A Segragated Time-Accurate Adjoint Method for Field Inversion of Unsteady Flow

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

Machine learning (ML) is a powerful tool for computational fluid dynamics (CFD) because it can use data to correct imperfect physical models (e.g., turbulence models) and improve the simulation accuracy for challenging flow conditions, e.g., separated flow. Among many ML approaches, field inversion ML has the advantages of model consistency and low data dependency and can train generalizable models with limited data. However, the field inversion has a high entry bar because it requires a CFD-solver-intrusive implementation of adjoint methods to compute gradients. Due to this limitation, all existing field inversion ML studies focused on improving steady-state flow predictions. This paper will break this limit by developing a time-accurate adjoint method to enable the field inversion of *unsteady* flow. We solve the unsteady incompressible flow using a segregated PIMPLE algorithm and then compute unsteady gradients using a new segregated adjoint method. We use the proposed PIMPLE-adjoint to demonstrate field inversion optimization for unsteady separated flow over a ramp, which is representative of many unsteady flows in aerospace engineering. We consider various options to formulate the objective function for unsteady field inversion using a laminar flow case. We find that the inverse flow field agrees well with the reference data if the objective function formulation includes all relevant unsteady flow physics. Then, we consider a turbulent case and multiply the production term of the Spalart–Allmaras (SA) turbulence model by a scalar field (β). We run unsteady field inversion to optimize the SA model's β field and make its predictions match the spatial-temporal distribution of reference data generated by the $k-\omega$ SST model. In addition, we evaluate the field inversion's capabilities to predict temporally unseen flow fields and predict flow variables not formulated in the objective function. The spatial-temporal distributions of inverse flow fields agree reasonably well with the reference data for all the above cases. The proposed method has the potential to train generalizable and accurate turbulence models for predicting challenging unsteady flow in aerospace engineering.

I. Introduction

Computational fluid dynamics (CFD) is a powerful tool to simulate the three-dimensional flow field and facilitate the analysis and design of aerospace engineering systems. One can run CFD with various levels of fidelity, ranging from the Reynolds-averaged Navier-Stokes (RANS) approach to direct numerical simulation (DNS). The RANS approach is commonly used in practical analysis, design, and optimization because of its relatively low computational cost. However, it is known that the RANS approach has imperfect physical models (e.g., turbulence models) that degrade its simulation accuracy for challenging flow conditions [1], such as flow separation and turbulence transition.

Researchers have proposed various machine learning (ML) methods to correct the imperfect physical models for CFD solvers [2–9]. Compared with traditional human-intuition-based physical model development, the ML methods rely solely on data and can accelerate the development of accurate and generalizable physical models. Field inversion ML is a model-intrusive method proposed by Duraisamy and co-workers [10–12]. Compared with solver-non-intrusive learning methods, such as physics-informed ML [13], the field inversion ML incorporates the entire CFD solution in the training phase and can use limited data for training. Because of its intrusive nature, the training and prediction in the field inversion ML are consistent at the discretized level, which further improves its accuracy and generalizability [9]. However, one bottleneck hindering the widespread usage of field inversion ML is that it requires CFD-solver-intrusive implementations of adjoint methods to compute gradients with respect to a large number of design variables. Because of this high-entry bar, existing field inversion ML studies focused on improving the prediction accuracy for steady-state flow only [14–20].

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

However, many aerospace engineering design problems require accurate simulations of unsteady or time-dependent flow processes, such as perturbation response, vertical-to-horizontal flight transition, and dynamic stall. To our best knowledge, unsteady field inversion has not been done before. This is mainly due to the lack of an efficient and accurate time-accurate adjoint method for computing unsteady gradients. The steady-state adjoint has been widely used in aerospace engineering optimization, including aircraft [21–24], wind turbine [25, 26], and gas turbines [27–30], to name a few. However, the unsteady adjoint is known to be computationally expensive because one needs to store all intermediate variables and repeatedly solve the adjoint equations in reverse, starting from the last time step. Most of the existing unsteady adjoint methods were proposed for fully-coupled Navier–Stokes (NS) solvers working in compressible flow regimes [31–35]. For low-speed flow, segregated NS solvers are more commonly used. Unsteady continuous adjoint methods were proposed for segregated NS solvers [36]. However, the accuracy of continuous adjoint methods is known to be sensitive to the mesh density and any simplifications and assumptions in their formulations [37–39]. Therefore, continuous adjoint methods are not the most robust option for large-scale field inversion optimization. An unsteady discrete adjoint method was developed for segregated NS solvers [40]. However, the above study used an automatic differentiation (AD) tool to differentiate the entire CFD solvers (a.k.a, brute-force AD), which is computationally inefficient.

Given the above motivation, the objective of this paper is to enable field inversion of unsteady flow by developing a segregated time-accurate discrete adjoint method for low-speed flows. The central recipe is that we will use the coupled pressure-implicit with splitting of operators (PISO) and Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm (aka PIMPLE) to compute the flow. Then, we will develop a time-accurate PIMPLE-adjoint method to compute the unsteady gradients, following a similar segregated solution process as the flow solver. The benefit of using the PIMPLE-adjoint approach is that one can relax the CFL number to be greater than one, reducing the number of adjoint equations to solve. This paper will elaborate on the proposed PIMPLE-adjoint algorithm and use it to demonstrate field inversion of unsteady flow over a ramp, which is representative of many unsteady flows in aerospace engineering. First, we will use a laminar flow case to consider the proper formulation of field inversion optimization problems. We add a Brinkman term with a Gaussian impermeability field (α) to the momentum equation to change the unsteady vortex evolution over the ramp. Then, we run unsteady field inversion to find the prescribed α field distribution, starting from a zero-α field. We will consider various probe point options to construct the objective function and evaluate their performance in finding the correct α and velocity fields. Then, we consider a turbulent case and multiply the production term of the Spalart–Allmaras (SA) turbulence model by a scalar field (β). We run unsteady field inversion to optimize the SA model's β field and make its predictions match the spatial-temporal distribution of reference data generated by the $k-\omega$ SST model. In addition, we evaluate the field inversion's capabilities to predict temporally unseen flow fields and predict flow variables not formulated in the objective function.

The rest of the paper is organized as follows. In Section II, we describe the mathematical background of the proposed adjoint approach. The unsteady field inversion results are presented and discussed in Section III and we summarize our findings in Section IV.

II. Method

In this section, we derive the PIMPLE time-accurate adjoint formulation for the incompressible turbulent Navier–Stokes equations. We also evaluate the speed, memory usage, and accuracy of the proposed PIMPLE-adjoint method.

A. Time-accurate flow simulation using the PIMPLE method

First, we analyze the primal flow problem with the PIMPLE method. It solves the incompressible Navier–Stokes equations, written as follows:

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

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

where p is the pressure, ρ is the density, $\boldsymbol{U}$ is the velocity vector $\boldsymbol{U} = [u, v, w]$, $\nu_{\text{eff}} = \nu + \nu_t$ with ν and ν_t being the kinematic and turbulent eddy viscosity, respectively, and $\alpha \boldsymbol{U}$ is the Brinkman term with an impermeability field (α).

To solve the continuity and momentum equations (1) and (2), the momentum equation is first discretized, and an intermediate velocity field is solved using the pressure field obtained from the previous iteration ($p^{t-\Delta t}$).

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

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, 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 from 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 [41] 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.

To connect the turbulent viscosity to the mean flow variables and close the system, a turbulence model must be used. The Spalart–Allmaras (SA) model solves:

$$\frac{\partial \tilde{\nu}}{\partial t} + \nabla \cdot (\boldsymbol{U}\tilde{\nu}) - \frac{1}{\sigma}\{\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] + C_{b2}|\nabla\tilde{\nu}|^2\} - \beta C_{b1}\tilde{S}\tilde{\nu} + C_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2 = 0. \tag{11}$$

where $\tilde{\nu}$ is the modified viscosity, which can be related to the turbulent eddy viscosity via

$$\nu_t = \tilde{\nu}\frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}. \tag{12}$$

Refer to Spalart and Allmaras [42] for a more detailed description of the terms and parameters in the SA model.

Another turbulence model used in this study is the $k-\omega$ SST model. It solves the following two equations:

$$\frac{\partial \omega}{\partial t} = \nabla \cdot (D_\omega \nabla \omega) + \frac{\gamma G}{\nu} - \frac{2}{3}\gamma\omega(\nabla \cdot \boldsymbol{U}) - \beta\omega^2 - (F_1 - 1)CD_{k\omega} + S_\omega, \tag{13}$$

$$\frac{\partial k}{\partial t} = \nabla \cdot (D_k \nabla k) + G - \frac{2}{3}k(\nabla \cdot \boldsymbol{U}) - \beta^*\omega k + S_k, \tag{14}$$

and then the turbulence viscosity is obtained as:

$$\nu_t = a_1 \frac{k}{\max(a_1\omega, b_1 F_{23} S)}. \tag{15}$$

Refer to Menter et al. [43] for a more detailed description of the terms and parameters in the $k-\omega$ SST model.

In addition to the PISO corrector loop mentioned above, the PIMPLE algorithm repeatedly solves Eqs. (3) to (15) multiple times until all the flow residuals are small (PIMPLE outer corrector loop). We use Eqs. 3, 8, 9, and 11 for the residuals of velocity, pressure, face flux, and turbulence variables, respectively. The PIMPLE method allows us to use a relatively large time step size (CFL>1), therefore, it can reduce the number of time steps for the flow and adjoint computation. In this study, we run the PIMPLE outer corrector loop until all the flow residuals drop 6 orders of magnitude. It takes about 30 outer iterations. Note that having tightly converged flow residuals is critical for the accuracy of the adjoint method, as shown in the following sections. Our proposed adjoint method applies to the PIMPLE method only. It will have large errors for other segregated time-accurate methods that do not have well-defined, tightly converged flow residuals at each time step, such as PISO.

B. General time-accurate discrete adjoint formulation for computing derivatives

Consider a general discretized time-marching primal problem using a first-order temporal scheme written in the residual form:

$$\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{w}) = \begin{bmatrix} \boldsymbol{R}^1(\boldsymbol{x}, \boldsymbol{w}^1, \boldsymbol{w}^0) \\ \boldsymbol{R}^2(\boldsymbol{x}, \boldsymbol{w}^2, \boldsymbol{w}^1) \\ \vdots \\ \boldsymbol{R}^K(\boldsymbol{x}, \boldsymbol{w}^K, \boldsymbol{w}^{K-1}) \end{bmatrix} = \boldsymbol{0}, \tag{16}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is the design variables, $\boldsymbol{w}^0 \in \mathbb{R}^{n_w}$ is the initial state, and K is the number of time steps. The time-marching primal problem in Eq. (16) depends on the design variables $\boldsymbol{x}$, and it is solved in a forward fashion to determine the state variable $\boldsymbol{w} \in \mathbb{R}^{Kn_w}$ consisting of the states for all time steps, i.e., $\boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^K \in \mathbb{R}^{n_w}$. Here we assume all the flow residuals converge tightly at each time step.

The objective function F depends on both the design variables $\boldsymbol{x}$ and the state variable $\boldsymbol{w}$ solved in Eq. (16), and in many applications including this study, the objective function F can be expressed as the average of a time series, that is:

$$F(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{K}\sum_{i=1}^{K} f^i(\boldsymbol{x}, \boldsymbol{w}^i), \tag{17}$$

where for each $1 \le i \le K$, f^i only depends on the design variables $\boldsymbol{x}$ and the corresponding state $\boldsymbol{w}^i$. This implies that the partial derivative $\partial F/\partial \boldsymbol{w}$ can be simplified as:

$$\underbrace{\frac{\partial F}{\partial \boldsymbol{w}}}_{1\times Kn_w} = \frac{1}{K}\Big[\underbrace{\frac{\partial f^1}{\partial \boldsymbol{w}^1}}_{1\times n_w}\ \underbrace{\frac{\partial f^2}{\partial \boldsymbol{w}^2}}_{1\times n_w}\ \cdots\ \underbrace{\frac{\partial f^K}{\partial \boldsymbol{w}^K}}_{1\times n_w}\Big]. \tag{18}$$

For other common types of objective functions, e.g., the variance of a time series, the partial derivatives of F can also be simplified in a similar manner.

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

$$\underbrace{\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}}_{1\times n_x} = \underbrace{\frac{\partial F}{\partial \boldsymbol{x}}}_{1\times n_x} + \underbrace{\frac{\partial F}{\partial \boldsymbol{w}}}_{1\times Kn_w}\underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}}}_{Kn_w\times n_x}, \tag{19}$$

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:

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{x}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}} \frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}} = 0. \tag{20}$$

Rearranging the above equation, we get the linear system

$$\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}}_{Kn_w \times Kn_w} \cdot \underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}}}_{Kn_w \times n_x} = - \underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}}_{Kn_w \times n_x}. \tag{21}$$

We can then substitute the solution for $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{x}$ from Eq. (21) into Eq. (19) to get

$$\underbrace{\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}}_{1 \times n_x} = \underbrace{\frac{\partial F}{\partial \boldsymbol{x}}}_{1 \times n_x} - \overbrace{\underbrace{\frac{\partial F}{\partial \boldsymbol{w}}}_{1 \times Kn_w} \underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}^{-1}}_{Kn_w \times Kn_w}}^{\boldsymbol{\psi}^T} \underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}}_{Kn_w \times n_x}. \tag{22}$$

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*,

$$\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}^T}_{Kn_w \times Kn_w} \cdot \underbrace{\boldsymbol{\psi}}_{Kn_w \times 1} = \underbrace{\frac{\partial F}{\partial \boldsymbol{w}}^T}_{Kn_w \times 1}, \tag{23}$$

where $\boldsymbol{\psi}$ is the *adjoint vector*. Then, we can compute the total derivative by substituting the adjoint vector into Eq. (22):

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}} = \frac{\partial F}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}. \tag{24}$$

Since the design variables are not explicitly present in Eq. 23, 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 field inversion because typically, there is only one objective function, but thousands of design variables can be used.

The adjoint equation in Eq. (23) can be simplified for the time-marching primal problem. As indicated in Eq. (16), for each $1 \leq i \leq K$, $\boldsymbol{R}^i$ only has dependency on $\boldsymbol{x}$, $\boldsymbol{w}^i$, and $\boldsymbol{w}^{i-1}$. Together with the simplification in Eq. (18), Eq. (23) can be rewritten as:

$$\begin{bmatrix} \frac{\partial \boldsymbol{R}^1}{\partial \boldsymbol{w}^1}^T & \frac{\partial \boldsymbol{R}^2}{\partial \boldsymbol{w}^1}^T & & & \\ & \frac{\partial \boldsymbol{R}^2}{\partial \boldsymbol{w}^2}^T & \frac{\partial \boldsymbol{R}^3}{\partial \boldsymbol{w}^2}^T & & \\ & & \ddots & \ddots & \\ & & & \frac{\partial \boldsymbol{R}^{K-1}}{\partial \boldsymbol{w}^{K-1}}^T & \frac{\partial \boldsymbol{R}^K}{\partial \boldsymbol{w}^{K-1}}^T \\ & & & & \frac{\partial \boldsymbol{R}^K}{\partial \boldsymbol{w}^K}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^1 \\ \boldsymbol{\psi}^2 \\ \vdots \\ \boldsymbol{\psi}^{K-1} \\ \boldsymbol{\psi}^K \end{bmatrix} = \frac{1}{K} \begin{bmatrix} \frac{\partial f^1}{\partial \boldsymbol{w}^1}^T \\ \frac{\partial f^2}{\partial \boldsymbol{w}^2}^T \\ \vdots \\ \frac{\partial f^{K-1}}{\partial \boldsymbol{w}^{K-1}}^T \\ \frac{\partial f^K}{\partial \boldsymbol{w}^K}^T \end{bmatrix}, \tag{25}$$

where the adjoint vector $\boldsymbol{\psi} \in \mathbb{R}^{Kn_w}$ is broken down into K parts that correspond to the time steps, i.e., $\boldsymbol{\psi}^1, \boldsymbol{\psi}^2, \ldots, \boldsymbol{\psi}^K \in$

$\mathbb{R}^{n_w}$. Then, Eq. (25) can be solved sequentially in a backward fashion as:

$$
\begin{aligned}
&\underbrace{\frac{\partial \boldsymbol{R}^{K}}{\partial \boldsymbol{w}^{K}}^{T}}_{n_w \times n_w} \cdot \underbrace{\boldsymbol{\psi}^{K}}_{n_w \times 1} = \frac{1}{K} \underbrace{\frac{\partial f^{K}}{\partial \boldsymbol{w}^{K}}^{T}}_{n_w \times 1}, \\
&\underbrace{\frac{\partial \boldsymbol{R}^{i}}{\partial \boldsymbol{w}^{i}}^{T}}_{n_w \times n_w} \cdot \underbrace{\boldsymbol{\psi}^{i}}_{n_w \times 1} = \frac{1}{K} \underbrace{\frac{\partial f^{i}}{\partial \boldsymbol{w}^{i}}^{T}}_{n_w \times 1} - \underbrace{\frac{\partial \boldsymbol{R}^{i+1}}{\partial \boldsymbol{w}^{i}}^{T}}_{n_w \times n_w} \cdot \underbrace{\boldsymbol{\psi}^{i+1}}_{n_w \times 1}, \; K-1 \geq i \geq 1,
\end{aligned}
\tag{26}
$$

which effectively breaks down the original adjoint equation in Eq. (23) into K much smaller sub-equations. The right-hand side terms in Eq. (26) can be efficiently evaluated with reverse-mode AD. In particular, the matrix-transpose-vector product $[\partial \boldsymbol{R}^{i+1}/\partial \boldsymbol{w}^{i}]^{T}\boldsymbol{\psi}^{i+1}$ is evaluated in a Jacobian-free manner right after we solve for $\boldsymbol{\psi}^{i+1}$, then it is passed to the right-hand side of the sub-equation for $\boldsymbol{\psi}^{i}$. We will elaborate on how we solve each sub-equation in Eq. (26) with a fixed-point iteration scheme in Sec. II.C.

The assumption that the objective function F is of the average type in Eq. (17) also simplifies the expression of the total derivative $\mathrm{d}F/\mathrm{d}\boldsymbol{x}$ in Eq. (24) as:

$$
\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}} = \sum_{i=1}^{K} \left(\frac{1}{K} \frac{\partial f_i}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^{i\,T} \frac{\partial \boldsymbol{R}^{i}}{\partial \boldsymbol{x}} \right). \tag{27}
$$

Therefore, we can calculate the total derivative accumulatively as we sequentially solve the sub-equations in Eq. (26).

The general time-accurate discrete adjoint formulation presented here assumes a first-order temporal scheme and it can be readily extended to second- or higher-order schemes. If we use a higher-order temporal scheme, $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^{T}$ will maintain an upper-triangular and banded structure similar to that shown on the left-hand side of Eq. (25), albeit with increased bandwidth, then the adjoint equation will still be solved sequentially in a backward fashion similar to Eq. (26). For example, with a second-order temporal scheme, Eq. (26) will be replaced by:

$$
\begin{aligned}
&\frac{\partial \boldsymbol{R}^{K}}{\partial \boldsymbol{w}^{K}}^{T} \cdot \boldsymbol{\psi}^{K} = \frac{1}{K} \frac{\partial f^{K}}{\partial \boldsymbol{w}^{K}}^{T}, \\
&\frac{\partial \boldsymbol{R}^{K-1}}{\partial \boldsymbol{w}^{K-1}}^{T} \cdot \boldsymbol{\psi}^{K-1} = \frac{1}{K} \frac{\partial f^{K-1}}{\partial \boldsymbol{w}^{K-1}}^{T} - \frac{\partial \boldsymbol{R}^{K}}{\partial \boldsymbol{w}^{K-1}}^{T} \cdot \boldsymbol{\psi}^{K}, \\
&\frac{\partial \boldsymbol{R}^{i}}{\partial \boldsymbol{w}^{i}}^{T} \cdot \boldsymbol{\psi}^{i} = \frac{1}{K} \frac{\partial f^{i}}{\partial \boldsymbol{w}^{i}}^{T} - \frac{\partial \boldsymbol{R}^{i+1}}{\partial \boldsymbol{w}^{i}}^{T} \cdot \boldsymbol{\psi}^{i+1} - \frac{\partial \boldsymbol{R}^{i+2}}{\partial \boldsymbol{w}^{i}}^{T} \cdot \boldsymbol{\psi}^{i+2}, \; K-2 \geq i \geq 1.
\end{aligned}
\tag{28}
$$

Note that the only difference between Eq. (28) that corresponds to a second-order temporal scheme scenario and Eq. (26) that assumes a first-order temporal scheme is an extra right-hand side term $[\partial \boldsymbol{R}^{i+2}/\partial \boldsymbol{w}^{i}]^{T}\boldsymbol{\psi}^{i+2}$ for $K-2 \geq i \geq 1$. This extra term for a second-order temporal scheme is evaluated with reverse-mode AD in a Jacobian-free manner right after we solve for $\boldsymbol{\psi}^{i+2}$, then it is passed to the right-hand side of the sub-equation for $\boldsymbol{\psi}^{i}$.

C. Segregated iteration scheme for solving the PIMPLE-adjoint equations

We present a segregated fixed-point iteration scheme to solve the adjoint equation's sub-equations in Eq. (26) for the PIMPLE method. Instead of rigorously deriving the primal preconditioner matrix to preserve duality as we did in previous work [44, 45], we use the primal equations' left-hand side coefficient matrices to form a block-diagonal matrix as its substitute. Then, we transpose this block-diagonal matrix to get the non-dual adjoint preconditioner (X_{nd}^{T}), as shown in Eq. (29). Using the turbulent flow with the SA model as an example, to determine $\boldsymbol{\psi}^{i}$, we solve:

$$
\underbrace{\begin{bmatrix} \boldsymbol{A}_{\boldsymbol{U}}^{i\,T} & & & \\ & \boldsymbol{A}_{p}^{i\,T} & & \\ & & -\boldsymbol{I}_{\phi} & \\ & & & \boldsymbol{A}_{\tilde{\nu}}^{i\,T} \end{bmatrix}}_{X_{\mathrm{nd}}^{T}} \Delta\boldsymbol{\psi}^{i} = -\frac{\partial \boldsymbol{R}^{i}}{\partial \boldsymbol{w}^{i}}^{T} \boldsymbol{\psi}^{i} \underbrace{- \frac{\partial \boldsymbol{R}^{i+1}}{\partial \boldsymbol{w}^{i}}^{T} \boldsymbol{\psi}^{i+1} + \frac{1}{K} \frac{\partial f^{i}}{\partial \boldsymbol{w}^{i}}^{T}}_{\boldsymbol{b}^{i}}, \tag{29}
$$

Table 1 Comparison of speed and memory between the unsteady flow and PIMPLE-adjoint. The adjoint is twice as slow as the flow. In addition, the adjoint uses 4 times as much memory as the flow.

	Flow	Adjoint	Adjoint/Flow
Runtime	123 s	242 s	1.97
Memory	88 MB	362 MB	4.11

Table 2 Verification of the adjoint accuracy. The average error between the adjoint derivatives and finite-difference references is less than 1%.

$\mathrm{d}f/\mathrm{d}\beta_i$	Adjoint	Finite-difference	Error
0	−0.05417	−0.05382	−0.64%
1	−0.14973	−0.14875	−0.65%
2	−0.01180	−0.01171	−0.77%
3	−0.17132	−0.17018	−0.67%
4	−0.07386	−0.07333	−0.72%

where $\Delta\boldsymbol{\psi}^i$ is the update on $\boldsymbol{\psi}^i$. $\boldsymbol{A}_{\boldsymbol{U}}^i$, $\boldsymbol{A}_p^i$, and $\boldsymbol{A}_{\tilde{\nu}}^i$ are respectively the left-hand side coefficient matrices of the discretized primal equations for $\boldsymbol{U}$, p, and $\tilde{\nu}$ at the i$^{\text{th}}$ time step, constructed from the converged state. The transposed ${\boldsymbol{A}_{\boldsymbol{U}}^i}^T$, ${\boldsymbol{A}_p^i}^T$, and ${\boldsymbol{A}_{\tilde{\nu}}^i}^T$ are constructed by swapping the upper and lower coefficient arrays, and their inverse is applied approximately through iterative methods. Note that the right-hand side terms under-braced as $\boldsymbol{b}^i$ in Eq. (29) remains constant during the fixed-point iterations, and the term $-[\partial \boldsymbol{R}^{i+1}/\partial \boldsymbol{w}^i]^T\boldsymbol{\psi}^{i+1}$ is omitted for $i = K$.

We solve Eq. (29) using the block Gauss-Seidel method, and therefore the adjoint iterations are performed in a *segregated* manner ($U \rightarrow p \rightarrow \phi \rightarrow \tilde{\nu}$), similar to that of the primal PIMPLE solver. To stabilize the solution, we use an under-relaxation factor of 0.3 for updating the pressure adjoint variable. No under-relaxation is used for other variables.

While the segregated fixed-point iteration scheme for time-accurate PIMPLE-adjoint presented here has the advantage of being relatively easy to implement, the lack of a rigorous duality-preserving property means that it may not converge well for all flow problems. In addition, the PIMPLE outer loop needs more iterations when larger CFL numbers are used. To further speed up the adjoint computation, we will use the Jacobian-free Krylov method [39] to solve Eq. 29 in a coupled manner in the future.

D. Performance evaluation for the proposed time-accurate PIMPLE-adjoint

First, we evaluate the computational speed of the proposed PIMPLE-adjoint algorithm. We use a turbulent flow over a ramp as the benchmark. The mesh has 4000 cells. We run the unsteady flow simulation for a short time period, i.e., 0.1 s. The step size is 0.005 s, so the simulation runs for 20 steps. The corresponding CFL number is about 3. This setup is sufficient for speed test purposes because the computational cost will scale linearly for longer simulations. After the flow simulation is done, we run the adjoint gradient computation. We ask the PIMPLE outer corrector loop to converge the flow and adjoint residuals by 6 and 5 orders of magnitude, respectively. The computation was done on the Iowa State University Nova HPC system. The Nova HPC is equipped with Intel Skylake Xeon processors running at 2.3G Hz. Table 1 shows the comparison of speed and memory usage between the unsteady flow and adjoint computation. The adjoint is twice as slow as the flow. In addition, the adjoint uses 4 times as much memory as the flow. Note that the flow simulation uses the AD-version of the primal solver pimpleFoam.

Next, we evaluate the total derivative accuracy of the proposed PIMPLE-adjoint approach, as shown in Table 2. The accuracy of the total derivatives is important for the robustness of field inversion. Inaccuracy derivatives will mislead the optimization and result in sub-optimal results. We pick the first five cells and run the unsteady field inversion to compute the total derivatives $\mathrm{d}f/\mathrm{d}\beta$. The objective function f is defined as the error between the CFD prediction and reference data. We use the finite-difference method to compute reference derivatives. The average error in the unsteady adjoint derivatives is less than 1%.

Overall, our PIMPLE-adjoint exhibits acceptable speed and accuracy and is ready to conduct field inversion optimization.

Table 3 Summary of the field inversion cases conducted in this study.

Case	Description
Lam-Single-Coeff	Laminar flow case that uses single probe point data for the objective function and the coefficients of the Gaussian α field as design variables.
Lam-Single-Field	Laminar flow case that uses single probe point data for the objective function and the cell-wise α field as design variables.
Lam-Profile-Field	Laminar flow case that uses probe point data from a vertical line for the objective function and the cell-wise α field as design variables.
Turb-Profile-Field	Turbulent flow case that uses probe point data from a vertical line for the objective function and the cell-wise α field as design variables.

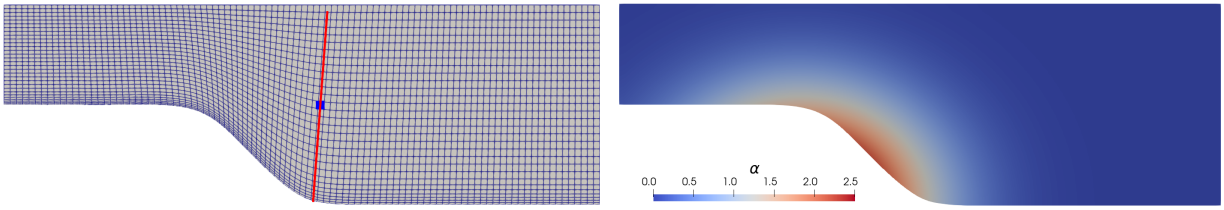

Fig. 1 Left: Structured mesh for the ramp with 3000 cells. The single probe point is denoted in blue, and the probed vertical profile is denoted in red. Right: the prescribed Gaussian α field.

III. Results and Discussion

In this section, we use the PIMPLE-adjoint to conduct unsteady field inversion using flow over a ramp as the benchmark. We first use a laminar case to consider the formulation of unsteady field inversion problems by using various options to construct the objective function. Then, we consider a turbulent case and use the unsteady field inversion to augment the production term of the Spalart–Allmaras (SA) turbulence model. We conduct four field inversion optimizations with different configurations, as shown in Table 3. We use the first three cases to discuss proper optimization formulations for unsteady field inversion. Finally, we demonstrate the capability to augment turbulence models for improving unsteady flow prediction using the last case.

A. Problem formulation for unsteady field inversion

Formulating a proper objective function, i.e., the error between the CFD prediction and reference data, is important for the success of field inversion. The unsteady field inversion considers the spatial-temporal evolution of the field fields. Therefore, it is more vulnerable to local minima issues than the steady-state field inversion cases, if the objective function is not properly formulated. To investigate this issue, this section considers a few objective function formulations.

As mentioned above, this section considers unsteady laminar flow over a 45-degree ramp, as shown in Fig. 1 left. 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 is 5000. We generated a structured mesh with 3000 cells and the y^+ is less than 1.0. We use a uniform velocity field (5 m/s) as the initial condition and run unsteady simulations for 1 s. The flow undergoes a transient process at and downstream the ramp, before reaching a steady state. The unsteady field inversion will consider the spatial-temporal distribution of the velocity during the transient process.

Next, we consider a Gaussian impermeability field (α) for the momentum equation (Eq. 2):

$$\alpha = ae^{\frac{(x-x_C)^2+(y-y_C)^2}{\sigma_x^2+\sigma_y^2}} \tag{30}$$

where x_C, y_C, σ_x, σ_y, and a are 0.9, −0.5, 0.6, 0.5, and 4.0, respectively. The contour of the prescribed α field is shown in Fig. 1 right. We place the α field near the ramp to block the flow and change the flow behavior during the transient process.

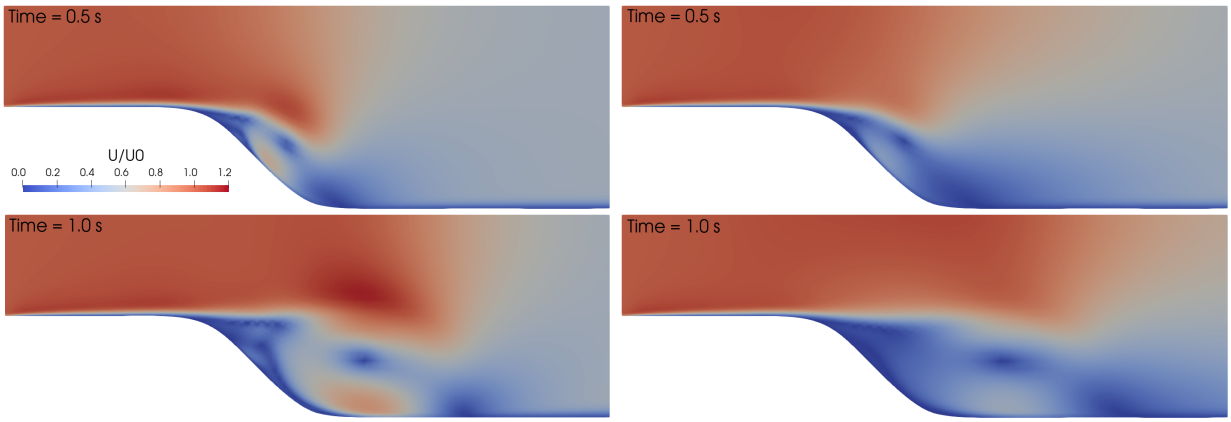

Fig. 2 Velocity contours at different times, computed with all-zero α field (left; baseline) and prescribed Gaussian α field (right; reference). The prescribed α field modifies the unsteady flow structure downstream.

Table 4 Formulation of the Lam-Single-Coeff case. We use a single probe point at $[x/h, y/h] = [1.6, 0.0]$ (see Fig. 1 left), and we use the Gaussian distribution coefficients as design variables. The objective function is the velocity time series error between the CFD prediction and reference data. The reference velocity time-series ($\boldsymbol{U}_{\mathrm{t}}^{\mathrm{ref}}$) is generated by running CFD with the prescribed α field.

	Function/Variable	Description	Quantity
Min	$\sum_{t=t_1:t_n} \parallel \boldsymbol{U}_{\mathrm{t}}^{\mathrm{CFD}} - \boldsymbol{U}_{\mathrm{t}}^{\mathrm{ref}} \parallel^2$	Velocity time series error between the CFD and reference at a single probe point	1
w.r.t.	x_C and y_C	Center coordinates of Gaussian function	2
	σ_x and σ_y	Standard deviation of Gaussian function	2
	a	Scaling factor of Gaussian function	1
		Total Design Variables	**5**

Figure 2 shows the velocity contours at $t = 0.5$ and 1.0 s with the left and right columns being the results simulated without and with the prescribed α field, respectively. A transient vortex develops at the ramp near $t = 0.5$ s, and this vortex propagates and interacts with the main flow further downstream. When comparing the left and right columns, we find that the prescribed α field reduces the strength of the vortex. We will use the flow field simulated with the prescribed α as the reference data (right column). Then, we run unsteady field inversion to find the prescribed α field distribution, starting from a flow field with zero-α (left column). We will use the error between the CFD-predicted velocity field and the above reference data as the objective function. In other words, the field inversion will change the α field to match the spatial-temporal velocity distribution from the right column. We consider three options to formulate the objective function and design variables.

Option 1: Gaussian coefficients as design variables and single probe point data for the objective function (Lam-Single-Coeff)

Table 4 shows the field inversion problem formulation for the first configuration (Lam-Single-Coeff). It simplifies the field inversion by using the Gaussian function coefficients (x_C, y_C, σ_x, σ_y, and a) as the design variables, instead of using the full (cell-wise) α field. Once these coefficients are determined, we will compute the α field and add it to the momentum equation. We use the square difference of CFD-predicted and reference velocity time series at a single probe point $[x/h, y/h] = [1.6, 0.0]$ as the objective function. The probe point is shown in blue in Fig. 1 left. This configuration mimics the calibration of physical model coefficients during machine learning. We use the sparse nonlinear optimizer (SNOPT [46]) to perform large-scale gradient-based optimization.

Figure 3 shows the optimized α field and velocity contour after the field inversion. The Gaussian function coefficients found by the field inversion are: $x_C = 0.9000$, $y_C = -0.5000$, $\sigma_x = 0.6000$, $\sigma_y = 0.5000$, and $\alpha = 4.0004$, which agree

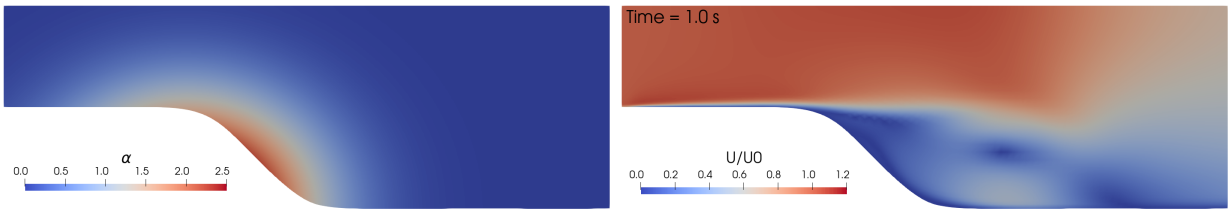

Fig. 3 Left: Optimized α field for the Lam-Single-Coeff case. The inverse coefficients are: $x_C = 0.9000$, $y_C = -0.5000$, $\sigma_x = 0.6000$, $\sigma_y = 0.5000$, and $\alpha = 4.0004$. Right: The resulting velocity field at Time = 1.0. The field inversion finds the correct coefficients.

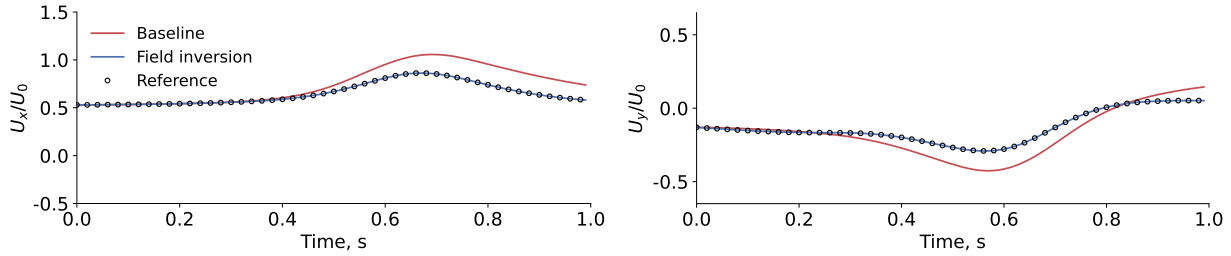

Fig. 4 Velocity time series at the single probe point (Lam-Single-Coeff). The field inversion matches the reference reasonably well.

reasonably well with the prescribed values (see Eq. 30). The inverse beta fields also agree well with the reference (comparing Fig. 3 left and Fig. 1 right). Moreover, the inverse velocity field agrees reasonably well with the reference one at t =1.0 s (comparing Fig. 3 right and Fig. 2 bot-right).

In summary, we find that using the velocity time series at a single probe point to formulate the objective function is sufficient to find the prescribed α Gaussian function coefficients for this configuration. This is expected because we have a relatively limited number of design freedom (only five design variables), so the data requirement is also minimized for the unsteady field inversion.

This case proves that our proposed PIMPLE-adjoint is robust and efficient in conducting unsteady field inversion optimization and serves as a benchmark for the following cases. Next, we will explore the impact of adding more design freedom by considering cell-wise field inversion.

Option 2: Cell-wise α field as design variables and single probe point data for the objective function (Lam-Single-Field)

Table 5 shows the field inversion problem formulation for the second configuration (Lam-Single-Field). This configuration is similar to the one in Table 4, except that we use the cell-wise α field as design variables. Therefore, this configuration has significantly larger design freedom (3000 design variables) and is more vulnerable to local minima issues.

We run the field inversion optimization for 100 iterations. The initial and optimized optimalities are 2.5e-2 and 2.0e-6, respectively. The initial and optimized objective functions are 5.4e-1 to 3.4e-6, respectively. The optimization converges well and tightly. This can be further confirmed by the velocity time-series at the single probe point, as shown in Fig. 4. The baseline (zero α field) velocity time series differs significantly from the reference value (prescribed α field). After the field inversion, the velocity time series agrees reasonably well with the reference. This is expected because we use the velocity time series prediction error as the objective function in the field inversion.

Figure 5 shows the optimized beta field and velocity contour after the field inversion. The field inversion finds the overall location of the prescribed α field (i.e., high α near the ramp). However, the detailed α field distribution is not correctly captured. It shows high concentrations of α at locations away from the ramp wall surface, which deviates significantly from the prescribed α in Fig. 1 right. As a result, the predicted velocity contour exhibits artificial oscillation at the upstream locations, as shown in Fig. 5 right.

To further illustrate the artificial oscillation, we plot the velocity profiles at various streamwise (x) locations (t = 1 s) in Fig. 6. Overall, the inverse velocity profiles agree with the reference values. However, there are noticeable

Table 5 Formulation of the Lam-Single-Field case. The configuration is similar to Table 5 except that we use the full cell-wise α field as the design variables.

	Function/Variable	Description	Quantity
Min	$\sum_{t=t_1:t_n} \parallel \boldsymbol{U}_{\mathrm{t}}^{\mathrm{CFD}} - \boldsymbol{U}_{\mathrm{t}}^{\mathrm{ref}} \parallel^2$	Velocity time series error between the CFD and reference at a single probe point	1
w.r.t.	$0 \leq \alpha \leq 5$	Cell-wise α field in the momentum equation	3000

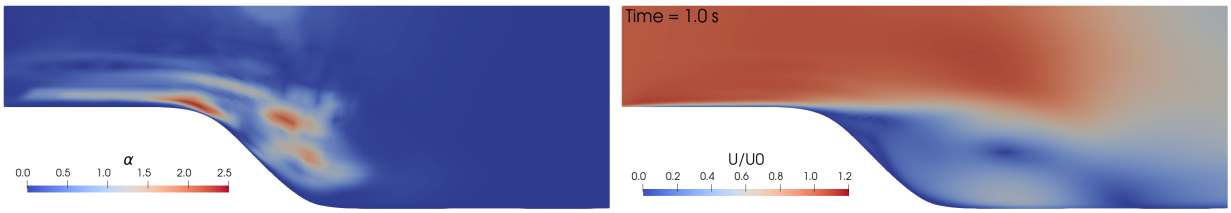

Fig. 5 Left: Optimized α field for the Lam-Single-Field case. It shows high concentrations of α at locations away from the ramp wall surface, which deviates significantly from the prescribed α in Fig. 1. Right: The resulting velocity field at Time = 1.0 s. Compared with the prescribed α reference in Fig. 2, it exhibits artificial U features at upstream locations.

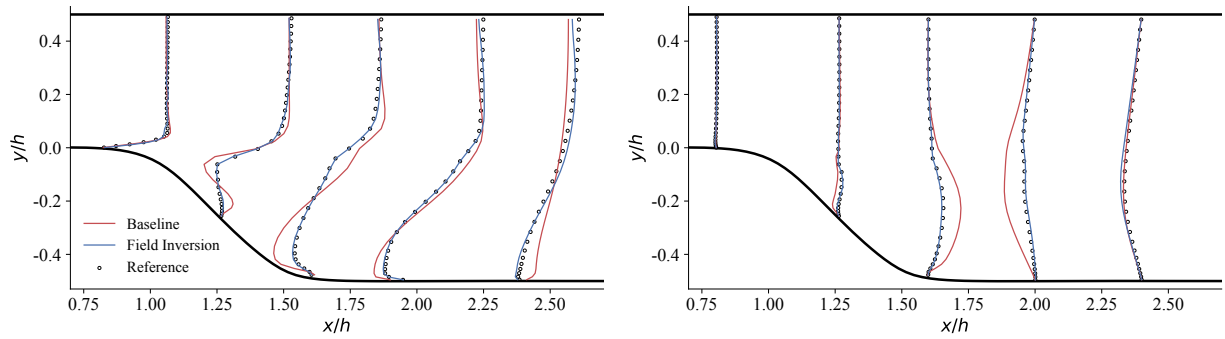

Fig. 6 Velocity profiles (left: U_x and right: U_y) at various streamwise locations ($t = 1$ s; Lam-Single-Field). Although the field inversion drives the velocity profiles to the reference values, there are small discrepancies, e.g., U_x at the upstream $[x/h, y/h] = [1.0, 0.1]$ and downstream $[x/h, y/h] = [2.5, 0.0]$ regions.

discrepancies in the upstream $[x/h, y/h] = [1.0, 0.1]$ and downstream $[x/h, y/h] = [2.5, 0.0]$ regions for U_x.

Based on the above results (the field inversion optimization converges tightly but the predicted velocity field has noticeable errors), we conclude that using velocity time series at one probe point to formulate the objective function is not sufficient when using field variables as design variables. This is because the flow data from one probe point includes flow physics only at a certain vertical (y) level. The field inversion can create a perfect α field to reproduce the exact flow physics at that vertical level. However, this does not guarantee the flow at other vertical levels is correctly captured. Although the error shown in Fig. 6 is within the acceptable range for this simple case, we expect the artificial oscillation to be amplified for more complex flows. Next, we will fix this issue by using more data to formulate the objective function.

Option 3: Cell-wise α field as design variables and probe point data from a vertical line for the objective function (Lam-Profile-Field)

Table 6 shows the field inversion problem formulation for the third configuration (Lam-Profile-Field). Here we use the velocity time series of probe points from a vertical profile ($x \approx 1.6$) to formulate the objective function. The profile is shown in red in Fig. 5 left. The objective function is calculated as the velocity time series error between the CFD and reference data at these probe points. Compared with the configuration used in Table 5, we use more data to construct the

Table 6 Formulation of the Lam-Profile-Field case. We use a slice of cells at $x/h \approx 1.6$ (red line in Fig. 1 left) as the probe points, and we use the full cell-wise α field as design variables. The objective function is the velocity time series error between the CFD prediction and reference data.

	Function/Variable	Description	Quantity
Min	$\sum_{y=y_1:y_m} \sum_{t=t_1:t_n} \parallel \boldsymbol{U}^{\text{CFD}}_{\text{t, y}} - \boldsymbol{U}^{\text{ref}}_{\text{t, y}} \parallel^2$	Velocity time series error between the CFD and reference at probe points from a vertical profile	1
w.r.t.	$0 \leq \alpha \leq 5$	Cell-wise α field in the momentum equation	3000

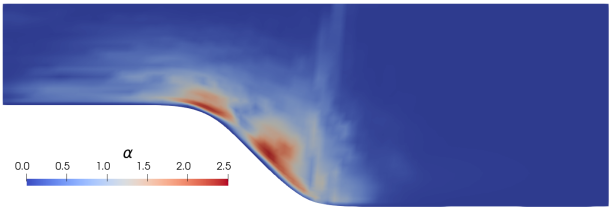

Fig. 7 Optimized α field for the Lam-Profile-Field case. The field inversion captures the overall trend of the prescribed α field.

objective function. We expect this objective function formulation will better cover unsteady flow physics from all the vertical locations.

Again, the field inversion optimization ran for 100 iterations. Similar to the previous case, the objective function and optimality drop 5 and 3 orders of magnitude, and the optimization converges well and tightly.

Figure 7 shows the optimized α field for this configuration. The inverse α field well captures the overall distribution of the prescribed reference (Fig. 1 right), i.e., there is a relatively high α region right above the ramp surface and the magnitude decreases outwards. Although the inverse α field is not as smooth as the reference, the agreement is significantly improved compared with the Lam-Single-Field case.

Figure. 8 shows the inverse velocity contours for the Lam-Profile-Field case. The velocity distribution is virtually identical to the reference one shown in Fig. 2 right. To further quantify the comparison, we plot the velocity profiles at various streamwise locations in Fig. 9. The baseline velocity profiles are significantly different from the reference at $t = 0.5$ and 1.0 s. After the field inversion, the inverse velocity profiles agree reasonably well at all streamwise and vertical locations. This good agreement supports the usage of probe point data from a vertical profile, instead of a single point.

To evaluate the field inversion's capability to capture the temporal evolution of velocity, we plot the velocity time series at three selected probe points ($[x/h, y/h] = [1.3, -0.1], [1.6, -0.2], [2.0, -0.2]$) in Fig. 10. Similar to what we find for the velocity profiles, the inverse velocity time series agrees reasonably well with the reference at all time instances. Note that we use only the velocity time series from one of the above three points ($[x/h, y/h] = [1.6, -0.2]$) to construct the objective function, and the other two probe points are not used in field inversion. This indicates that using probe points from one vertical profile for objective function can effectively find the α field that predicts the correct spatial-temporal distribution of the whole velocity field.

In summary, we find that using probe point data from one velocity profile for the objective function is sufficient for the field inversion. The spatial-temporal distribution of the inverse velocity field agrees well with the reference value at all time instances and all locations. The streamwise location of the probe profile plays an important role in formulating the field inversion problems. We set the probe profile to be at $x \approx 1.6$ because it is downstream of the most important unsteady flow physics (i.e., the ramp vortex). Therefore, the velocity time-series data include most of the important information that will be propagated back to compute the correct α field during the field inversion process. If we were to set the probe location to further upstream (e.g., on top of the ramp, $x = 1.0$), the velocity time series would not include any information about the flow separation on the ramp. Therefore, the field inversion would not compute the correct α distribution. Another thing to point out is that, although the inverse velocity fields agree reasonably well with the reference, the inverse α field is still not identical to the prescribed reference. This suggests that we have multiple local minima for this field inversion problem, which is not unexpected given the large design freedom. It is still unclear how

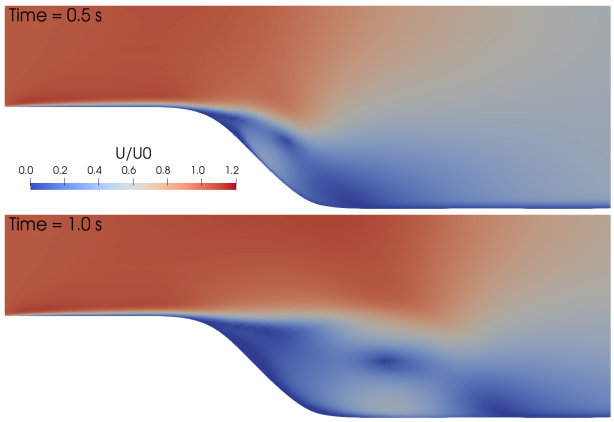

Fig. 8 Inverse velocity contours for the Lam-Profile-Field case. Top: $t = 0.5$ s. Bot: $t = 1.0$ s. They agree well with the reference in Fig. 2 right (prescribed α).

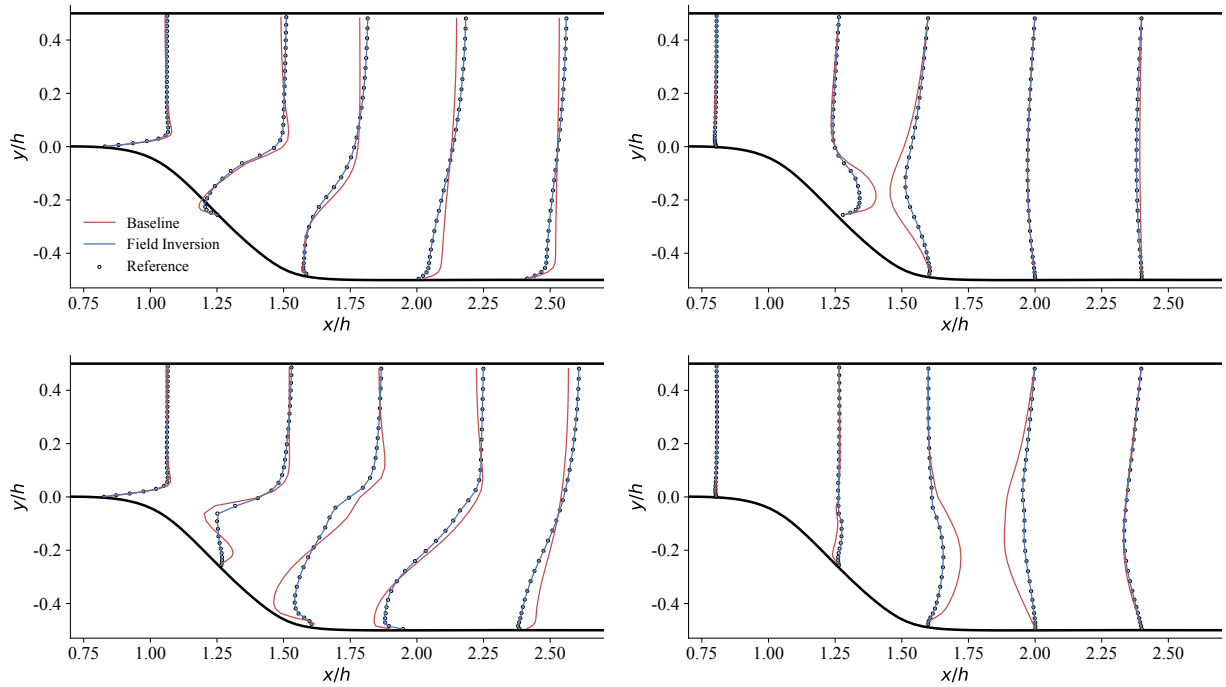

Fig. 9 Velocity profiles (left: U_x and right: U_y) at various streamwise locations (top: $t = 0.5$ s. bot: $t = 1.0$ s; Lam-Profile-Field). The inverse velocity agrees reasonably well with the reference.

the multiple local minima will impact machine learning when we parameterize the inverse α field for predicting unseen flow conditions. Further studies are needed in this direction.

B. Field inversion to augment turbulence models for predicting unsteady flow

Having formulated a proper field inversion problem using unsteady laminar flow, we consider a turbulent flow case in this section. We use the same ramp geometry and refine the mesh near the wall to maintain $y^+ < 1$. The turbulent case has a mesh of 4000 cells. We increase the incoming flow velocity to 10 m/s and reduce the viscosity to 1×10^{-4}, so the Reynolds number is 10^5. To generate a reference velocity field, we run the unsteady simulation for 3 s using the $k - \omega$ SST turbulence model. The time step is 0.005 s with a CFL number of about 3.5.

Then, we run field inversion using the Spalart–Allmaras turbulence model. We multiply the SA model's production

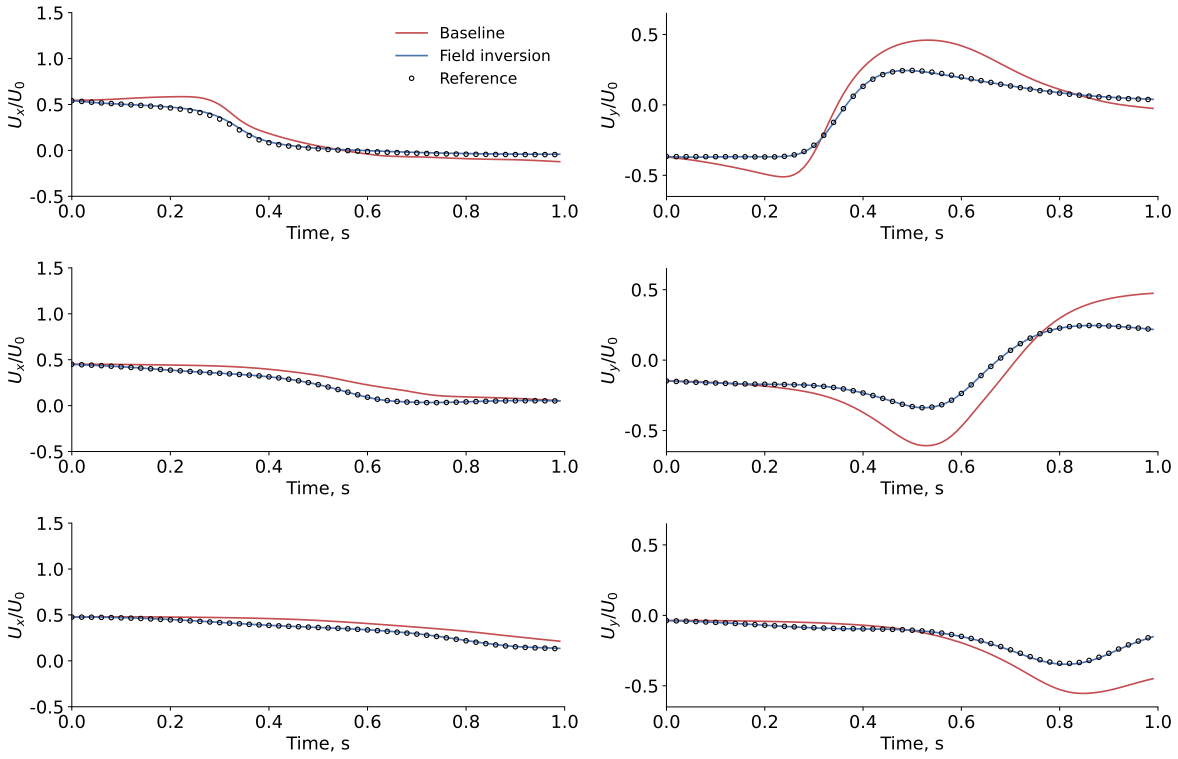

Fig. 10 Velocity time series at various probe points (Top: $[x/h, y/h] = [1.3, -0.1]$, Mid: $[x/h, y/h] = [1.6, -0.2]$, Bot: $[x/h, y/h] = [2.0, -0.2]$. Lam-Profile-Field). The field inversion matches the reference reasonably well.

Table 7 Formulation of the Turb-Profile-Field case. We use a slice of cells at $x/h \approx 1.6$ (red line in Fig. 1 left) as the probe points. We use the full cell-wise β field in the Spalart–Allmaras model as design variables. The objective function is the velocity time series error between the SA prediction and reference data ($k-\omega$ SST).

	Function/Variable	Description	Quantity
Min	$\sum_{y=y_1:y_m} \sum_{t=t_1:t_n} \parallel \boldsymbol{U}^{\mathrm{SA}}_{\mathrm{t,\,y}} - \boldsymbol{U}^{\mathrm{SST}}_{\mathrm{t,\,y}} \parallel^2$	Velocity time series error between the SA and SST reference at probe points from a vertical profile	1
w.r.t.	$0 \le \beta \le 5$	Cell-wise β field in the SA model	4000

term by a β scalar field and optimize the β field distribution to make the SA prediction of unsteady flow match the SST's. To construct the objective function, we extract the velocity time series of probe points from a vertical profile located at $x \approx 1.6$ (Table 7), similar to what we use for option 3 in the previous section. As mentioned above, the velocity time series includes all unsteady flow physics at this probed location, so it can propagate the information back to the β field during the field inversion. The design variables are the cell-wise β field. In total, we have 4000 design variables. To evaluate the capability to predict temporally unseen unsteady flow, we run the field inversion using flow data from 0 to 1 s for the SA model. Then, we add the inverse β field to the SA model, re-run the simulation from 0 to 2 s, and compare the augmented SA's prediction with the SST's.

Note that in practical field inversion, it is more common to use either experimental or high-fidelity (e.g., large-eddy or direct numerical simulation) data as the reference values. Here we use the SST model's simulation data as a reference to demonstrate the field inversion of unsteady flow. The proposed field inversion framework can be easily extended to use experimental or LES/DNS data.

Figure 11 shows the velocity contours at different times, computed with the Spalart–Allmaras model (left; baseline) and $k-\omega$ SST model (right; reference). The flow separates at the ramp and generates a vortex propagating downstream. The ramp vortex structures are similar between the SA and SST predictions at $t = 0.7$ s. The ramp vortex quickly

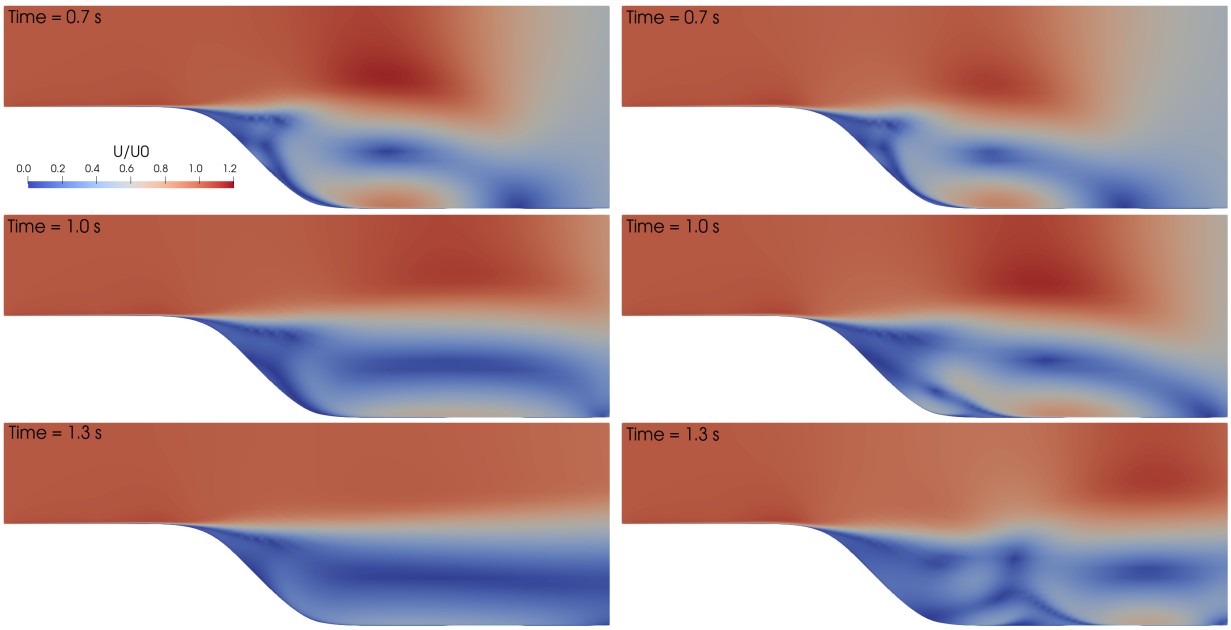

Fig. 11 Velocity contours at different times, computed with the Spalart–Allmaras model (left; baseline) and $k-\omega$ SST model (right; reference). The SST model predicts a strong interaction between the ramp vortex and downstream flow, especially in $t > 1.0$ s, while the vortex intensity predicted by the SA model is mild.

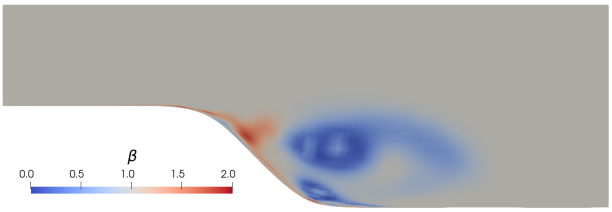

Fig. 12 Optimized β field for the Turb-Profile-Field case. There are strong β variations downstream of the ramp.

dissipates for the SA model, as can be seen in the relatively smooth velocity contour at $t = 1.0$ and 1.3 s (Fig. 11 left). However, the SST model predicts a strong interaction between the ramp vortex and downstream flow, resulting in a much more complex wake structure in $[x/h, y/h] = [2.0, -0.2]$ at $t = 1.0$ and 1.3 s (Fig. 11 right). This is consistent with our previous experience that the SA model generally predicts less flow separation than the SST model. Next, we will run an unsteady field inversion to make the SA model behave like the SST model.

The field inversion runs for 100 iterations. The objective function and optimality are reduced by 2 and 1 orders of magnitude, respectively. The turbulent case does not converge as tightly as the laminar case. However, this level of convergence is acceptable for field inversion, as will be shown later. Figure 12 shows the contour of the inverse β field. We observe strong β variations in the ramp vortex region, which is expected because this region has the highest discrepancy between the SA and SST's velocity predictions. Therefore, the field inversion optimizes the β field in the SA model to match SST's behavior in this region.

Figure 13 shows the inverse velocity contour at different times after the field inversion. The augmented SA model predicts virtually identical velocity distribution at $t = 0.7$ and 1.0 s as the SST model (comparing Fig. 13 with Fig. 11 right). This indicates that using probe points from one vertical profile to construct the objective function is also sufficient for the turbulent case. However, when we use the augmented SA model to predict temporally unseen unsteady flow at $t = 1.3$ s (note that the field inversion ends at 1 s), there are noticeable differences. Although the overall flow structures are similar between the augmented SA and SST models, the ramp vortex structures downstream are slightly different (comparing Fig. 13 bot with Fig. 11 bot-right).

To further investigate this difference, we plot the velocity profiles at various streamwise locations for $t = 0.7$, 1.0,

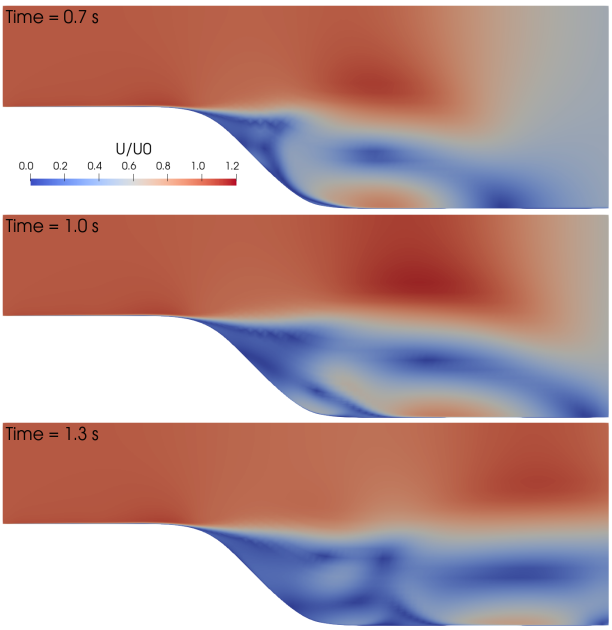

Fig. 13 Inverse velocity contours for the Turb-Profile-Field case. The field inversion captures the overall trend of the SST reference albeit with some discrepancies.

and 1.3 s. Similar to what we observed in Fig. 13, the inverse velocity agrees reasonably well with the reference data for $t = 0.7$ and 1.0 s. However, the error increases at $t = 1.3$ s, especially in the region $[x/h, y/h] = [2.0, -0.3]$. This can be also seen in the velocity time-series at various probe points (Fig. 15). The temporal evolution of velocity predicted by the augmented SA model matches the SST model reasonably well in $t < 1.0$ s. However, in $t > 1.0$ s, there are noticeable differences. In general, the error grows as the prediction time increases.

Given the above results, we conclude that using probe points from one vertical profile to construct the objective function is sufficient to include all relevant unsteady flow physics for the turbulent case. Again, this is mainly because the ramp vortex evolution and interaction with the downstream flow propagates through the probe profile at $x \approx 1.6$, so the flow physics will be propagated back to the β field in the field inversion process. Within the field inversion time window ($t < 1.0$ s), the augmented SA model matches the SST reference reasonably well, which confirms the success of the field inversion. However, the augmented SA model has noticeable errors when predicting temporally unseen flow in the field inversion, although it captures the overall spatial-temporal distribution of the flow. This is expected because there are important flow physics happens at $t \approx 1.3$ s (the interaction between the ramp vortex and downstream flow). So not formulating the above flow physics in the field inversion objective function may induce prediction error. In summary, we should carefully set probe points that spatially and temporally cover all relevant unsteady flow physics in the field inversion.

Last, we evaluate the capability to predict variables that are not used in the field inversion. To be more specific, we use only the velocity variable to construct the objective function. So the field inversion will optimize the β field to make the SA's velocity prediction match the SST's. In this process, the pressure variable (data) from the SST model is never used. However, one of the advantages of field inversion is that it is solver-in-loop, meaning the entire unsteady CFD solution process is embedded in the field inversion process. Therefore, a corrected velocity field can lead to a corrected pressure field, because the velocity and pressure fields are coupled in the momentum equation used in the field inversion. This allows us to use a small subset of flow variables for augmenting turbulence models. To demonstrate this, we use the field inversion augmented SA model to predict the pressure distribution on the bottom wall and compare it with the SST model, as shown in Fig. 16. The pressure predicted by the velocity-augmented SA model agrees well with the SST reference at $t = 0.7$ and 1.0 s. The prediction accuracy decreases at $t = 1.3$ s because the field inversion ends at $t = 1.0$ s, as mentioned above. Nevertheless, we show that the SA model augmented by using only the velocity data can accurately predict pressure distribution. This is practically useful because not all variables can be easily obtained for challenging flow phenomena, e.g., high Reynolds number flow.

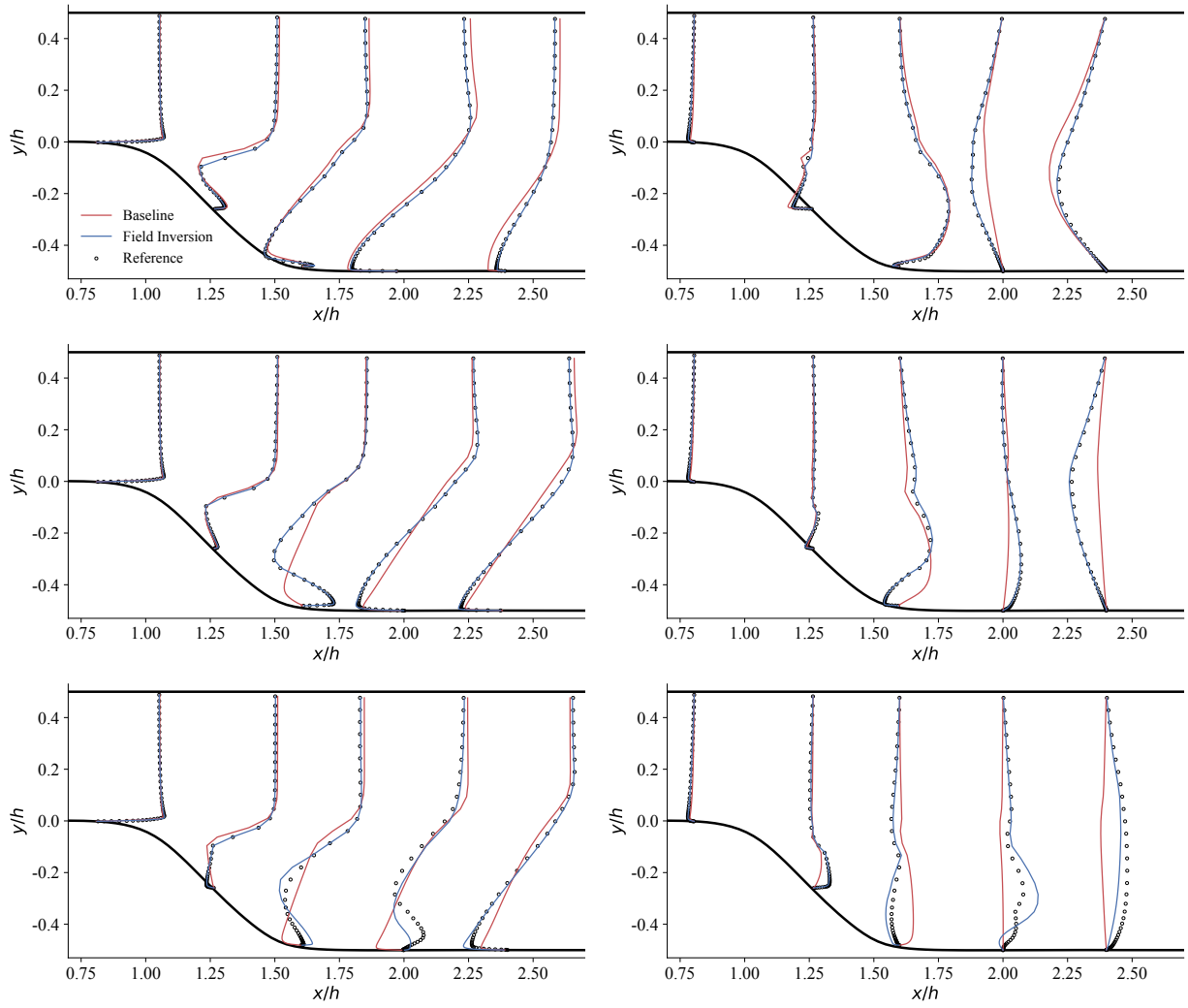

Fig. 14 Velocity profiles (left: U_x and right: U_y) at various streamwise locations (top: $t = 0.7$ s. mid: $t = 1.0$ s. bot: $t = 1.3$ s; Turb-Profile-Field). The inverse velocity agrees reasonably well with the reference before $t = 1$ s when the training ends. There are noticeable errors when predicting unsteady flow for $t > 1$ s (outside of the training window).

IV. Conclusion

In this paper, we develop a segregated time-accurate adjoint solver to enable field inversion of unsteady flow. We solve the unsteady incompressible flow using a segregated PIMPLE algorithm. We then compute unsteady gradients using a new segregated adjoint method. The benefit of using the PIMPLE-adjoint approach is that one can relax the CFL number to be greater than one, reducing the number of adjoint equations to solve. We evaluate the speed and accuracy of the proposed PIMPLE-adjoint method and find that the adjoint is twice as slow as the flow and the gradient computation error is less than 1%, which is acceptable for unsteady field inversion optimization.

Next, we use the proposed PIMPLE-adjoint to demonstrate field inversion optimization for unsteady separated flow over a ramp, which is representative of many unsteady flows in aerospace engineering. We first consider the proper formulation of unsteady field inversion optimization problems using a laminar case. We add a Brinkman term with a Gaussian impermeability field (α) to the momentum equation to change the unsteady vortex evolution over the ramp. Then, we run unsteady field inversion to find the prescribed α field distribution, starting from a zero-α field. We find that using velocity probe point data from a vertical profile downstream of the ramp to formulate the objective function is sufficient for field inversion. This is because this objective function formulation includes all relevant unsteady flow

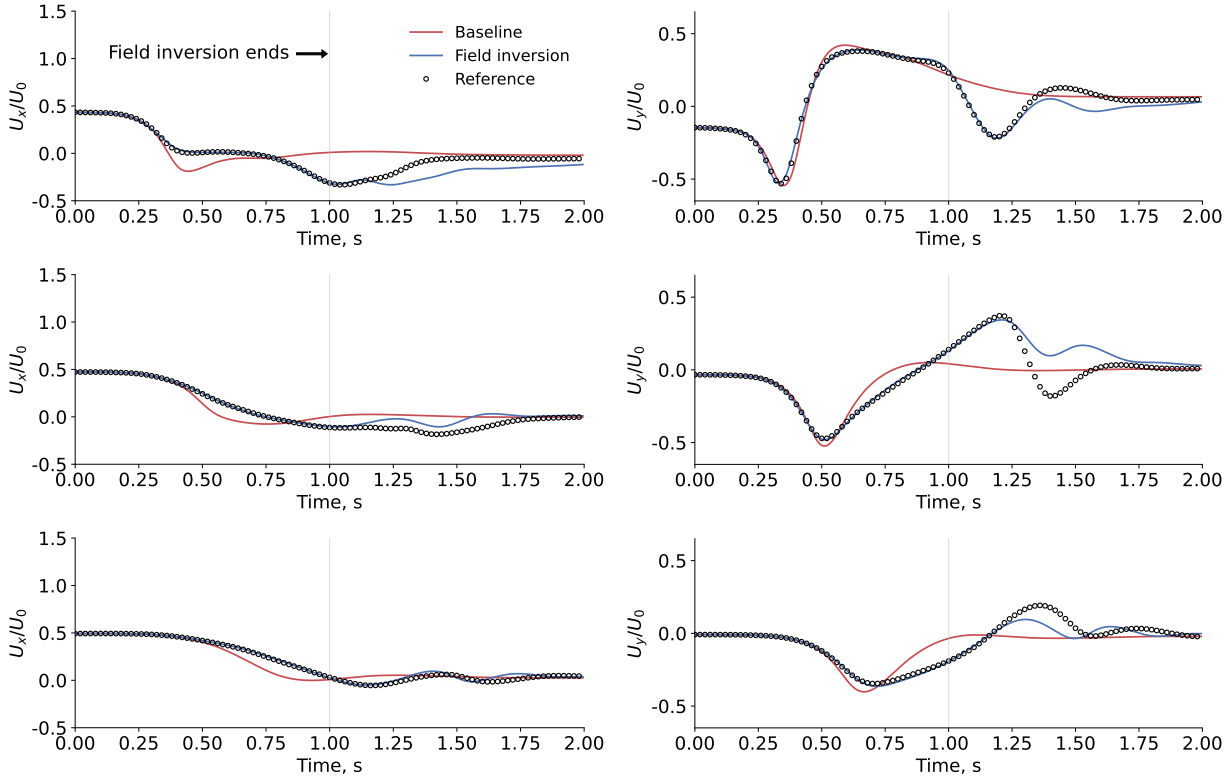

Fig. 15 Velocity time series at various probe points (Top: $[x/h, y/h] = [1.6, -0.2]$, Mid: $[x/h, y/h] = [2.0, -0.2]$, Bot: $[x/h, y/h] = [2.4, -0.2]$. Turb-Profile-Field). The inverse velocity agrees reasonably well with the reference before $t = 1$ s when the training ends (denoted by the gray vertical line). There are noticeable errors when predicting unsteady flow for $t > 1$ s (outside of the training window).

physics. Other options, such as using the velocity time series from a single probe point or setting the probe point upstream of the ramp, do not fully include the unsteady flow physics and their inverse velocity fields have large errors.

With the above problem formulation, we consider a turbulent case and multiply the production term of the Spalart–Allmaras (SA) turbulence model by a scalar field (β). We run unsteady field inversion to optimize the SA model's β field and make its predictions match the spatial-temporal distribution of reference data generated by the $k-\omega$ SST model. We run the field inversion from 0 to 1 s and use the augmented SA model to predict flow from 0 to 2 s. The inverse velocity agrees reasonably well with the reference in $t < 1$ s; however, there are noticeable errors for $t > 1$ s. This result suggests that we need to run field inversion long enough such that all the relevant unsteady flow physics are formulated in the objective function. In addition, we use the velocity-data-augmented SA model to predict the pressure distribution on the bottom field. We find that the inverse pressures field agrees reasonably with the reference. This indicates that the field inversion has the capability to use a small subsection of variables to augment turbulence models so that they can accurately predict other variables not formulated in the objective function.

The proposed method has the potential to train generalizable and accurate turbulence models for predicting challenging unsteady flow in aerospace engineering. We are incorporating the unsteady field inversion into our open-source multidisciplinary design optimization tool, DAFoam [47–50].

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number 2223676.

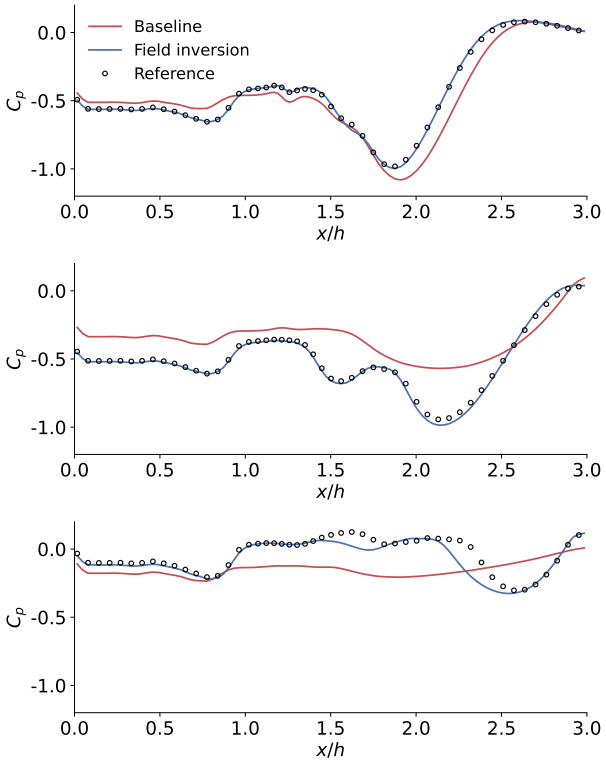

Fig. 16 Pressure distribution on the bottom wall at various times (top: $t = 0.7$ s, mid: $t = 1.0$ s, and bot: $t = 1.3$ s). The inverse pressure agrees reasonably well with the reference before $t = 1$ s. There are noticeable errors when predicting unsteady flow for $t = 1.3$ s (outside of the training window).

References

[1] Spalart, P. R., and Venkatakrishnan, V., "On the role and challenges of CFD in the aerospace industry," *The Aeronautical Journal*, Vol. 120, No. 1223, 2016, pp. 209–232.

[2] Wu, J.-L., Xiao, H., and Paterson, E., "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework," *Physical Review Fluids*, Vol. 3, No. 7, 2018, p. 074602.

[3] Wang, J.-X., Wu, J.-L., and Xiao, H., "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data," *Physical Review Fluids*, Vol. 2, No. 3, 2017, p. 034603.

[4] Weatheritt, J., and Sandberg, R., "A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship," *Journal of Computational Physics*, Vol. 325, 2016, pp. 22–37.

[5] Duraisamy, K., Iaccarino, G., and Xiao, H., "Turbulence modeling in the age of data," *Annual review of fluid mechanics*, Vol. 51, 2019, pp. 357–377.

[6] Brunton, S. L., Noack, B. R., and Koumoutsakos, P., "Machine learning for fluid mechanics," *Annual review of fluid mechanics*, Vol. 52, 2020, pp. 477–508.

[7] Zhu, L., Zhang, W., Kou, J., and Liu, Y., "Machine learning methods for turbulence modeling in subsonic flows around airfoils," *Physics of Fluids*, Vol. 31, No. 1, 2019.

[8] Volpiani, P. S., Meyer, M., Franceschini, L., Dandois, J., Renac, F., Martin, E., Marquet, O., and Sipp, D., "Machine learning-augmented turbulence modeling for RANS simulations of massively separated flows," *Physical Review Fluids*, Vol. 6, No. 6, 2021, p. 064607.

[9] Duraisamy, K., "Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence," *Physical Review Fluids*, Vol. 6, No. 5, 2021, p. 050504.

[10] Parish, E. J., and Duraisamy, K., "A paradigm for data-driven predictive modeling using field inversion and machine learning," *Journal of computational physics*, Vol. 305, 2016, pp. 758–774. Publisher: Elsevier.

[11] Singh, A. P., Medida, S., and Duraisamy, K., "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils," *AIAA journal*, Vol. 55, No. 7, 2017, pp. 2215–2227.

[12] Singh, A. P., and Duraisamy, K., "Using field inversion to quantify functional errors in turbulence closures," *Physics of Fluids*, Vol. 28, No. 4, 2016.

[13] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L., "Physics-informed machine learning," *Nature Reviews Physics*, Vol. 3, No. 6, 2021, pp. 422–440.

[14] Wu, C., and Zhang, Y., "Enhancing the shear-stress-transport turbulence model with symbolic regression: A generalizable and interpretable data-driven approach," *Physical Review Fluids*, Vol. 8, No. 8, 2023, p. 084604. https://doi.org/10.1103/PhysRevFluids.8.084604.

[15] Yang, M., and Xiao, Z., "Improving the k–ω–γ–Ar transition model by the field inversion and machine learning framework," *Physics of Fluids*, Vol. 32, No. 6, 2020.

[16] Ho, J., and West, A., "Field inversion and machine learning for turbulence modelling applied to three-dimensional separated flows," *AIAA aviation 2021 forum*, 2021, p. 2903.

[17] Hafez, A. M., El-Rahman, A., Ahmed, I., and Khater, H. A., "Field inversion for transitional flows using continuous adjoint methods," *Physics of Fluids*, Vol. 34, No. 12, 2022.

[18] Ferrero, A., Iollo, A., and Larocca, F., "Field inversion for data-augmented RANS modelling in turbomachinery flows," *Computers & Fluids*, Vol. 201, 2020, p. 104474.

[19] Bidar, O., He, P., Aderson, S., and Qin, N., "An open-source adjoint-based field inversion tool for data-driven RANS modelling," *AIAA aviation 2022 forum*, 2022.

[20] Bidar, O., He, P., Anderson, S., and Qin, N., "Turbulent Mean Flow Reconstruction Based on Sparse Multi-sensor Data and Adjoint-based Field Inversion," *AIAA AVIATION 2022 Forum*, 2022, p. 3900.

[21] Reuther, J., Jameson, A., Farmer, J., Martinelli, L., and Saunders, D., "Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation," *Proceedings of the 34th AIAA aerospace sciences meeting and exhibit*, Reno, Nevada, 1996.

[22] Kenway, G. K. W., and Martins, J. R. R. A., "Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration," *Journal of Aircraft*, Vol. 51, No. 1, 2014, pp. 144–160. https://doi.org/10.2514/1.C032150.

[23] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "Aerodynamic shape optimization investigations of the Common Research Model wing benchmark," *AIAA Journal*, Vol. 53, No. 4, 2015, pp. 968–985. https://doi.org/10.2514/1.J053318.

[24] Koyuncuoglu, H., and He, P., "Coupled wing-propeller aerodynamic optimization using the adjoint method," *AIAA SCITECH 2022 forum*, 2022, p. 15. https://doi.org/10.2514/6.2022-0015.

[25] Dhert, T., Ashuri, T., and Martins, J. R. R. A., "Aerodynamic shape optimization of WIND TURBINE blades using a Reynolds-averaged Navier–Stokes model and an adjoint method," *Wind Energy*, Vol. 20, No. 5, 2017, pp. 909–926. Publisher: Wiley Online Library.

[26] Madsen, M. H. A., Zahle, F., Sørensen, N. N., and Martins, J. R. R. A., "Multipoint high-fidelity CFD-based aerodynamic shape optimization of a 10 MW wind turbine," *Wind Energy Science*, Vol. 4, 2019, pp. 163–192. https://doi.org/10.5194/wes-4-163-2019.

[27] Wang, D. X., and He, L., "Adjoint aerodynamic design optimization for blades in multistage turbomachines—Part I: Methodology and verification," *Journal of Turbomachinery*, Vol. 132, No. 2, 2010, p. 21011. Publisher: American Society of Mechanical Engineers.

[28] Ma, C., Su, X., and Yuan, X., "An efficient unsteady adjoint optimization system for multistage turbomachinery," *Journal of Turbomachinery*, Vol. 139, No. 1, 2017, p. 11003. Publisher: American Society of Mechanical Engineers.

[29] He, P., Martins, J. R. R. A., Mader, C. A., and Maki, K., "Aerothermal optimization of a ribbed U-Bend cooling channel using the adjoint method," *International Journal of Heat and Mass Transfer*, Vol. 140, 2019, pp. 152–172. https://doi.org/10.1016/j.ijheatmasstransfer.2019.05.075.

[30] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "Aerothermal optimization of internal cooling passages using a discrete adjoint method," *AIAA/ASME joint thermophysics and heat transfer conference*, Atlanta, GA, 2018. https://doi.org/10.2514/6.2018-4080.

[31] Mani, K., and Mavriplis, D. J., "Adjoint-based sensitivity formulation for fully coupled unsteady aeroelasticity problems," *AIAA journal*, Vol. 47, No. 8, 2009, pp. 1902–1915.

[32] 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.

[33] Nielsen, E. J., and Diskin, B., "Discrete adjoint-based design for unsteady turbulent flows on dynamic overset unstructured grids," *AIAA Journal*, Vol. 51, No. 6, 2013, pp. 1355–1373. https://doi.org/10.2514/1.J051859.

[34] Economon, T. D., Palacios, F., and Alonso, J. J., "Unsteady continuous adjoint approach for aerodynamic design on dynamic meshes," *AIAA Journal*, Vol. 53, No. 9, 2015, pp. 2437–2453.

[35] Apponsah, K. P., and Zingg, D. W., "Aerodynamic Shape Optimization for Unsteady Flows: Some Benchmark Problems," *AIAA Scitech 2020 Forum*, 2020, p. 0541.

[36] Kapellos, C. S., Papoutsis-Kiachagias, E. M., Giannakoglou, K. C., and Hartmann, M., "The unsteady continuous adjoint method for minimizing flow-induced sound radiation," *Journal of Computational Physics*, Vol. 392, 2019, pp. 368–384.

[37] Giles, M. B., and Pierce, N. A., "An introduntion to the adjoint approach to design," *Flow, Turbulence and Combustion*, Vol. 65, No. 3, 2000, pp. 393–415. https://doi.org/10.1023/a:1011430410075, iSBN: 1386-6184.

[38] Peter, J. E. V., and Dwight, R. P., "Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches," *Computers and Fluids*, Vol. 39, No. 3, 2010, pp. 373–391. https://doi.org/10.1016/j.compfluid.2009.09.013.

[39] 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.

[40] Sen, A., Towara, M., and Naumann, U., "Discrete Adjoint of an implicit coupled solver based on foam-extend using Algorithmic Differentiation," *16th european workshop on automatic differentiation*, 2014.

[41] 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.

[42] Spalart, P., and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th aerospace sciences meeting and exhibit*, 1992. https://doi.org/10.2514/6.1992-439.

[43] Menter, F. R., Kuntz, M., Langtry, R., et al., "Ten years of industrial experience with the SST turbulence model," *Turbulence, heat and mass transfer*, Vol. 4, No. 1, 2003, pp. 625–632.

[44] Fang, L., and He, P., "A consistent fixed-point discrete adjoint method for segregated Navier–Stokes solvers," *AIAA AVIATION 2022 forum*, 2022, p. 4000.

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

[46] 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. https://doi.org/10.1137/S0036144504446096, publisher: SIAM.

[47] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "DAFoam: An open-source adjoint framework for multidisciplinary design optimization with OpenFOAM," *AIAA Journal*, Vol. 58, No. 3, 2020, pp. 1304–1319. https://doi.org/10.2514/1.J058853.

[48] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "An aerodynamic design optimization framework using a discrete adjoint approach with OpenFOAM," *Computers & Fluids*, Vol. 168, 2018, pp. 285–303. https://doi.org/10.1016/j.compfluid.2018.04.012.

[49] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K., "An object-oriented framework for rapid discrete adjoint development using OpenFOAM," *AIAA scitech 2019 forum*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2019. https://doi.org/10.2514/6.2019-1210.

[50] "DAFoam documentation," , 2022. URL https://dafoam.github.io.