses the time-marching method for the primal solution to ensure robustness and uses the time-spectral method to solve the adjoint equations to save computational cost. For general unsteady flow that is not necessarily periodic, Im et al. [9] proposed a mapped Chebyshev pseudospectral method that bypasses the ill-conditioning of the standard Chebyshev points and accurately predicted the flow around an oscillating airfoil. Prasad et al. [10] also developed the adjoint solver for the standard Chebyshev pseudospectral method. The major downside for the time-spectral approach is that a more complex unsteady flow problem usually requires a higher number of modes or collocation points, which inevitably leads to a highly complex fully coupled system that is hard to converge efficiently and robustly.

An alternative approach is to use higher-order implicit Runge-Kutta (IRK) schemes for the time-marching method to significantly reduce the required number of time steps. With an IRK scheme that has a higher order of accuracy than the commonly used backward difference scheme, the CFL number can be relaxed to a large value without compromising stability or temporal accuracy. Jameson [11] has evaluated the Gauss and Radau schemes and proposed a dual time stepping procedure to solve the resulting fully coupled system. Franco et al. [12] implemented the Radau schemes for a compressible flow solver, developed the corresponding discrete adjoint solver, and successfully performed optimizations of a 2D airfoil. The Gauss and Radau IRK schemes have also been widely used in the optimal control community under the name of the pseudospectral or collocation methods [13]. The Radau scheme and its corresponding adjoint capacity have been implemented in a popular trajectory optimization package called Dymos [14].

The aforementioned IRK schemes can also be interpreted as applying the time-spectral method to local time steps to achieve higher-order accuracy. Therefore, they can be viewed as a hybrid approach that combines the time-marching method and the time-spectral method. For each time step, an IRK scheme couples all stage values together and produces a fully coupled system whose complexity increases as the stage order goes up. As a result, unlike the commonly used temporal schemes based on finite difference, the implementation of an IRK scheme is not straightforward and requires special consideration. A suitable iterative method needs to be deployed together with the IRK scheme to efficiently solve the fully-couple system for each time step.

Previous work on IRK schemes for fluid problems has mostly focused on fully coupled solvers for compressible flow. To the best of our knowledge, the application of IRK schemes to segregated solvers widely used for incompressible flow is yet to be established, and corresponding adjoint solvers for the IRK schemes are also much needed. In this work, we develop a higher-order-accurate IRK-PIMPLE method for segregated flow solvers. We modify the standard PIMPLE algorithm with block Gauss-Seidel sweeps to work with the two-stage Gauss scheme. We also derive the discrete adjoint formulation for the Gauss IRK scheme with special consideration for the proposed IRK-PIMPLE method, and we evaluate the adjoint gradient accuracy with the scalar transport problem. We are working on the implementation of the Gauss IRK-adjoint solver for the IRK-PIMPLE method, and we will investigate the Radau scheme and IRK schemes of higher-stage orders in future work.

The rest of the paper is organized as follows. In Section II, we describe the mathematical background of the proposed IRK-PIMPLE method and the Gauss IRK-adjoint formulation. The IRK-PIMPLE flow simulation results and the adjoint gradient verification are presented in Section III, and we summarize our findings in Section IV.

II. Method

In this section, we first elaborate on the standard PIMPLE algorithm for a temporal scheme based on finite difference. Then, we explain the unsteady scalar transport problem that we use as the testbed for the IRK schemes and the corresponding adjoint method. We derive the Gauss IRK scheme from the point of view of the collocation method, and demostrate how we solve the fully coupled systems for each time step with block Gauss Seidel sweeps. We then proceed to derive the adjoint formulation for the two-stage Gauss IRK scheme with special consideration for a segregated solver such as PIMPLE.

A. PIMPLE method for unsteady flow simulation

We consider incompressible, unsteady laminar flow governed by the N-S equations:

$$\nabla \cdot \boldsymbol{U} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U} \cdot \nabla)\boldsymbol{U} + \nabla p - \nabla \cdot \nu(\nabla \boldsymbol{U} + [\nabla \boldsymbol{U}]^T) = 0, \tag{2}$$

where t is the time, p is the pressure, $\boldsymbol{U}$ is the velocity vector $\boldsymbol{U} = [u, v, w]$, and ν is the kinematic eddy viscosity, respectively.

As mentioned above, we solve the above N-S equations using the PIMPLE method, which is a combination of the PISO and SIMPLE algorithms. The steps are briefly summarized as follows.

First, the momentum equation is discretized, and an intermediate velocity field is solved using the pressure field obtained from the previous iteration ($p^{t-\Delta t}$) or an initial guess. Without loss of generality, we assume the first-order Euler scheme is used for temporal discretization.

$$a_P \boldsymbol{U}_P^t = -\sum_N a_N \boldsymbol{U}_N^t + \frac{\boldsymbol{U}_P^{t-\Delta t}}{\Delta t} - \nabla p^{t-\Delta t} = \boldsymbol{H}(\boldsymbol{U}) - \nabla p^{t-\Delta t}, \tag{3}$$

where a is the coefficient resulting from the finite-volume discretization, subscripts P and N denote the control volume cell and all of its neighboring cells, respectively, $\boldsymbol{U}^{t-\Delta t}$ is the velocity from the previous time step, and

$$\boldsymbol{H}(\boldsymbol{U}) = -\sum_N a_N \boldsymbol{U}_N^t + \frac{\boldsymbol{U}_P^{t-\Delta t}}{\Delta t} \tag{4}$$

represents the influence of velocity from all the neighboring cells and from the previous iteration. A new variable ϕ (face flux) is introduced to linearize the convective term:

$$\int_S \boldsymbol{U}\boldsymbol{U} \cdot \mathrm{d}\boldsymbol{S} = \sum_f \boldsymbol{U}_f \boldsymbol{U}_f \cdot \boldsymbol{S}_f = \sum_f \phi \boldsymbol{U}_f, \tag{5}$$

where the subscript f denotes the cell face. ϕ can be obtained from the previous iteration or an initial guess. Solving the momentum equation (3), we obtain an intermediate velocity field that does not yet satisfy the continuity equation.

Next, the continuity equation is coupled with the momentum equation to construct a pressure Poisson equation, and a new pressure field is computed. The discretized form of the continuity equation is

$$\int_S \boldsymbol{U} \cdot \mathrm{d}\boldsymbol{S} = \sum_f \boldsymbol{U}_f \cdot \boldsymbol{S}_f = 0. \tag{6}$$

Instead of using a simple linear interpolation, $\boldsymbol{U}_f$ in this equation is computed by interpolating the cell-centered velocity $\boldsymbol{U}_P$—obtained from the discretized momentum equation (3)—onto the cell face as follows:

$$\boldsymbol{U}_f = \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f. \tag{7}$$

This idea of momentum interpolation was initially proposed by Rhie and Chow [15] and is effective in mitigating the oscillating pressure (checkerboard) issue resulting from the collocated mesh configuration. Substituting Eq. (7) into Eq. (6), we obtain the pressure Poisson equation:

$$\nabla \cdot \left(\frac{1}{a_P} \nabla p\right) = \nabla \cdot \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right). \tag{8}$$

Solving Eq. (8), we obtain an updated pressure field p^t. Then, the new pressure field p^t is used to correct the face flux

$$\phi^t = \boldsymbol{U}_f \cdot \boldsymbol{S}_f = \left[\left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p^t)_f\right] \cdot \boldsymbol{S}_f, \tag{9}$$

and velocity field

$$\boldsymbol{U}^t = \frac{1}{a_P}[\boldsymbol{H}(\boldsymbol{U}) - \nabla p^t]. \tag{10}$$

The $\boldsymbol{H}(\boldsymbol{U})$ term depends on $\boldsymbol{U}$ but has not been updated so far. To account for this, we need to repeatedly solve the Eqs. (4) to (10) (PISO corrector loop). We use two PISO corrector loops in this paper.

In addition to the PISO corrector loop mentioned above, the PIMPLE algorithm repeatedly solves Eqs. (3) to (10) multiple times until all the flow residuals are small (PIMPLE corrector loop). To ensure the PIMPLE stability, we need to under-relax the momentum equation (3) solution and the pressure update after solving the pressure Poisson equation (8), except for the last PIMPLE corrector loop. We use Eqs. (3), (8), and (9) for the residuals of velocity, pressure, and face flux, respectively.

We choose the iterative method PIMPLE over the popular non-iterative method PISO for the following reasons:

1) The IRK schemes couple all stage values together, which breaks the non-iterative nature of the PISO algorithm. The PIMPLE algorithm, on the other hand, can be readily modified to solve the fully coupled system produced by the IRK schemes.
2) The PISO algorithm requires a small time step size (CFL<1) in order to be stable, but the PIMPLE method allows us to use a relatively large time step size (CFL>1) while still maintaining stability. With a commonly used up-to-second-order accurate temporal scheme based on finite difference, a larger time step may compromise temporal accuracy, and we address this issue by incorporating the IRK schemes that have higher-order accuracy.
3) The PIMPLE algorithm as an iterative method provides a clearly defined residual function at each time step for the purpose of adjoint formulation. The PISO algorithm, on the other hand, is non-iterative and needs to be treated as explicit forward marching in order to derive its corresponding adjoint solver. This means that all intermediate calculations during the PISO loops need to be viewed as state variables, which exacerbates the already high computational cost and makes the file I/O potentially prohibitive.

In this work, for the benchmark flow simulation that uses the second-order backward scheme, we run the PIMPLE correctors to converge the flow residuals by 6 orders of magnitude or a maximum of 20 PIMPLE corrector loops.

B. Unsteady scalar transport equation

The governing equation for the scalar transport equation is as follows.

$$\frac{\partial T}{\partial t} + (\boldsymbol{U} \cdot \nabla)T - \alpha \nabla^2 T = 0, \tag{11}$$

where T is the temperature, t is the time, U is the velocity, and α is the thermal diffusivity. The convective term can be written as:

$$(\boldsymbol{U} \cdot \nabla)T = \nabla \cdot (\phi, T), \tag{12}$$

where ϕ is the face flux and is fixed during the simulation.

Without loss of generality, we express the fully discretized scalar transport equation with the first-order Euler scheme:

$$\frac{\boldsymbol{T}^n - \boldsymbol{T}^{n-1}}{\Delta t} + \boldsymbol{A}_T \boldsymbol{T}^n = \boldsymbol{b}_T, \tag{13}$$

where the left-hand side $\boldsymbol{A}_T$ and the right-hand side $\boldsymbol{b}_T$ arise from the spatial discretization and boundary conditions. Then, Eq. (13) can be solved with any iterative linear solver for each time step.

C. Implicit Runge-Kutta scheme as a collocation method

We now derive the Gauss implicit Runge-Kutta scheme for a general semi-discrete problem from the perspective of the collocation method:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \overline{\boldsymbol{R}}(\boldsymbol{u}) = 0, \tag{14}$$

where $\boldsymbol{u}$ is the discretized state vector.

We discretize the time domain $[0, T]$ using K time steps as $0 = t_0^1 < t_0^2 < \ldots < t_0^K < t_0^{K+1} = T$. For the i-th time step, its size is $\Delta t^i = t_0^{i+1} - t_0^i$. We transform linearly $t \in [t_0^i, t_0^{i+1}]$ to $\eta \in [-1, 1]$ using:

$$t = \frac{t_0^{i+1} - t_0^i}{2}\eta + \frac{t_0^{i+1} + t_0^i}{2}. \tag{15}$$

We denote η_1, η_2, ... η_N as the roots of the N-th order Legendre polynomial lying in $[-1, 1]$, and their corresponding collocation points in $[t_0^i, t_0^{i+1}]$ determined through Eq. (15) are t_1^i, t_2^i, ... t_N^i. We denote $\boldsymbol{u}_n^i = \boldsymbol{u}(t_n^i)$, $1 \le n \le N$; they are the collocation states in the i-th time step, also known as the stage values from the Runge-Kutta perspective.

Within the i-th time step, we can then approximate the true solution $\boldsymbol{u}$ as a Lagrangian polynomial interpolated at t_0^i, t_1^i, ..., t_N^i, i.e. the starting point and the N collocation points of the local time step. That is:

$$\boldsymbol{u}^i(\eta) = \sum_{n=0}^{N} \boldsymbol{u}_n^i L_n(\eta), \tag{16}$$

where L_0, L_1, ... L_N are the Lagrangian basis:

$$L_n(\eta) = \prod_{k \neq n} \frac{(\eta - \eta_k)}{(\eta_n - \eta_k)}, 0 \le n \le N. \tag{17}$$

Due to the chain rule $\mathrm{d}\boldsymbol{u}/\mathrm{d}t = \mathrm{d}\boldsymbol{u}/\mathrm{d}\eta \cdot \mathrm{d}\eta/\mathrm{d}t$, the approximated time derivative of $\boldsymbol{u}$ is:

$$\frac{\mathrm{d}\boldsymbol{u}^i}{\mathrm{d}t} = \frac{2}{\Delta t^i} \sum_{n=0}^{N} \boldsymbol{u}_n^i L_n'(\eta). \tag{18}$$

With Eq. (18) we enforce Eq. (14) at the collocation points:

$$\boldsymbol{R}_j^i(\boldsymbol{u}_0^i, \boldsymbol{u}_1^i, ..., \boldsymbol{u}_N^i, \boldsymbol{x}) = \frac{2}{\Delta t^i} \sum_{n=0}^{N} \boldsymbol{u}_n^i L_n'(\eta_j) + \overline{\boldsymbol{R}}_j^i = 0, 1 \le j \le N, \tag{19}$$

where $\overline{\boldsymbol{R}}_j^i = \overline{\boldsymbol{R}}(\boldsymbol{u}_j^i, \boldsymbol{x})$. After we solve for $\boldsymbol{u}_1^i$, $\boldsymbol{u}_2^i$, ... $\boldsymbol{u}_N^i$ in Eq. (19), we numerically integrate Eq. (14) over $t \in [t_0^i, t_0^{i+1}]$ using Gauss Quadrature and get:

$$\boldsymbol{u}_0^{i+1} = \boldsymbol{u}_0^i - \frac{\Delta t^i}{2} \sum_{n=1}^{N} \omega_n \overline{\boldsymbol{R}}_n^i, \tag{20}$$

where ω_n, $1 \le n \le N$, are the weights for Gauss Quadrature. We repeatedly apply Eq. (19) and Eq. (20) to solve for the whole time domain.

In the current work, we use the second-order Legendre-Gauss collocation method as the temporal scheme. That is, we choose $N = 2$ in Eq. (19) and Eq. (20):

$$\begin{aligned}
\boldsymbol{R}_1^i &= \frac{2}{\Delta t^i}\left(D_{10}\boldsymbol{u}_0^i + D_{11}\boldsymbol{u}_1^i + D_{12}\boldsymbol{u}_2^i\right) + \overline{\boldsymbol{R}}_1^i = 0, \\
\boldsymbol{R}_2^i &= \frac{2}{\Delta t^i}\left(D_{20}\boldsymbol{u}_0^i + D_{21}\boldsymbol{u}_1^i + D_{22}\boldsymbol{u}_2^i\right) + \overline{\boldsymbol{R}}_2^i = 0, \\
\boldsymbol{u}_0^{i+1} &= \boldsymbol{u}_0^i - \frac{\Delta t^i}{2}\left(\overline{\boldsymbol{R}}_1^i + \overline{\boldsymbol{R}}_2^i\right),
\end{aligned} \tag{21}$$

where the coefficient matrix $\boldsymbol{D}$ is:

$$\boldsymbol{D} = \begin{bmatrix} -\sqrt{3} & 1.5 & -1.5 + \sqrt{3} \\ \sqrt{3} & -1.5 - \sqrt{3} & 1.5 \end{bmatrix}. \tag{22}$$

Note that the two-stage Gauss implicit Runge-Kutta scheme for Eq. (14) is:

$$\begin{aligned}
\boldsymbol{u}_1^i &= \boldsymbol{u}_0^i - \Delta t^i \left(A_{11} \overline{\boldsymbol{R}}_1^i + A_{12} \overline{\boldsymbol{R}}_2^i \right), \\
\boldsymbol{u}_2^i &= \boldsymbol{u}_0^i - \Delta t^i \left(A_{21} \overline{\boldsymbol{R}}_1^i + A_{22} \overline{\boldsymbol{R}}_2^i \right), \\
\boldsymbol{u}_0^{i+1} &= \boldsymbol{u}_0^i - \frac{\Delta t^i}{2} \left(\overline{\boldsymbol{R}}_1^i + \overline{\boldsymbol{R}}_2^i \right),
\end{aligned} \tag{23}$$

where the coefficient matrix $\boldsymbol{A}$ is:

$$\boldsymbol{A} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \end{bmatrix}. \tag{24}$$

By rearranging terms, we can see that the second-order Legendre-Gauss collocation method in Eq. (21) is equivalent to the two-stage Gauss implicit Runge-Kutta scheme in Eq. (23), and therefore it is *fourth* order accurate.

Note that when implemented for a segregated incompressible solver, such as the PIMPLE algorithm in this study, Eq. (21) is only applicable to the velocity equations. Other state variables, namely pressure and face flux, do not have time derivative terms in their respective equations, and they are coupled with velocity at both stages. We will elaborate on how to solve such a fully coupled system in the next subsection.

D. Iterative methods for fully coupled implicit Runge-Kutta schemes

We propose the use of block Gauss-Seidel sweeps with under-relaxation to solve the fully coupled system in Eq. (21).

For the scalar transport problem, the fully coupled system for each time step takes the block matrix form:

$$\begin{bmatrix} \frac{2D_{11}}{\Delta t}\boldsymbol{I} + \boldsymbol{A}_{T_1} & \frac{2D_{12}}{\Delta t}\boldsymbol{I} \\ \frac{2D_{21}}{\Delta t}\boldsymbol{I} & \frac{2D_{22}}{\Delta t}\boldsymbol{I} + \boldsymbol{A}_{T_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{T}_1 \\ \boldsymbol{T}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{T_1} - \frac{2D_{10}}{\Delta t}\boldsymbol{T}_0 \\ \boldsymbol{b}_{T_2} - \frac{2D_{20}}{\Delta t}\boldsymbol{T}_0 \end{bmatrix}, \tag{25}$$

where $\boldsymbol{T}_0$ is the old time step value, $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ are the two-stage values, and $\boldsymbol{I}$ is the identity matrix. The left-hand side $\boldsymbol{A}_{T_1}$ and $\boldsymbol{A}_{T_2}$ as well as the right-hand side $\boldsymbol{b}_{T_1}$ and $\boldsymbol{b}_{T_2}$ arise from the spatial discretization and boundary conditions. Note that Eq. (25) is simply Eq. (13) modified with the two-stage Gauss scheme equivalent in Eq. (21). We iteratively solve the fully coupled system in Eq. (25) with block Gauss-Seidel sweeps with appropriate under-relaxation.

For the IRK-PIMPLE algorithm, we perform block Gauss-Seidel sweeps on the PIMPLE loops of the two-stage values with appropriate under-relaxation:

1) We first solve the modified intermediate velocity equation for the first stage according to Eq. (21):

$$\frac{2D_{11}}{\Delta t}\boldsymbol{U}_1^* + \boldsymbol{A}_{U_1}\boldsymbol{U}_1^* = \boldsymbol{b}_{U_1} - \frac{2D_{10}}{\Delta t}\boldsymbol{U}_0 - \frac{2D_{12}}{\Delta t}\boldsymbol{U}_2, \tag{26}$$

   where $\boldsymbol{U}_0$ is the old time step velocity, $\boldsymbol{U}_1^*$ is the intermediate velocity for the first stage, and $\boldsymbol{U}_2$ is the velocity for the second stage. The left-hand side $\boldsymbol{A}_{U_1}$ and the right-hand side $\boldsymbol{b}_{U_1}$ arise from the spatial discretization and boundary conditions.
2) We solve for the updated pressure field, correct the face flux, and correct the velocity field according to Eqs. (4) to (10) (PISO corrector loop) for the first stage. We repeat the PISO corrector loop two times to obtain the first stage value $\boldsymbol{U}_1$, $\boldsymbol{p}_1$ and $\boldsymbol{\phi}_1$.
3) We then solve the modified intermediate velocity equation for the second stage according to Eq. (21):

$$\frac{2D_{22}}{\Delta t}\boldsymbol{U}_2^* + \boldsymbol{A}_{U_2}\boldsymbol{U}_2^* = \boldsymbol{b}_{U_2} - \frac{2D_{20}}{\Delta t}\boldsymbol{U}_0 - \frac{2D_{21}}{\Delta t}\boldsymbol{U}_1, \tag{27}$$

   where $\boldsymbol{U}_2^*$ is the intermediate velocity for the second stage. Similarly, $\boldsymbol{A}_{U_2}$ and $\boldsymbol{b}_{U_2}$ arise from the spatial discretization and boundary conditions for the second stage.

4) We repeat the PISO corrector loop two times for the second stage and obtain $\boldsymbol{U}_2$, $\boldsymbol{p}_2$ and $\boldsymbol{\phi}_2$.

5) We repeat all steps above until the desired level of convergence is reached for both stages.

In this work, for both the IRK scalar transport simulation and the IRK-PIMPLE flow simulation, we perform 10 block Gauss-Seidel sweeps with all variables under-relaxed by 0.8. The residuals at both stages are converged by 4 to 6 orders of magnitude.

E. Gauss IRK-adjoint formulation

As discussed previously, the IRK-PIMPLE primal solver with the two-stage Gauss scheme converges the fully coupled system of the stage values for each time step, then it updates the velocity field of the new time step explicitly. Therefore, the time-marching primal solution process can be viewed as:

$$\boldsymbol{R}(\boldsymbol{w},\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{R}^1(\boldsymbol{w}^1, \boldsymbol{U}_0^1, \boldsymbol{x}) \\ \boldsymbol{R}^2(\boldsymbol{w}^2, \boldsymbol{U}_0^2, \boldsymbol{x}) \\ \vdots \\ \boldsymbol{R}^K(\boldsymbol{w}^K, \boldsymbol{U}_0^K, \boldsymbol{x}) \end{bmatrix} = \boldsymbol{0}, \tag{28}$$

$$\boldsymbol{U}_0^{i+1} = \boldsymbol{U}_0^i - \frac{\Delta t^i}{2}(\omega_1 \overline{\boldsymbol{R}_{U1}}^i + \omega_2 \overline{\boldsymbol{R}_{U2}}^i), 1 \le i \le K, \tag{29}$$

where a superscript denotes the time step index with K being the total number of time steps, a subscript denotes the stage index with 0 indicting the old time step, and $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is the design variable vector. For each time step, $\boldsymbol{w}^i \in \mathbb{R}^{2n_w}$ consists of velocity, pressure, and face flux for both stages; $\boldsymbol{U}_0^i$ is the old time step velocity, and $\boldsymbol{R}^i \in \mathbb{R}^{2n_w}$ is the flow residual vector of the fully coupled system for the two-stage values. $\overline{\boldsymbol{R}_U}$ is the non-time-derivative portion of the velocity residual function. The weights are $\omega_1 = \omega_2 = 1$ for the two-stage Gauss scheme. Note that $\boldsymbol{U}_0^1, \boldsymbol{U}_0^2, ..., \boldsymbol{U}_0^K$ are not state variables. $\boldsymbol{U}_0^1$ is a part of the initial condition; $\boldsymbol{U}_0^2, \boldsymbol{U}_0^3, ..., \boldsymbol{U}_0^K$ are determined explicitly through Eq. (29) and they are not unknowns solved in Eq. (28).

The objective function F depends on both the design variables $\boldsymbol{x}$ and the state variable vector $\boldsymbol{w}$ solved through Eq. (28) and Eq. (29), and in many applications, including this study, the objective function F is derived from a time-averaged value with respect to continuous time. Then it can be expressed as the weighted summation of a time series through Gauss quadrature:

$$\begin{aligned} F(\boldsymbol{w},\boldsymbol{x}) &= \frac{1}{T}\int_0^T f\ dt \\ &= \frac{1}{T}\sum_{i=1}^{K}\int_{t_0^i}^{t_0^{i+1}} f\ dt \\ &\approx \sum_{i=1}^{K} \underbrace{\frac{\Delta t^i}{2T}\left(\omega_1 f_1^i(\boldsymbol{w}_1^i, \boldsymbol{x}) + \omega_2 f_2^i(\boldsymbol{w}_2^i, \boldsymbol{x})\right)}_{f^i(\boldsymbol{w}^i,\boldsymbol{x})}, \end{aligned} \tag{30}$$

Then, the partial derivative $\partial F/\partial \boldsymbol{w}$ can be simplified as:

$$\underbrace{\frac{\partial F}{\partial \boldsymbol{w}}}_{1\times 2Kn_w} = [\underbrace{\frac{\partial f^1}{\partial \boldsymbol{w}^1}}_{1\times 2n_w}, \underbrace{\frac{\partial f^2}{\partial \boldsymbol{w}^2}}_{1\times 2n_w}, \cdots, \underbrace{\frac{\partial f^K}{\partial \boldsymbol{w}^K}}_{1\times 2n_w}], \tag{31}$$

and for each time step:

$$\underbrace{\frac{\partial f^i}{\partial \boldsymbol{w}^i}}_{1\times 2n_w} = \frac{\Delta t^i}{2T}\,[\omega_1 \underbrace{\frac{\partial f_1^i}{\partial \boldsymbol{w}_1^i}}_{1\times n_w}, \omega_2 \underbrace{\frac{\partial f_2^i}{\partial \boldsymbol{w}_2^i}}_{1\times n_w}], \ 1 \le i \le K. \tag{32}$$

To obtain the total derivative $\mathrm{d}F/\mathrm{d}\boldsymbol{x}$ for gradient-based optimization, we apply the chain rule as follows:

$$\underbrace{\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}}_{1\times n_x} = \underbrace{\frac{\partial F}{\partial \boldsymbol{x}}}_{1\times n_x} + \underbrace{\frac{\partial F}{\partial \boldsymbol{w}}}_{1\times 2Kn_w} \underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}}}_{2Kn_w\times n_x}, \tag{33}$$

where the partial derivatives $\partial F/\partial \boldsymbol{x}$ and $\partial F/\partial \boldsymbol{w}$ are relatively cheap to evaluate because they only involve explicit computations. The total derivative $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{x}$ matrix, on the other hand, is expensive, because $\boldsymbol{w}$ and $\boldsymbol{x}$ are implicitly linked by the residual equations $\boldsymbol{R}(\boldsymbol{w}, \boldsymbol{x}) = 0$.

To solve for $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{x}$, we can apply the chain rule for $\boldsymbol{R}$. We then use the fact that the governing equations should always hold, independent of the values of design variables $\boldsymbol{x}$. Therefore, the total derivative $\mathrm{d}\boldsymbol{R}/\mathrm{d}\boldsymbol{x}$ must be zero:

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{x}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}} = 0. \tag{34}$$

Rearranging the above equation, we get the linear system

$$\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}}_{2Kn_w\times 2Kn_w} \cdot \underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}}}_{2Kn_w\times n_x} = - \underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}}_{2Kn_w\times n_x}. \tag{35}$$

We can then substitute the solution for $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{x}$ from Eq. (35) into Eq. (33) to get

$$\underbrace{\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}}_{1\times n_x} = \underbrace{\frac{\partial F}{\partial \boldsymbol{x}}}_{1\times n_x} - \overbrace{\underbrace{\frac{\partial F}{\partial \boldsymbol{w}}}_{1\times 2Kn_w} \underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}^{-1}}_{2Kn_w\times 2Kn_w}}^{\boldsymbol{\psi}^T} \underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}}_{2Kn_w\times n_x}. \tag{36}$$

Now we can transpose the Jacobian and solve with $[\partial F/\partial \boldsymbol{w}]^T$ as the right-hand side, which yields the *adjoint equation*,

$$\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}^T}_{2Kn_w\times 2Kn_w} \cdot \underbrace{\boldsymbol{\psi}}_{2Kn_w\times 1} = \underbrace{\frac{\partial F}{\partial \boldsymbol{w}}^T}_{2Kn_w\times 1}, \tag{37}$$

where $\boldsymbol{\psi}$ is the *adjoint vector*. Then, we can compute the total derivative by substituting the adjoint vector into Eq. (36):

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}} = \frac{\partial F}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}. \tag{38}$$

Since the design variables are not explicitly present in Eq. (37), we need to solve the adjoint equation only once for each objective function. Therefore, the computational cost is independent of the number of design variables but proportional to the number of objective functions. This approach of computing derivatives introduced so far is also known as the *adjoint method*. It is advantageous for optimization problems in Aerospace Engineering because typically, there is only one objective function, but hundreds or thousands of design variables may be used.

The adjoint equation in Eq. (37) can be expressed in a block-structured matter with respect to the time segments:

$$\begin{bmatrix} \frac{\partial \boldsymbol{R}^1}{\partial \boldsymbol{w}^1}^T & \frac{\partial \boldsymbol{R}^2}{\partial \boldsymbol{w}^1}^T & \frac{\partial \boldsymbol{R}^3}{\partial \boldsymbol{w}^1}^T & \cdots & \frac{\partial \boldsymbol{R}^K}{\partial \boldsymbol{w}^1}^T \\ & \frac{\partial \boldsymbol{R}^2}{\partial \boldsymbol{w}^2}^T & \frac{\partial \boldsymbol{R}^3}{\partial \boldsymbol{w}^2}^T & \cdots & \frac{\partial \boldsymbol{R}^K}{\partial \boldsymbol{w}^2}^T \\ & & \ddots & \ddots & \vdots \\ & & & \frac{\partial \boldsymbol{R}^{K-1}}{\partial \boldsymbol{w}^{K-1}}^T & \frac{\partial \boldsymbol{R}^K}{\partial \boldsymbol{w}^{K-1}}^T \\ & & & & \frac{\partial \boldsymbol{R}^K}{\partial \boldsymbol{w}^K}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^1 \\ \boldsymbol{\psi}^2 \\ \vdots \\ \boldsymbol{\psi}^{K-1} \\ \boldsymbol{\psi}^K \end{bmatrix} = \begin{bmatrix} \frac{\partial f^1}{\partial \boldsymbol{w}^1}^T \\ \frac{\partial f^2}{\partial \boldsymbol{w}^2}^T \\ \vdots \\ \frac{\partial f^{K-1}}{\partial \boldsymbol{w}^{K-1}}^T \\ \frac{\partial f^K}{\partial \boldsymbol{w}^K}^T \end{bmatrix}, \tag{39}$$

where the adjoint vector $\boldsymbol{\psi} \in \mathbb{R}^{2Kn_w}$ is broken down into K parts that correspond to the time segments, i.e., $\boldsymbol{\psi}^1, \boldsymbol{\psi}^2, \ldots, \boldsymbol{\psi}^K \in \mathbb{R}^{2n_w}$. For each $1 \le i \le K$, $\boldsymbol{\psi}^i$ consists of $\boldsymbol{\psi}_1^i$ and $\boldsymbol{\psi}_2^i \in \mathbb{R}^{n_w}$ which correspond to the two stages of the time step. The left-hand side of Eq. (39) has an upper-triangular block structure because the residual function for any time step has no dependency on future states. As indicated in Eq. (28), for any $2 \le i \le K$, $\boldsymbol{R}^i$ explicitly depends on a state variable $\boldsymbol{w}^i$ and a non-state intermediate variable $\boldsymbol{U}_0^i$, and through Eq. (29), $\boldsymbol{R}^i$ also has explicit dependency on $\boldsymbol{w}^1, \boldsymbol{w}^2, ..., \boldsymbol{w}^{i-1}$. Therefore, the upper off-diagonal blocks on the left-hand side of Eq. (39) are always non-zero. Then, Eq. (39) can be solved sequentially in a backward fashion as:

$$\begin{aligned}
&\frac{\partial \boldsymbol{R}^{K\,T}}{\partial \boldsymbol{w}^K} \cdot \boldsymbol{\psi}^K = \frac{\partial f^{K\,T}}{\partial \boldsymbol{w}^K}, \\
&\frac{\partial \boldsymbol{R}^{i\,T}}{\partial \boldsymbol{w}^i} \cdot \boldsymbol{\psi}^i = \frac{\partial f^{i\,T}}{\partial \boldsymbol{w}^i} \underbrace{- \sum_{m=i+1}^{K} \frac{\partial \boldsymbol{R}^{m\,T}}{\partial \boldsymbol{w}^i} \cdot \boldsymbol{\psi}^m}_{\boldsymbol{S}^i}, \quad K-1 \ge i \ge 1,
\end{aligned} \tag{40}$$

where the right-hand side $[\partial f^i / \partial \boldsymbol{w}^i]^T$ expressed in Eq. (32) can be efficiently evaluated with reverse-mode automatic differentiation (AD), and the underbraced term $\boldsymbol{S}^i$ requires further simplification so that it can computed efficiently.

We now simplify the above right-hand side term $\boldsymbol{S}^i$ in Eq. (40). We first deploy the chain rule:

$$\frac{\partial \boldsymbol{R}^m}{\partial \boldsymbol{w}^i} = \frac{\partial \boldsymbol{R}^m}{\partial \boldsymbol{U}_0^m} \frac{\partial \boldsymbol{U}_0^m}{\partial \boldsymbol{w}^i}, \quad 1 \le i < m \le K. \tag{41}$$

Therefore, the expression for $\boldsymbol{S}^i$ becomes:

$$\boldsymbol{S}^i = - \sum_{m=i+1}^{K} \frac{\partial \boldsymbol{U}_0^{m\,T}}{\partial \boldsymbol{w}^i} \cdot \left(\frac{\partial \boldsymbol{R}^{m\,T}}{\partial \boldsymbol{U}_0^m} \cdot \boldsymbol{\psi}^m \right), \quad K-1 \ge i \ge 1. \tag{42}$$

We then sum Eq. (29) over the index i and get:

$$\boldsymbol{U}_0^m = \boldsymbol{U}_0^1 - \sum_{i=1}^{m-1} \frac{\Delta t^i}{2} (\omega_1 \overline{\boldsymbol{R}_{U1}}^i + \omega_2 \overline{\boldsymbol{R}_{U2}}^i), \quad 2 \le m \le K, \tag{43}$$

which leads to:

$$\frac{\partial \boldsymbol{U}_0^m}{\partial \boldsymbol{w}^i} = \begin{bmatrix} \dfrac{\partial \boldsymbol{U}_0^m}{\partial \boldsymbol{w}_1^i} \\ \\ \dfrac{\partial \boldsymbol{U}_0^m}{\partial \boldsymbol{w}_2^i} \end{bmatrix} = -\frac{\Delta t^i}{2} \begin{bmatrix} \omega_1 \dfrac{\partial \overline{\boldsymbol{R}_{U1}}^i}{\partial \boldsymbol{w}_1^i} \\ \\ \omega_2 \dfrac{\partial \overline{\boldsymbol{R}_{U2}}^i}{\partial \boldsymbol{w}_2^i} \end{bmatrix}, \quad 1 \le i < m \le K, \tag{44}$$

where the expression for $\partial \boldsymbol{U}_0^m / \partial \boldsymbol{w}^i$ no longer involves the index m. Therefore, the expression for $\boldsymbol{S}^i$ becomes:

$$\boldsymbol{S}^i = \frac{\Delta t^i}{2} \begin{bmatrix} \omega_1 \dfrac{\partial \overline{\boldsymbol{R}_{U1}}^{i\,T}}{\partial \boldsymbol{w}_1^i} \\ \\ \omega_2 \dfrac{\partial \overline{\boldsymbol{R}_{U2}}^{i\,T}}{\partial \boldsymbol{w}_2^i} \end{bmatrix} \cdot \underbrace{\left(\sum_{m=i+1}^{K} \frac{\partial \boldsymbol{R}^{m\,T}}{\partial \boldsymbol{U}_0^m} \cdot \boldsymbol{\psi}^m \right)}_{\xi^i}, \quad K-1 \ge i \ge 1. \tag{45}$$

$\boldsymbol{\xi}^i$ in Eq. (45) can be calculated accumulatively as:

$$\begin{aligned}
\boldsymbol{\xi}^{K-1} &= \frac{\partial \boldsymbol{R}_1^K}{\partial \boldsymbol{U}_0^K}^T \cdot \boldsymbol{\psi}_1^K + \frac{\partial \boldsymbol{R}_2^K}{\partial \boldsymbol{U}_0^K}^T \cdot \boldsymbol{\psi}_2^K, \\
\boldsymbol{\xi}^{i-1} &= \boldsymbol{\xi}^i + \left(\frac{\partial \boldsymbol{R}_1^i}{\partial \boldsymbol{U}_0^i}^T \cdot \boldsymbol{\psi}_1^i + \frac{\partial \boldsymbol{R}_2^i}{\partial \boldsymbol{U}_0^i}^T \cdot \boldsymbol{\psi}_2^i \right), \; K-1 \geq i \geq 1,
\end{aligned} \tag{46}$$

and then $\boldsymbol{S}^i$ is calculated as:

$$\boldsymbol{S}^i = \frac{\Delta t^i}{2} \begin{bmatrix} \omega_1 \dfrac{\partial \overline{\boldsymbol{R}_{U1}}^i}{\partial \boldsymbol{w}_1^i}^T \cdot \boldsymbol{\xi}^i \\ \\ \omega_2 \dfrac{\partial \overline{\boldsymbol{R}_{U2}}^i}{\partial \boldsymbol{w}_2^i}^T \cdot \boldsymbol{\xi}^i \end{bmatrix}, \; K-1 \geq i \geq 1. \tag{47}$$

Note that the right-hand side matrix-transpose-vector products in Eq. (46) and Eq. (47) can also be efficiently evaluated with reverse-mode AD, and each evaluation only involves the states and residuals of one *local* time step.

As indicated in Eq. (30), the objective function F is of the summation type. Therefore, the total derivative $\mathrm{d}F/\mathrm{d}\boldsymbol{x}$ in Eq. (38) also becomes a summation of a time series:

$$\begin{aligned}
\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}} &= \sum_{i=1}^{K} \left(\frac{\partial f^i}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^{iT} \left(\frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}_0^i} \frac{\partial \boldsymbol{U}_0^i}{\partial \boldsymbol{x}} \right) \right) \\
&= \sum_{i=1}^{K} \left(\frac{\partial f^i}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^{iT} \frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{x}} \right) \underbrace{- \sum_{i=1}^{K} \boldsymbol{\psi}^{iT} \frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}_0^i} \frac{\partial \boldsymbol{U}_0^i}{\partial \boldsymbol{x}}}_{\boldsymbol{M}^T}.
\end{aligned} \tag{48}$$

Note that $\boldsymbol{R}^i$ depends on the intermediate variable $\boldsymbol{U}_0^i$, which leads to the underbraced term $\boldsymbol{M}^T$ in Eq. (48) that requires further simplification. With Eq. (43) $\boldsymbol{M}$ becomes:

$$\begin{aligned}
\boldsymbol{M} &= -\sum_{i=1}^{K} \frac{\partial \boldsymbol{U}_0^1}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}_0^i}^T \boldsymbol{\psi}^i + \sum_{i=2}^{K} \sum_{m=1}^{i-1} \frac{\Delta t^m}{2} \left(\omega_1 \frac{\partial \overline{\boldsymbol{R}_{U1}}^m}{\partial \boldsymbol{x}}^T + \omega_2 \frac{\partial \overline{\boldsymbol{R}_{U2}}^m}{\partial \boldsymbol{x}}^T \right) \frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}_0^i}^T \boldsymbol{\psi}^i \\
&= -\frac{\partial \boldsymbol{U}_0^1}{\partial \boldsymbol{x}}^T \sum_{i=1}^{K} \frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}_0^i}^T \boldsymbol{\psi}^i + \sum_{i=1}^{K-1} \frac{\Delta t^i}{2} \left(\omega_1 \frac{\partial \overline{\boldsymbol{R}_{U1}}^i}{\partial \boldsymbol{x}}^T + \omega_2 \frac{\partial \overline{\boldsymbol{R}_{U2}}^i}{\partial \boldsymbol{x}}^T \right) \left(\sum_{m=i+1}^{K} \frac{\partial \boldsymbol{R}^m}{\partial \boldsymbol{U}_0^m}^T \boldsymbol{\psi}^m \right) \\
&= -\frac{\partial \boldsymbol{U}_0^1}{\partial \boldsymbol{x}}^T \boldsymbol{\xi}^0 + \sum_{i=1}^{K-1} \frac{\Delta t^i}{2} \left(\omega_1 \frac{\partial \overline{\boldsymbol{R}_{U1}}^i}{\partial \boldsymbol{x}}^T + \omega_2 \frac{\partial \overline{\boldsymbol{R}_{U2}}^i}{\partial \boldsymbol{x}}^T \right) \boldsymbol{\xi}^i.
\end{aligned} \tag{49}$$

For most design optimization problems, the initial flow field has no dependency on the design variables, and the term involving $\partial \boldsymbol{U}_0^1 / \partial \boldsymbol{x}$ in Eq. (49) can often be omitted. Eq. (48) and Eq. (49) indicate that the total derivative $\mathrm{d}F/\mathrm{d}\boldsymbol{x}$ can be efficiently calculated on the fly as we sequentially solve the adjoint equations, and each required reverse-mode AD tape only involves one local time step. This is desirable because we do not need to access all state variables in the memory, nor do we need to save the adjoint vectors for all time steps to the disk.

Note that the Gauss IRK-adjoint formulation derived here is also compatible with a fully coupled primal solver. For example, for the scalar transport problem considered in this study, we can simply replace the state variables with $\boldsymbol{T}_1^i$ and $\boldsymbol{T}_2^i$, and the intermediate variables with $\boldsymbol{T}_0^i$. Some partial derivatives can also be reduced to constant coefficients.

In this work, we solve the fully coupled IRK-adjoint equation at each step through block Gauss-Seidel sweeps with under-relaxation, which is similar to that of the IRK primal solution process. The Gauss IRK-adjoint solver

is implemented for the scalar transport problem and we are working on its implementation for the IRK-PIMPLE method.

III. Results

In this section, we first use the two-stage Gauss IRK-PIMPLE solver to simulate unsteady laminar flow over a ramp. We compare the resulting flow field to that obtained from the standard PIMPLE solver with the second-order backward scheme. Then, we evaluate the adjoint gradient accuracy for the proposed Gauss IRK-adjoint formulation using the scalar transport problem.

A. IRK-PIMPLE simulation for unsteady laminar flow over a 45-degree ramp

As mentioned above, this subsection considers unsteady laminar flow over a 45-degree ramp. The flow is from left to right. The incoming flow velocity is 5 m/s at the inlet. The channel height at the outlet is 1 m, and the channel length is 3 m. The viscosity is 10^{-3} and the Reynolds number based on the channel height at the outlet is 5000. We generated a structured mesh with 3000 cells and the average y^+ is about 1.3. We first develop the initial flow field by marching from a uniform flow field of 5 m/s for 0.02 s with the standard PIMPLE solver and the second-order backward scheme. We use this spun-up flow field as the initial condition for the flow simulations in this study.

We then use the IRK-PIMPLE solver with the two-stage Gauss scheme to simulate the unsteady flow up to $t = 1$ s and verify the resulting flow field against that obtained from the standard PIMPLE solver and the second-order backward scheme. For the two-stage Gauss scheme, the time step size is 0.02 s and the corresponding CFL number is about 7. For the second-order backward scheme, the time step size is 0.002 s and the corresponding CFL number is about 0.6. We normalize the velocity field error by the incoming flow velocity 5 m/s. At $t = 1$ s, the infinity-norm error for the IRK-PIMPLE velocity field is about 3.3%, and the normalized L2-norm error is about 0.3%. As shown in Figure 1, the resulting velocity fields are visually identical. Therefore, we can conclude that the proposed IRK-PIMPLE is working as intended.

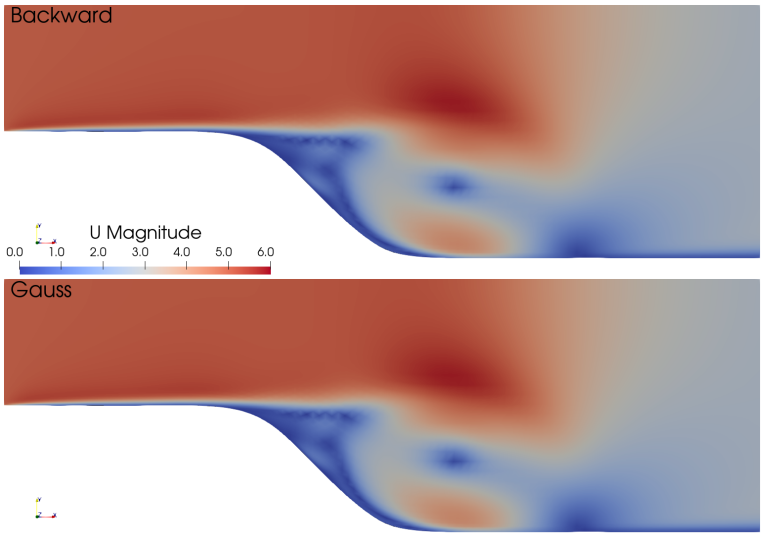

Fig. 1 Velocity field comparison for an 45-degree ramp geometry at $t = 1$ s. Top: standard PIMPLE with the second-order backward scheme; bottom: IRK-PIMPLE with the two-stage Gauss scheme. The resulting flow fields are visually identical.

B. IRK-adjoint gradient accuracy evaluation with the unsteady scalar transport problem

We verify the Gauss IRK-adjoint formulation with the scalar transport problem using a channel flow problem. The flow goes from left to right. The temperature is 1.0 at the channel inlet, 1.2 at the upper wall, and 0.3 at the lower wall. The velocity field is prescribed and not coupled with temperature. We first develop the initial temperature field by marching from a uniform all-zero field for 10^{-4} s with the standard scalar transport solver and the second-order backward scheme. We use this spun-up temperature field as the initial condition for the scalar transport simulations in this study.

We run the Gauss IRK scalar transport solver up to $t = 0.5$ s and verify the temperature field against that obtained from the standard scalar transport solver and the second-order backward scheme. The time step size is 10^{-3} s for the two-stage Gauss scheme, and 2×10^{-6} for the backward scheme. At $t = 0.5$ s, the infinity-norm error for the IRK-PIMPLE velocity field is about 2.8×10^{-5}, and the normalized L2-norm error is about 5.2×10^{-6}. As shown in Figure 2, the resulting temperature fields are visually identical. Therefore, we can conclude that the proposed IRK scalar transport primal solver is working as intended.

We evaluate the adjoint gradient using a one-time-step setup. The objective function is the time-averaged cell-wise mean temperature of the whole domain. The design variable is the inlet temperature. The reference value of the derivative obtained through central difference is 8.086×10^{-4}, and the derivative calculated by the proposed Gauss IRK-adjoint method is 8.170×10^{-4}. The relative error is about 1%, and we are working on improving the derivative accuracy.

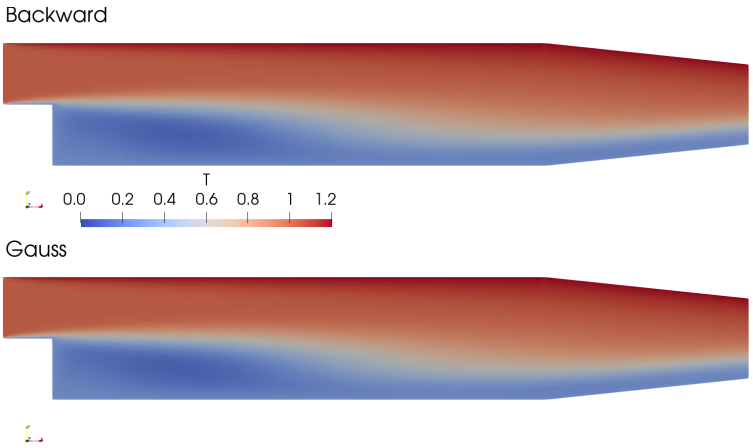

Fig. 2 Temperature field comparison for the channel flow scalar transport problem at $t = 0.5$ s. Top: standard scalar transport solver with the second-order backward scheme; bottom: IRK scalar transport solver with the two-stage Gauss scheme. The resulting temperature fields are visually identical.

Table 1 Verification of the adjoint gradient accuracy for the scalar transport problem. The derivative values are in units of 10^{-4}.

	Adjoint	FD-Ref	Error
$d\overline{T}/dU_{\text{in}}$	8.170	8.086	1.0%

IV. Conclusion

In this paper, we propose the higher-order-accurate IRK-PIMPLE method for segregated flow solvers. We modify the standard PIMPLE algorithm and perform block Gauss-Seidel sweeps with under-relaxation to solve the coupled systems produced by the Gauss IRK scheme. We derive the Gauss IRK-adjoint formulation that

is compatible with a segregated flow solver such as the proposed IRK-PIMPLE. We validate the proposed IRK-PIMPLE method by simulating unsteady flow over a ramp geometry and comparing the resulting flow field against the second-order backward scheme reference. We implement the Gauss IRK-adjoint for the scalar transport problem and evaluate the adjoint gradient. The adjoint derivative shows about 1% error compared with the finite difference reference. The proposed IRK-PIMPLE and its corresponding adjoint solver have the potential to significantly reduce the computation cost for time-resolved unsteady optimization using segregated flow solvers. In the future, we will implement the Gauss IRK-adjoint solver corresponding to the proposed IRK-PIMPLE and improve the adjoint accuracy. We will also investigate the Radau scheme and IRK schemes of higher-stage orders in future work.

V. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number 2223676. This work used the Stampede 3 supercomputer at Texas Advanced Computing Center through allocation ATM-140019 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

[1] Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260. https://doi.org/10.1007/BF01061285.

[2] Kenway, G. K., Mader, C. A., He, P., and Martins, J. R., "Effective adjoint approaches for computational fluid dynamics," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100542. https://doi.org/10.1016/j.paerosci.2019.05.002, publisher: Pergamon.

[3] Fang, L., and He, P., "Field inversion machine learning augmented turbulence modeling for time-accurate unsteady flow," *Physics of Fluids*, Vol. 36, No. 5, 2024.

[4] Fang, L., and He, P., "A duality-preserving adjoint method for segregated Navier–Stokes solvers," *Journal of Computational Physics*, 2024.

[5] McMullen, M., Jameson, A., and Alonso, J., "Application of a non-linear frequency domain solver to the Euler and Navier-Stokes equations," *40th AIAA aerospace sciences meeting & exhibit*, 2002, p. 120.

[6] Hall, K. C., Thomas, J. P., and Clark, W. S., "Computation of unsteady nonlinear flows in cascades using a harmonic balance technique," *AIAA journal*, Vol. 40, No. 5, 2002, pp. 879–886.

[7] He, P., Luder, A., Mader, C., Martins, J. R., and Maki, K., "A time-spectral adjoint approach for aerodynamic shape optimization under periodic wakes," *AIAA Scitech 2020 Forum*, 2020, p. 2114.

[8] He, P., and Martins, J. R. R. A., "A hybrid time-spectral approach for aerodynamic shape optimization with unsteady flow," *AIAA scitech 2021 forum*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2021. https://doi.org/10.2514/6.2021-0278.

[9] Im, D. K., Choi, S., McClure, J. E., and Skiles, F., "Mapped Chebyshev pseudospectral method for unsteady flow analysis," *AIAA Journal*, Vol. 53, No. 12, 2015, pp. 3805–3820.

[10] Prasad, R., Choi, J.-Y., Pisharoti, N., and Choi, S., "Chebyshev pseudo-spectral and Adjoint-based sensitivity analysis for Unsteady Flow Design," *2018 Multidisciplinary Analysis and Optimization Conference*, 2018, p. 2926.

[11] Jameson, A., "Evaluation of fully implicit Runge Kutta schemes for unsteady flow calculations," *Journal of Scientific Computing*, Vol. 73, No. 2, 2017, pp. 819–852.

[12] Franco, M., Persson, P.-O., Pazner, W., and Zahr, M. J., "An adjoint method using fully implicit runge-kutta schemes for optimization of flow problems," *AIAA Scitech 2019 Forum*, 2019, p. 0351.

[13] Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica*, Vol. 46, No. 11, 2010, pp. 1843–1851.

[14] Falck, R., Gray, J. S., Ponnapalli, K., and Wright, T., "dymos: A Python package for optimal control of multidisciplinary systems," *Journal of Open Source Software*, Vol. 6, No. 59, 2021, p. 2809.

[15] Rhie, C. M., and Chow, W. L., "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA Journal*, Vol. 21, No. 11, 1983, pp. 1525–1532. https://doi.org/10.2514/3.8284.