This is a repository copy of *An open-source adjoint-based field inversion tool for data-driven RANS modelling*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/188728/

Version: Accepted Version

Proceedings Paper:
Bidar, O., He, P., Anderson, S. orcid.org/0000-0002-7452-5681 et al. (1 more author) (2022) An open-source adjoint-based field inversion tool for data-driven RANS modelling. In: AIAA AVIATION 2022 Forum. AIAA AVIATION 2022 Forum, 27 Jun - 01 Jul 2022, Chicago, IL, USA (and online). AIAA Aviation Forum Proceedings (2022). American Institute of Aeronautics and Astronautics .

https://doi.org/10.2514/6.2022-4125

© 2022 by Omid Bidar. Published by the American Institute of Aeronautics and Astronautics, Inc. This is an author-produced version of a paper subsequently published in AIAA AVIATION 2022 Forum. Uploaded in accordance with the publisher's self-archiving policy.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

An open-source adjoint-based field inversion tool for data-driven RANS modelling

Omid Bidar*[1], Ping He†[2], Sean Anderson‡[1], and Ning Qin§[1]

[1]*The University of Sheffield, Western Bank, Sheffield, S10 2TN, UK*
[2]*Iowa State University, Ames, Iowa, 50011, USA*

This paper presents an open-source tool for using high-fidelity simulation or experimental data to improve steady Reynolds-averaged Navier-Stokes (RANS) turbulence models. The field inversion approach employed, involves perturbations of the production term in the model transport equation through a spatial field and the iterative optimisation of this field such that the error between model prediction and data is minimised. This highly dimensional inverse problem requires the adjoint method for efficient gradient-based optimisation. It has been successfully applied to reconstruct turbulent mean flows with limited data. However, the implementation is a high barrier to entry as the intrusive development process involves the CFD solver, the adjoint solutions, and the optimiser, making it a time-consuming and laborious task. In this work we integrate open-source codes to enable a flexible framework for field inversion application, open to all interested CFD practitioners. The software capabilities are demonstrated using three flow cases where traditional turbulence models (Spalart-Allmaras and Wilcox $k-\omega$ for this work) perform poorly due to flow separation and adverse pressure gradients. The data used include wind-tunnel experiments and direct numerical simulations, and field inversion scenarios considered integral (e.g. lift coefficient), surface (e.g. skin friction), and volume (e.g. velocity profiles) data, in order of decreasing sparsity.

I. Introduction

Recent years have seen a steady rise in the use of data to augment the shortfalls of Reynolds-averaged Navier-Stokes (RANS) turbulence models in complex flow conditions. Turbulence modelling is required for the Reynolds stress tensor in the RANS equations, and inaccuracies in existing models arise from: model parameters (closure coefficients tuned based on limited set of canonical flows); functional errors due to the mathematical formulation of the turbulence model variables, and structural errors due to simplifying modelling assumptions, such as the Boussinesq approximation [1].

A plethora of ideas based on machine learning and data assimilation techniques have been introduced for improved RANS turbulence modelling. These include: novel machine learning architectures with embedded invariance properties for eddy viscosity Reynolds stress models based on isotropic basis tensors [2, 3]; and formulation of algebraic nonlinear closures using gene expression programming and symbolic regression [4–6]. For comprehensive reviews the reader is referred to the following papers: [1, 7, 8]. Most of the approaches just outlined are known as *a priori* or CFD-free approaches, where the data-driven model is trained directly on inputs and features from high-fidelity data, and the baseline CFD solver is not part of training. This can give rise to inconsistency between the data-driven model and the baseline turbulence model during predictive simulations when the CFD solver becomes part of the process. This issue, along with the detailed high-fidelity data requirement—often difficult to generate—in *a priori* approaches, makes the alternative *model-consistent* techniques particularly appealing.

In model-consistent formulations, a number of inverse problems are solved where the goal is to reduce the error between a turbulence model prediction and data—a process called field inversion. These approaches have also been shown to work well with limited datasets. Broadly, two approaches to inverse modelling has been pursued in parallel: ensemble-based, and adjoint-based methods. Xiao and colleagues have used the ensemble-based Kalman filter (EnKF) approach to model the discrepancy in baseline turbulence models through eigenvalue perturbations [9]. The advantages of this method includes: relatively easy code development, and the ability to compute confidence bounds

*PhD candidate, Dept. of Automatic Control and Systems Engineering, and Dept. of Mechanical Engineering, obidar1@sheffield.ac.uk
†Assistant Professor, Department of Aerospace Engineering, AIAA Senior Member
‡Senior Lecturer, Department of Automatic Control and Systems Engineering
§Professor, Department of Mechanical Engineering, AIAA Associate Fellow, n.qin@sheffield.ac.uk

for the quantity/quantities of interest, albeit, complicated by the need to map the control parameters from a high to low-dimensional space to reduce computational costs.

The alternative adjoint-based method originally proposed by Duraisamy and co-workers relies on solving a gradient-based optimisation problem, where the derivatives of the cost function is computed using the adjoint approach. A number of different variants of this method has been studied: perturbations of the Reynolds stress anisotropy eigenvalues [10]; perturbations of the eigenvalues as well as the eigenvectors of the anisotropy tensor [11]; and modification of the turbulent model transport equation through a spatial scalar field defined over the entire flow domain [12, 13]. In this work, we use the latter since it has proven to be simple and computationally cheap relative to the other variants. Besides, the formulations which directly address the Reynolds stress tensor have not been demonstrated to be have a far superior improvements in the Reynolds stress predictions [8].

The adjoint-method is, in principle, capable of recovering finer scales of turbulent mean flow compared to the ensemble-based method. However, a fully Bayesian formulation with this method is computationally expensive [12], and most researchers have opted for deterministic formulations. Another limitation of the adjoint-based field inversion is the time-consuming and laborious software development process due to the intrusive nature of adjoint-based optimisation, which is a high barrier to entry for many researchers. In this work we aim to tackle this problem by introducing an open-source tool which interested practitioners can use to study and apply the adjoint-based field inversion method. Besides, the two field inversion approaches have not been systematically compared against consistent metrics (e.g. data requirement, computational cost, etc.) largely due to the challenge of software implementation. The recent introduction of DAFI [14], an open-source tool for ensemble-based field inversion, and the outcome of present work should reduce the implementation barrier, and encourage a comprehensive study of the two approaches.

In terms of the adjoint implementation there are two approaches: continuous and discrete adjoint. He et al. [15] have applied the field inversion framework to the Spalart-Allmaras model using a continuous adjoint implementation. In this framework, the adjoint equations for the governing equations are derived in the continuous form, and then discretised for numerical solutions. They achieved promising results on a number of cases that included a three-dimensional wall-mounted cube. The continuous adjoint method has the advantages of low computational cost due to lower memory requirement and simpler to implement in existing CFD codes. However, derivations of the adjoint equations are complex, and has to be repeated for every new turbulence models, boundary conditions, and objective function formulations (depending on what quantity is used from data) [16]. To avoid these, we use the discrete-adjoint method in this work.

In the discrete approach the adjoint equations are derived for the discretised governing equation from the outset. This method has the limitation of large memory requirements. However, the advantages are: ability to achieve more accurate gradient information since these are consistent with the discretised objective function evaluations; and the ability to use algorithmic differentiation which does not require an updated adjoint equation derivation for every new model, boundary condition, or objective function formulation. The preliminary implementation is available on Github at: `https://github.com/obidar/dafoam`.

The remainder of the paper is structured as follows: in Section II we formulate field inversion; summarise the processes involved in implementing the chosen method; and briefly outline the open-source packages we integrate to enable the application of field inversion: OpenFOAM for CFD solver, DAFoam for discrete-adjoint solutions, and pyOptSparse for optimisation. In Section III, results for three complex flow cases—a wind-turbine airfoil at a high incidence, flow over a converging-diverging channel with incipient separation, and a highly separated periodic hill flow. The Spalart-Allmaras (S-A) model is use in the former two cases, while for the periodic hill flow both S-A and $k-\omega$ models are employed. Finally, some conclusions are drawn in Section IV.

II. Methods

A. Field inversion formulation

To perform field inversion, a multiplicative scalar field is introduced in the production term of the transport equation of an existing turbulence model. For the one-equation Spalart-Allmaras model [17], for instance, the general form of the model transport equation for the surrogate variable, $\tilde{\nu}$, is modified as follows:

$$\frac{D\tilde{\nu}}{Dt} = \beta\left(\boldsymbol{x}, t\right)\mathcal{P}\left(\tilde{\nu}, \boldsymbol{w}\right) + \mathcal{T}\left(\tilde{\nu}, \boldsymbol{w}\right) - \mathcal{D}\left(\tilde{\nu}, \boldsymbol{w}\right), \tag{1}$$

where $\mathcal{P}$, $\mathcal{T}$, and $\mathcal{D}$ are the production, transport, and dissipation terms of the transport equation respectively, and are functions of the surrogate viscosity variable $\tilde{\nu}$ and $\boldsymbol{w}$, which represents all the Reynolds-averaged conserved flow

variables. $\beta(\boldsymbol{x}, t) \in \mathbb{R}^{n_\beta}$ with n_β representing the number of mesh cells, is the discrepancy field, and $\beta = 1$ everywhere in the mesh recovers the baseline model.

In case of turbulence models with multiple transport equations, such as the Wilcox $k-\omega$ [18], one or all the transport equations can be modified using the same approach described above. In this work, we modify the $k-\omega$ model by perturbing the dissipation rate (ω) transport equations, by rewriting it as $\beta \cdot \mathcal{P}(\omega, \boldsymbol{w})$.

The optimum discrepancy field, which reduces the functional error in the baseline turbulence model, can be found by reducing the error between high-fidelity data, $\boldsymbol{d} \in \mathbb{R}^{N_d}$ where N_d is the number of data points, and the RANS output, $\mathcal{G}(\beta)$, by minimising an objective function of the following form:

$$\min_{\beta} \quad \mathcal{J} = \|\mathcal{G}(\beta) - \boldsymbol{d}\|_2^2 + \lambda \|\beta - \beta_{\text{prior}}\|_2^2, \tag{2}$$

where $\|\cdot\|_2$ is the $L2$ norm, λ is a relaxation or regularisation parameter, and β_{prior} is typically assumed to be 1, to bias the solution closer the baseline model to avoid an ill-posed optimisation problem.

B. Field inversion implementation

A high-level flow chart for the field inversion process is shown in Fig. 1. The process involves:

1) solution of the governing equations, including the baseline turbulence model using OpenFOAM,
2) using these results to compute the objective function (Eqn. 2),
3) computing the derivative of the objective function with respect to the design variable, (β), using DAFoam,
4) using an optimiser to update the β field such that the least-squares difference between model predictions and data is minimised, using the optimiser suite pyOptSparse,
5) and repeating steps 1-4 until a user-specified optimisation convergence criterion has been met.

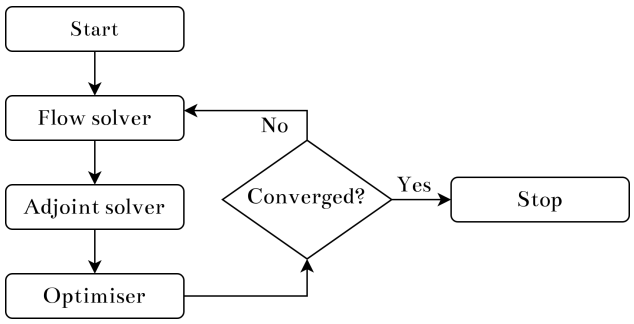


Fig. 1 The flow diagram of the iterative field inversion process. At the start $\beta = 1$, and the goal is to obtain the optimum β that minimises the difference baseline model predictions and data.

Note that before the above iterative optimisation can be carried out, a) the baseline turbulence model must be modified, as described in the previous section, b) the codes for objective function calculation must be implemented, and c) the high-fidelity data must be prepared for use during optimisation. The first two steps are one-time implementations and can be reused for all future flow cases, while the third step is specific to the particular flow of interest and the available data. The following sections briefly outline the open-source software integrated to perform field inversion.

C. CFD flow solver—OpenFOAM

Open field operation and manipulation (OpenFOAM) is popular, open-source CFD package, based on the finite volume method and written in C++, with an active developer and user base [19]. In this work, we have employed two of its main solvers: `simpleFoam`, used for the solution of the steady Navier-Stokes equations for incompressible fluids, and `rhoSimpleFoam` which is a compressible steady-state solver. Both solvers, use the semi-implicit method for pressure-linked equations (SIMPLE) algorithm to solve the coupled continuity and momentum equations.

Many of the popular turbulence models are available on OpenFOAM, and the modified Spalart-Allmaras and $k-\omega$ turbulence models for field inversion, are based on the original OpenFOAM implementation. The field inversion models require minimal code modification, and extensions to other turbulence models do not require deep C++ programming skills.

D. Discrete adjoint solver—DAFoam

Discrete adjoint-Foam (DAFoam) is another open-source package, that allows effective adjoint solutions. It has been specifically tailored for OpenFOAM and has been successfully applied for multi-disciplinary design optimisation [20]. Before outlining the adjoint solution procedure, we first derive the discrete adjoint equations below.

Given a set of discretised governing equations, $\mathcal{R}(\beta, \boldsymbol{w}) = 0$, where $\beta \in \mathbb{R}^{n_\beta}$ is the vector of design variables, $\boldsymbol{w} \in \mathbb{R}^{n_w}$ is the vector of state variables, and $\mathcal{R} \in \mathbb{R}^{n_w}$ is the residuals vector, and the cost function $\mathcal{J} = f(\beta, \boldsymbol{w})$, the goal in discrete adjoint method is to efficiently compute the derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\beta$. The discrete adjoint equations will be derived in this section, following Kenway et al. [16].

The derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\beta$ can be expressed as,

$$\underbrace{\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\beta}}_{1\times n_\beta} = \underbrace{\frac{\partial \mathcal{J}}{\partial \beta}}_{1\times n_\beta} + \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}}}_{1\times n_w} \underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\beta}}_{n_w \times n_\beta}, \tag{3}$$

where the chain rule has been applied. The partial derivatives $\partial\mathcal{J}/\partial\beta$ and $\partial\mathcal{J}/\partial\boldsymbol{w}$ are computationally less expensive to calculate since these require explicit computations only. On the other hand, the total derivative matrix $\mathrm{d}\boldsymbol{w}/\mathrm{d}\beta$ requires implicit treatment through the residual equations, $\mathcal{R}(\beta, \boldsymbol{w}) = 0$, making it computationally expensive.

By applying the chain rule to the residual equations $\boldsymbol{\mathcal{R}}$,

$$\frac{\mathrm{d}\boldsymbol{\mathcal{R}}}{\mathrm{d}\beta} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \beta} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} \frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{\beta}} = 0, \tag{4}$$

and noting that $\mathrm{d}\boldsymbol{\mathcal{R}}/\mathrm{d}\beta$ must equal zero in order for $\boldsymbol{\mathcal{R}}(\beta, \boldsymbol{w}) = 0$ to hold, the total derivative $\mathrm{d}\boldsymbol{w}/\mathrm{d}\beta$ can be expressed as the following linear system

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\beta} = -\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}}^{-1} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \beta}. \tag{5}$$

Substituting the expression for $\dfrac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\beta}$ above, into Eqn. 4 leads to

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\beta} = \frac{\partial \mathcal{J}}{\partial \beta} - \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}}^{-1}}_{\boldsymbol{\psi}^T} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \beta}, \tag{6}$$

where $\boldsymbol{\psi}$ is the adjoint vector. Eqn. 6 can be manipulated further to achieve the *adjoint equations*,

$$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} \boldsymbol{\psi} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{w}}^T. \tag{7}$$

After solving Eqn. 7, the total derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\beta$ is computed by substituting the adjoint vector $\boldsymbol{\psi}$ into Eqn. 4, leading to the following final relation:

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\beta} = \frac{\partial \mathcal{J}}{\partial \beta} - \boldsymbol{\psi}^T \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \beta}. \tag{8}$$

The extended design structure matrix (XDSM) diagram in Fig. 2 outlines the processes and data flow for the discrete adjoint approach in DAFoam. In particular, we implement the discrete adjoint equations using the Jacobian free adjoint approach, as detailed in Kenway et al. [16]. The partial derivatives (e.g., $\partial\mathcal{J}/\partial\beta$) and the matrix-vector products (e.g., $[\partial\boldsymbol{\mathcal{R}}/\partial\beta]^T\boldsymbol{\psi}$) are computed using the revere-mode automatic-differentiation.

E. Large-scale optimiser—pyOptSparse

Once the total derivative $\partial\mathcal{J}/\partial\beta$ is computed, we use the pyOptSparse package to find the optimal β field that minimises the objective function. pyOptSparse is a object-oriented Python interface of various optimisers which can be used for formulating and solving constrained nonlinear optimisation [21]. It allows efficient handling of large-scale optimisations through the use of sparse matrices in the code. For this work, we have successfully employed two of the large-scale optimisation packages available in pyOptSparse: SNOPT (based on sequential quadratic programming, and requires paid license), and IPOPT (based on a primal-dual interior point method, with open-source code).

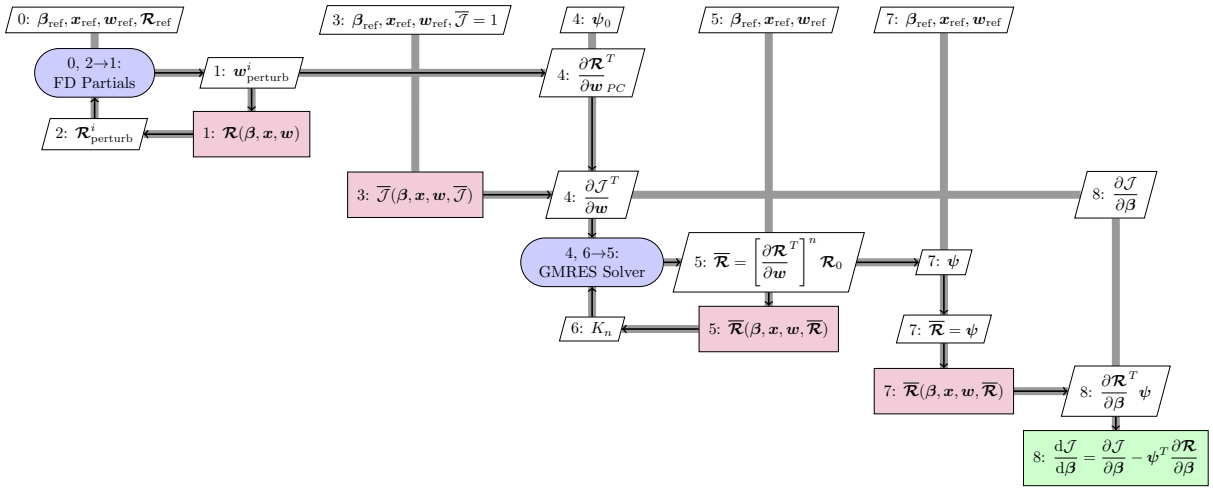


Fig. 2 Jacobian free adjoint XDSM diagram. The modules are represented by the diagonal nodes, while the off-diagonal nodes represent data. The process flow is represented by the black lines, while the thick grey lines are showing data flow. The execution order is illustrated by the number in each node.

F. Python user interface

A high-level Python layer is used to set and run the field inversion simulation process outlined in Fig. 1. We plan to develop detailed tutorials on how to use and extend the tool in the near future. Specifically, the following parameters are set in a Python script: primal flow solver (i.e. `simpleFoam`); flow solver boundary conditions and residuals convergence tolerance; the field inversion objective function specialisation and the relevant parameters (so far, the following have been implemented: full fields, velocity profiles, surface pressure and skin friction, and aerodynamic force coefficients) and the regularisation constant, λ in Eqn. 2; adjoint solver parameters such as state normalisation constants, and the equation solution options; and finally the optimiser and its parameters such as β field constraints (upper and lower bounds), convergence tolerance, maximum number of iterations, etc.

III. Results and Discussions

In this section we demonstrate field inversion results on a number of cases, summarised in Table 1. The cases are the NREL S809 wind turbine airfoil at a high incidence (Fig. 3) [22], flow through a converging-diverging channel (Fig. 8) [23, 24], and flow over a periodic hill (Fig. 15) [25]. These were chosen based on data availability, and to demonstrate likely field inversion scenarios based on types of data used for flow reconstruction. Generally, three types of data sources can be considered: *integral data* (e.g. lift or drag coefficients), *surface data* (e.g. surface pressure, or skin friction), and *volume data* (e.g. velocity fields/profiles at certain locations). We consider all three scenarios in the following sections.

Table 1 Summary of case setups, where C_l, C_p, C_f, and U_x are the lift coefficient, pressure coefficient, skin friction coefficient, and the streamwise velocity, respectively. Dimension column refers to the size of the data used.

Case	Geometry	Data	Definition	Data size	Data source	Turbulence model	Re
1a	S809 Airfoil	C_l	$L/(p_{\text{dyn}}A)$	1	Integral	Spalart-Allmaras	2×10^6
1b		C_p	$(p - p_\infty)/p_{\text{dyn}}$	32	Surface		
2a	Conv.-Div. Channel	C_f	τ_w/p_{dyn}	564	Surface	Spalart-Allmaras	12,600
2b		U_x	-	98,700	Volume		
3a	Periodic Hill	U_x	-	447	Volume	Wilcox $k-\omega$	5,600
3b		U_x	-	447	Volume	Spalart-Allmaras	

A. Airfoil flow at high incidence

Existing turbulence models are known to perform poorly in predicting the flow over airfoils at high incidence angles, where the flow generally separates. We utilise field inversion to improve the prediction of separated flow over the NREL S809 horizontal-axis wind turbine section, also investigated in [11, 13]. Experimental studies by Somers [22] found that at high angles-of-attack ($\alpha \gtrapprox 10^\circ$) the flow separates near mid-chord. We take the flow at $\alpha = 14.24^\circ$ as a test case, where the available experimental data include surface pressure C_p, and lift coefficient C_l, at the following flow conditions: Reynolds number based on chord length, $Re_c = 2 \times 10^6$, and freestream Mach number, $M_\infty = 0.2$.

Since the Mach number is relatively low, the flow can be assumed to be incompressible. However, to demonstrate the capabilities of the developed tool, we employ both an incompressible (`simpleFOAM`) and a compressible solver (`rhoSimpleFOAM`). The latter case solves an energy equation along with the Navier-Stokes equations. The flow is assumed to be two-dimensional and steady, and the Spalart-Allmaras model is used as the baseline turbulence model. The simulations use an structured O-grid mesh, Fig. 3, with an average non-dimensional wall distance, $y^+ < 1$ on the airfoil. The relatively dense mesh is used to reduce mesh-related inaccuracies.

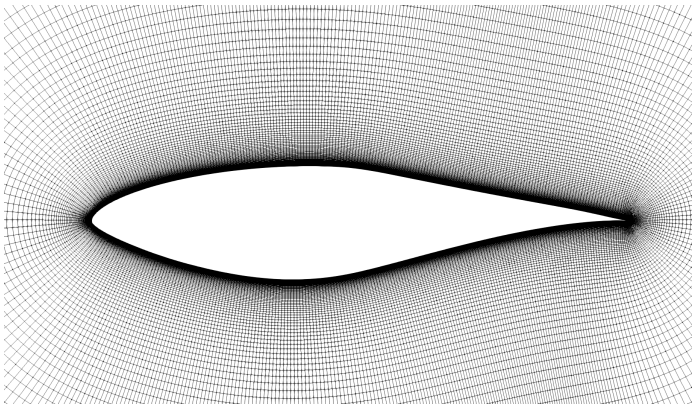


Fig. 3 Close-up of the mesh for the S809 airfoil, with around 8.9×10^5 cells.

As summarised in Table 1, two types of data are considered for field inversion: lift coefficient C_l and surface pressure coefficient C_p. The C_p data is extracted from [22], and only the values on the suction-side are used for field inversion—this is the region most prone to inaccurate predictions by the baseline model. The regularisation constant λ in Eqn. 2 is set to 10^{-4} following [13].

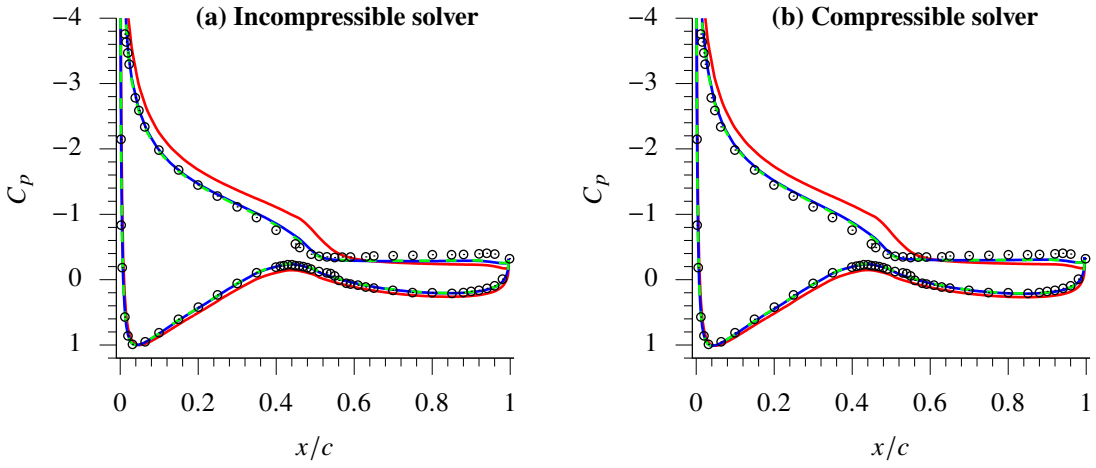
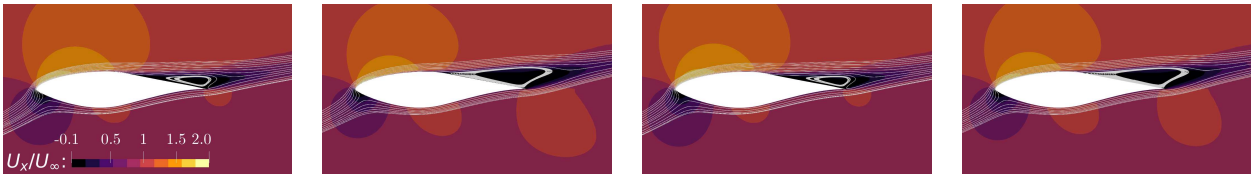


Fig. 4 Comparison of the pressure distribution for S809 airfoil. Legend: Experiment (○), Spalart-Allmaras (——), field inversion, C_l data (——), and field inversion, C_p data (- - -).

Table 2 Comparison of lift-coefficient prediction. Experimental $C_l = 1.083$.

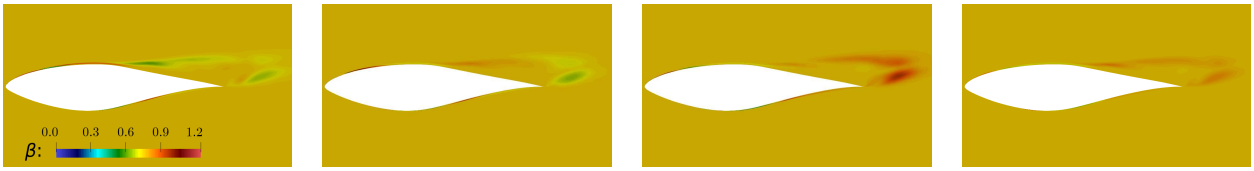
Scenario	Incompressible	Error	Compressible	Error
Baseline Spalart-Allmaras	1.310	20.9%	1.346	24.3%
Field inversion, C_l data	1.107	2.2%	1.145	5.7%
Field inversion, C_p data	1.093	0.9%	1.133	4.6%

All field inversion scenarios result in significant error reduction in the C_l and C_p predictions, as shown in Table 2 and Fig. 4, respectively. The use of surface pressure data results in slightly better improvement of the baseline results, compared to only lift-coefficient value. This is expected due to the significant difference between the size of data used for field inversion, as outline in Table 1. Both the compressible and incompressible solvers produce similar results in terms of the surface pressure distribution, lift coefficient, and velocity fields.



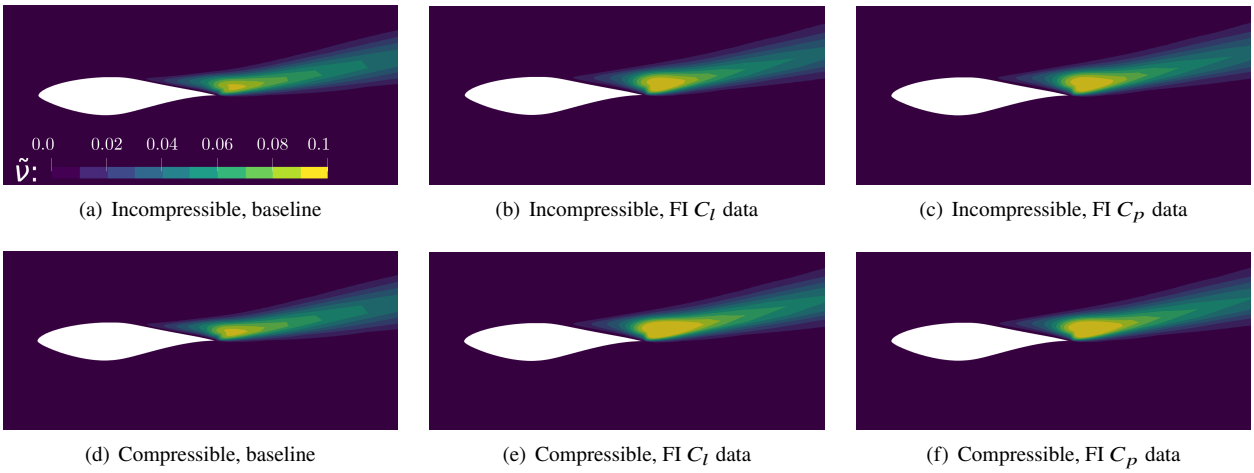
(a) Incompressible, baseline (b) Incompressible, FI C_p data (c) Compressible, baseline (d) Compressible, FI C_l data

Fig. 5 Comparison of the streamwise velocity field along with streamlines for the different S809 simulations. The velocity fields for incompressible FI using C_l data and compressible FI using C_p data are very similar to two field inversion velocity predictions shown here, thus removed for brevity.



(a) Incompressible, C_l data (b) Incompressible, C_p data (c) Compressible, C_l data (d) Compressible, C_p data

Fig. 6 Comparison of the corrective field, β, for the different S809 cases.



(a) Incompressible, baseline (b) Incompressible, FI C_l data (c) Incompressible, FI C_p data

(d) Compressible, baseline (e) Compressible, FI C_l data (f) Compressible, FI C_p data

Fig. 7 Comparison of the surrogate turbulence variable $\tilde{\nu}$ in the S-A model before and after modification by the corrective scalar field β, shown in Fig. 6.

The baseline model over-predicts the lift generated, which is also observed in the over-prediction of the pressure on the suction side. The baseline model also under-predicts the flow separation location and the size of the separation

bubble, as shown in the velocity field contours in Fig. 5. The corrective fields, β, shown in Fig. 6, account for the errors in the baseline model by reducing the turbulent production (i.e. regions with $\beta < 1$), and hence predicting an earlier separation, and a larger separation bubble.

Most significant changes made by the β field for the different scenarios are in the boundary layer close to the airfoil. Additionally, it is interesting to note that the relatively different β field distributions shown in Fig. 6 lead to similar distribution of the surrogate turbulence variable $\tilde{\nu}$ (the quantity that is directly modified, as described in Section II.A) and the velocity field, as shown in Fig. 7 and Fig. 5, respectively. A similar observation was made by He et al. [15] who argue that this might be due to the eddy viscosity hypothesis, which assumes that the Reynolds stress tensor can be modelled using a scalar in the form of eddy viscosity. It is worth reiterating that the corrective field changes the entire balance of the turbulence model transport equation, which may explain the multi-optimal nature of the optimisation results in this, and following cases.

B. Mildly separated flow in a converging-diverging channel

The next case is the flow over a smooth converging-diverging channel, Fig. 8. The flow involves adverse pressure gradients and a small separation bubble on the curved region of the lower wall which cannot be predicted accurately by the S-A model. Of the cases investigated in this work, the converging-diverging channel has the richest dataset available based on direct numerical simulation (DNS) results of Laval et al. [23]. We consider using two types of data to perform field inversion: the entire streamwise velocity field (interpolated from the original dense mesh to a coarse RANS mesh), and the skin friction distribution on the lower wall, as summarised in Table 1.

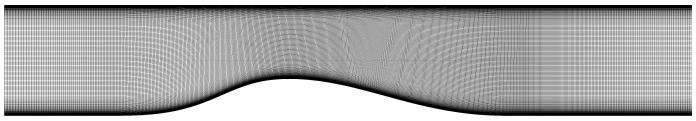


Fig. 8 Mesh for the converging-divering channel, with 9.87×10^5 cells.

We use the two-dimensional, steady, incompressible Navier-Stokes equations and the Spalart-Allmaras model to simulate the flow. The Reynolds number based on the channel half-height and maximum inlet velocity is 12,600. The structured mesh used for the simulations are from [24], with an average $y^+ < 0.2$. Using the same approach as McConkey et al. [24] the inlet boundary conditions are generated by simulating a fully-developed boundary layer using the same turbulence model, and Reynolds number. The regularisation constant λ is set to 10^{-6}—a small value to reflect high-confidence in the available low-noise data.

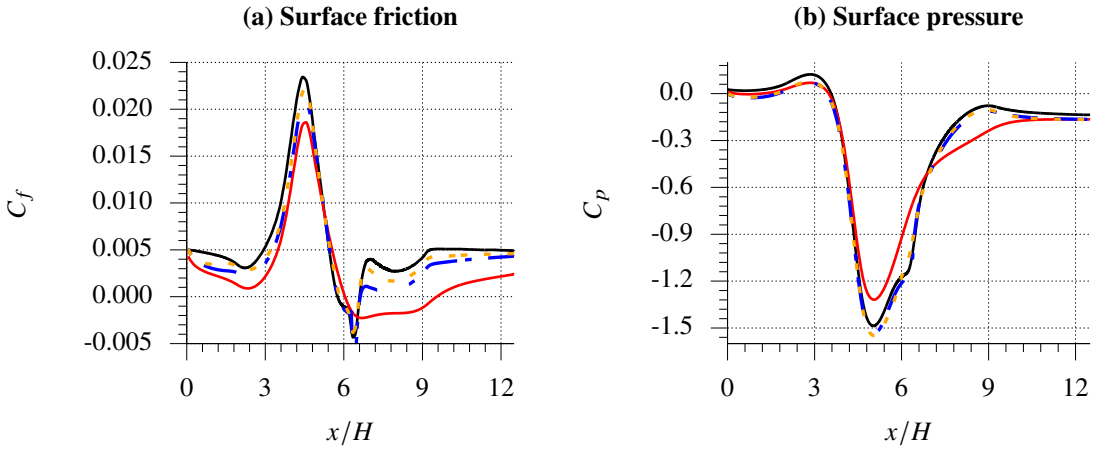


Fig. 9 Comparison of the lower wall surface pressure and surface friction for the converging-diverging channel. Legend: DNS (——), Spalart-Allmaras (——), FI, U_x field data (- - -), and FI, C_f data (- - -).

Fig. 9 shows that field inversion can significantly improve the skin friction and surface pressure distributions on the lower wall. The skin friction distribution for the scenario where C_f data is used is better than the case where the entire

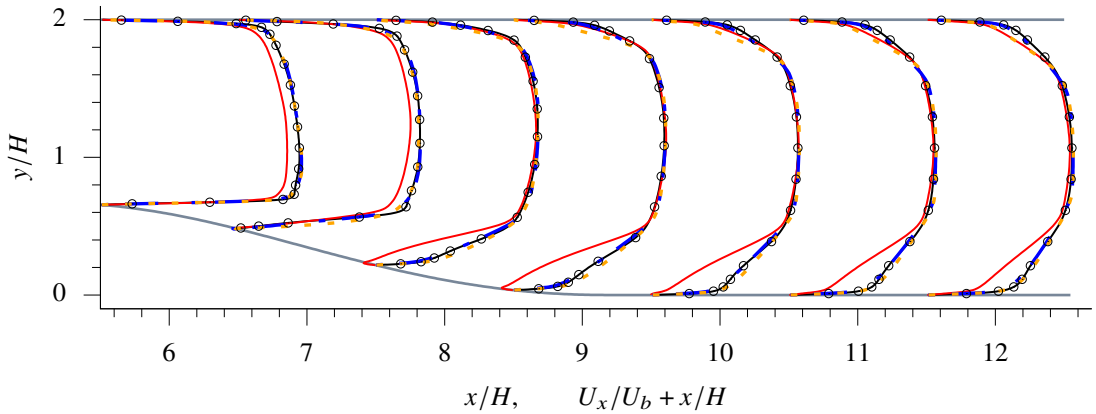


Fig. 10 Comparison of streamwise velocity profiles for the converging-diverging channel. Legend: DNS (—/○), Spalart-Allmaras (—), field inversion, U_x field data (- - -), field inversion, C_f data (- - -), and channel outline (—).

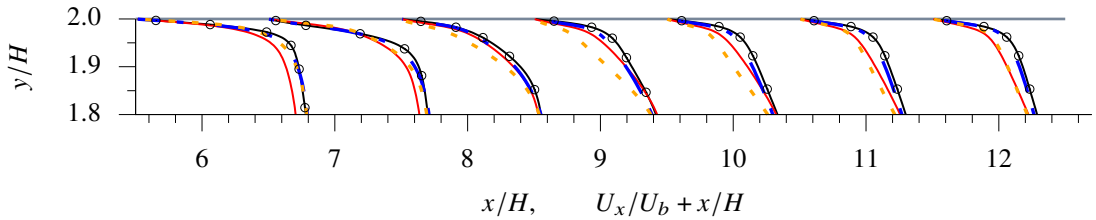


Fig. 11 Close-up of streamwise velocity profiles for the converging-diverging channel near the top wall. For legend see Fig.10.

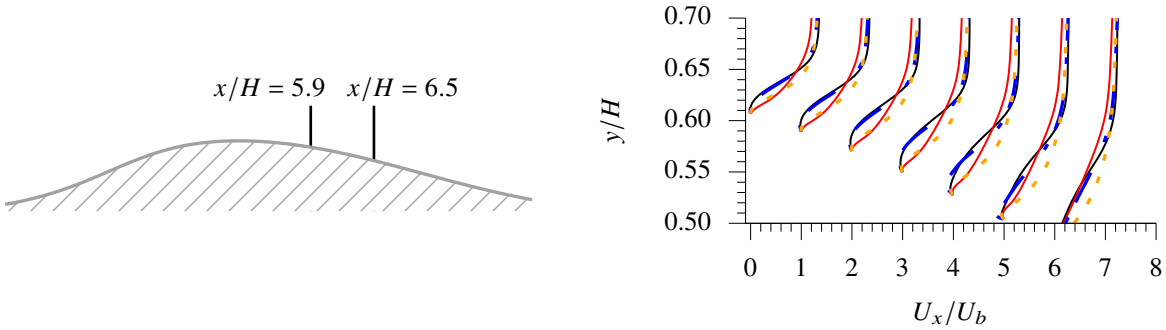


Fig. 12 Close-up of streamwise velocity profiles for the converging-diverging channel in the separation region, $5.9 \leq x/H \leq 6.5$, plotted every $x/H = 0.1$. For legend see Fig.10. Note: beginning from the profile at x/H, subsequent profiles have been shifted by 1, horizontally.

streamwise velocity field data is used. The baseline model massively over-predicts the size of the separation bubble, and the separation and reattachment locations. This is is also clear in the velocity profiles shown in Fig. 10 where the baseline model predicts a very large recirculation region downstream of the hill.

Fig. 10 shows that using just the lower wall skin friction distribution results in a huge overall improvement of the velocity field. However, closer inspection reveals some discrepancies in the boundary layer close to the upper wall, shown in Fig. 11. Similarly, the velocity profiles in the separation bubble, Fig. 12, shows that field inversion performed with C_f data does not necessarily improve the velocity prediction in this region. As expected, using the velocity field data for reconstruction leads to accurate velocity predictions in the separation bubble too.

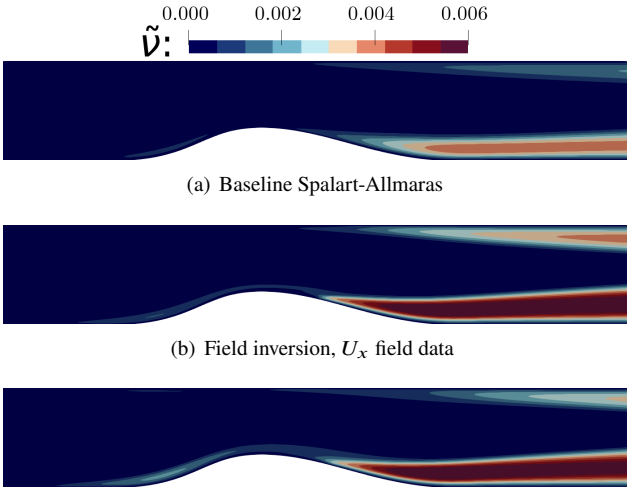
Fig. 13 shows two very different corrective fields for the field inversion scenarios. For the case with velocity field

(a) Field inversion, U_x velocity field data

(b) Field inversion, C_f skin friction data

Fig. 13 Comparison of the corrective field, β, for the two converging-diverging channel field inversion scenarios.

as data, the corrective field makes considerable modifications everywhere in the domain by generally increasing $\tilde{\nu}$ production away from the hill wall, while a complex combination of $\tilde{\nu}$ production damping and magnifications near the hill wall (especially, in and around the separation bubble). For the field inversion case with C_f data, the significant modifications are concentrated near the hill (with regions away from the wall with β values close to 1), and as the other case, includes both damping and magnifying $\tilde{\nu}$ production.



(a) Baseline Spalart-Allmaras

(b) Field inversion, U_x field data

(c) Field inversion, C_f skin friction data

Fig. 14 Comparison of the surrogate turbulence variable $\tilde{\nu}$ in the S-A model before and after modification by the corrective field β, shown in Fig. 13.

As in the airfoil case, the corrective fields should be interpreted with reference to the turbulence model variable $\tilde{\nu}$ shown in Fig. 14. It is clear that both field inversion cases significantly increases $\tilde{\nu}$ and thus the turbulence eddy viscosity downstream of the hill, and to some extend at the upper wall (towards the outlet). For the reconstruction case with U_x field data, the turbulence levels next to the upper wall (downstream region) is higher than the baseline and field inversion case with C_f data, which allows it to better fit the DNS velocity in that region, as previously shown in Fig. 11.

C. Fully separated flow over periodic hill

The last case, is the periodic hill geometry, which has become a prototypical case for testing turbulence models. Most linear eddy-viscosity based RANS models are known to perform poorly in predicting the flow with large separation after the initial hill. In this work we use the DNS dataset supplied by Xiao et al. [25].

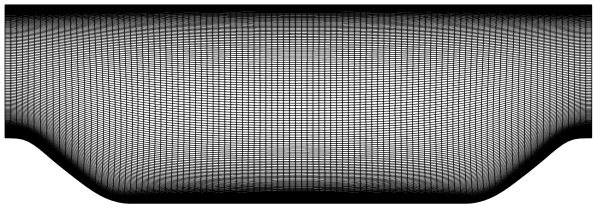


Fig. 15 The mesh used for the periodic hills case supplied with the dataset, with $\sim 1.4 \times 10^4$ cells.

We use the streamwise velocity profiles at $(x/H = 0, 3, 6)$ for field inversion. The Reynolds number is set to 5,600 following the data, and the two-dimensional, incompressible, Navier-Stokes equations are solved. A source term is added to the momentum equation to achieved a set bulk velocity. No-slip boundary conditions are applied at the walls, and cyclic boundary conditions are set at the inlet and outlet. The structured mesh has an average $y^+ < 1$ on walls.

In order to demonstrate the capability of field inversion and the software to work for different turbulence models, we report the results for the modified Wilcox $k - \omega$, as well as the modified Spalart-Allmaras model used in the previous two cases.

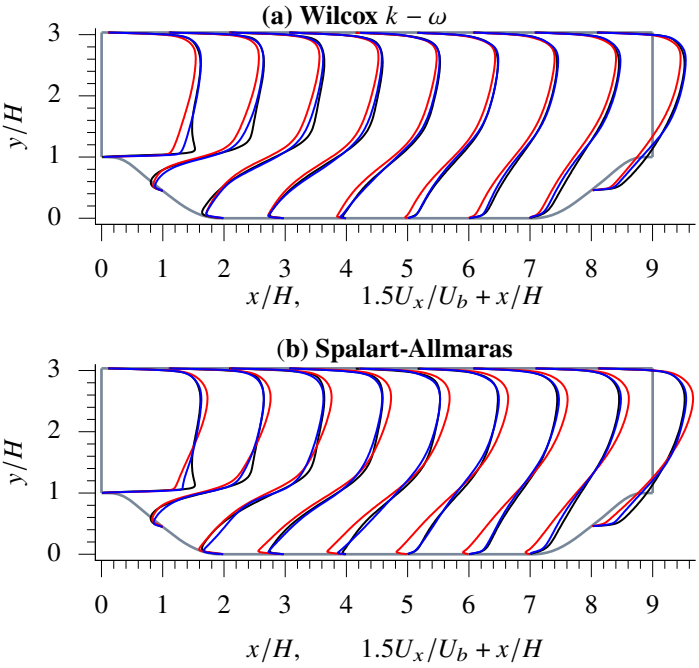
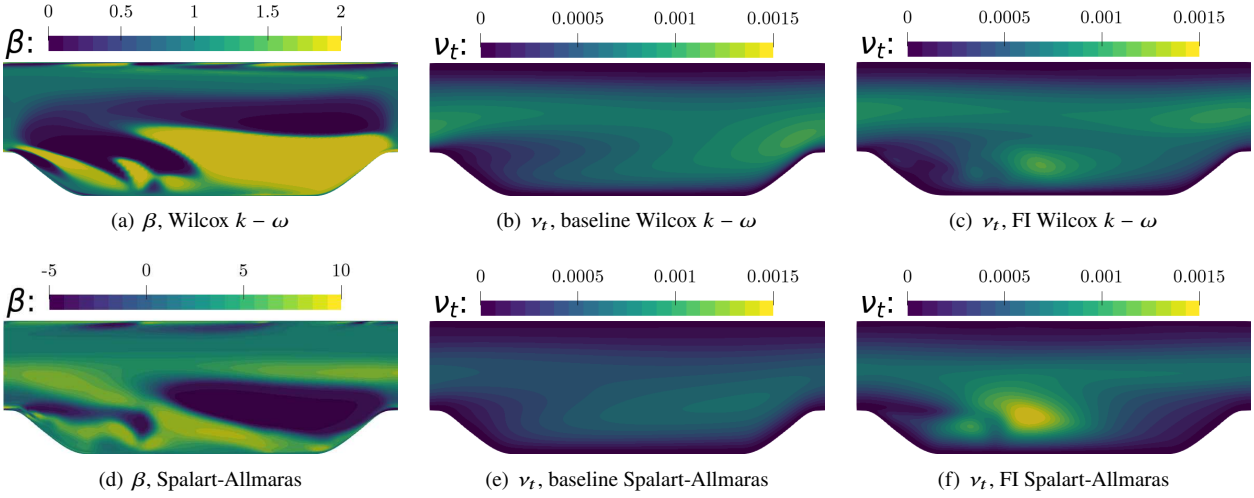


Fig. 16 Periodic hill streamwise velocity profiles. Legend: baseline turbulence model (—), field inversion (—), DNS(—) and periodic hill outline (—).



(a) β, Wilcox $k - \omega$ (b) ν_t, baseline Wilcox $k - \omega$ (c) ν_t, FI Wilcox $k - \omega$

(d) β, Spalart-Allmaras (e) ν_t, baseline Spalart-Allmaras (f) ν_t, FI Spalart-Allmaras

Fig. 17 Comparison of the corrective field, β, applied to the transport equations of the two RANS models.

The velocity profiles at nine stations, shown in Fig. 16, illustrate that the baseline $k - \omega$ model provides a more accurate velocity predictions compared to the baseline S-A model. However, both models over-predict the

separation region after the first hill, and thus the follow attaches too late. For many RANS models this is attributed to the under-predicted eddy-viscosity in the shear layer. Field inversion seems to account for this through a complex combination of eddy viscosity magnification and dampening in the domain, shown in the β fields in Fig. 17 (a) and (d). Clearly, more significant changes are made in the S-A model, including regions with $\beta < 0$ where eddy viscosity is decreased. Although, the baseline eddy-viscosity for the two models are entirely different (Fig. 17 b and e)—expected due to wholly different model structures—the eddy-viscosity predicted after field inversion is qualitatively similar in both cases (Fig. 17 c and f), although more intense regions are shown in the case of S-A model.

D. Verification of adjoint derivatives

The accuracy for adjoint derivative computation is critical to the field inversion robustness. Inaccurate derivative may abort the optimisation and result in sub-optimal β field. In this subsection, we verify the adjoint accuracy by comparing the adjoint derivative with the forward-mode automatic differentiation (AD). The forward-mode AD differentiates the entire CFD code in the forward direction, so it is commonly used as the reference values for adjoint derivative verification[16].

We use the periodic hill case as the benchmark, and both the Spalart–Allmaras and $k-\omega$ turbulence models are considered. We first run the adjoint solver to compute $\mathrm{d}\mathcal{J}/\mathrm{d}\beta$ for all β field. We then select ten cells in the flow fields, shown in Fig. 18. Then, we set the AD seed for each of these ten cells and run tens CFD simulations to compute the forward-mode AD derivatives. The adjoint derivatives agree reasonably well with the forward-mode AD derivatives. The averaged error is less than 0.01%.

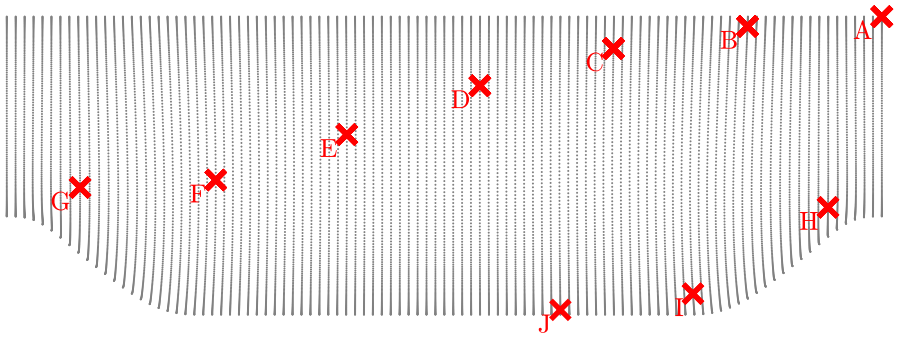


Fig. 18 Locations of cells for adjoint verification—starting from the first cell, every 1500th cell is chosen out of a total $\sim 1.5 \times 10^4$ cells. The point markers show cell-centres of all the cells in the mesh.

Table 3 Comparison between the DAFoam adjoint derivatives $\mathrm{d}\mathcal{J}/\mathrm{d}\beta$ ($\times 10^{-5}$) with the reference values computed by forward-mode AD. Spalart–Allmaras turbulence model is used. Cases are run in serial.

Cell	Forward AD	Adjoint	Cell	Forward AD	Adjoint
A	−0.04659769	−0.04659868	F	−65.57843895	−65.57986848
B	−4.54223082	−4.54233907	G	−38.87547050	−38.87622607
C	−0.84757061	−0.84759873	H	−2.43771924	−2.43751746
D	−1.54776607	−1.54779623	I	−0.42978504	−0.42979436
E	−24.18868695	−24.18920827	J	0.04248109	0.04248059

IV. Conclusion

An open-source tool was introduced which allows the application of field inversion to recover mean turbulent flows based on limited data. The field inversion formulation is based on the discrete-adjoint method, and requires intrusive code implementation involving the CFD flow solver, the adjoint solver, and the optimiser. In this work, open-source packages OpenFOAM, DAFoam and pyOptSparse are integrated to achieve this. To enable adjoint-based field inversion, three pre-processing steps are required: 1) modification of the baseline turbulence model, 2) adding the code to calculate a particular objective function, and 3) preparing the high-fidelity data. Steps 1-2 require some rudimentary knowledge

of the C++ programming language. As most of the popular RANS turbulence models have already been implemented in OpenFOAM, only minor changes are required to modify the production term of the transport equation. In terms of the objective function we have already implemented a number of these that can work with integral (e.g. aerodynamic force coefficient), surface (e.g. skin friction, surface pressure), and limited (e.g. velocity profiles) and full-field data. These two steps require one-time implementation, and can be subsequently used/generalised for any flow simulations. The simulation is set-up and performed using a Python script. The script is used to set the flow solver parameters (e.g. boundary conditions, residual tolerance, etc.); the adjoint solver parameters (e.g. values for state normalisation), the objective function and the optimiser parameters (e.g. convergence tolerance, etc.). Finally, promising results have been demonstrated on a range of cases, with two turbulence models.

Acknowledgement

Omid Bidar's work is funded by an Engineering and Physical Sciences Research Council (UK) scholarship.

References

[1] Duraisamy, K., Iaccarino, G., and Xiao, H., "Turbulence Modeling in the Age of Data," *Annual Review of Fluid Mechanics*, Vol. 51, No. 1, 2019, pp. 357–377. https://doi.org/10.1146/annurev-fluid-010518-040547.

[2] Ling, J., Kurzawski, A., and Templeton, J., "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, Vol. 807, 2016, pp. 155–166. https://doi.org/10.1017/jfm.2016.615.

[3] Kaandorp, M. L., and Dwight, R. P., "Data-driven modelling of the Reynolds stress tensor using random forests with invariance," *Computers & Fluids*, Vol. 202, 2020, p. 104497. https://doi.org/10.1016/j.compfluid.2020.104497.

[4] Weatheritt, J., and Sandberg, R., "The development of algebraic stress models using a novel evolutionary algorithm," *International Journal of Heat and Fluid Flow*, Vol. 68, 2017, pp. 298–318. https://doi.org/10.1016/j.ijheatfluidflow.2017.09.017.

[5] Schmelzer, M., Dwight, R. P., and Cinnella, P., "Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression," *Flow, Turbulence and Combustion*, Vol. 104, No. 2-3, 2019, pp. 579–603. https://doi.org/10.1007/s10494-019-00089-x.

[6] Beetham, S., and Capecelatro, J., "Formulating turbulence closures using sparse regression with embedded form invariance," *Physical Review Fluids*, Vol. 5, No. 8, 2020. https://doi.org/10.1103/physrevfluids.5.084611.

[7] Xiao, H., and Cinnella, P., "Quantification of model uncertainty in RANS simulations: A review," *Progress in Aerospace Sciences*, Vol. 108, 2019, pp. 1–31. https://doi.org/10.1016/j.paerosci.2018.10.001.

[8] Duraisamy, K., "Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence," *Physical Review Fluids*, Vol. 6, No. 5, 2021. https://doi.org/10.1103/physrevfluids.6.050504.

[9] Xiao, H., Wu, J.-L., Wang, J.-X., Sun, R., and Roy, C., "Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach," *Journal of Computational Physics*, Vol. 324, 2016, pp. 115–136. https://doi.org/10.1016/j.jcp.2016.07.038.

[10] Duraisamy, K., Singh, A.-P., and Pan, S., "Augmentation of Turbulence Models Using Field Inversion and Machine Learning," *55th AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, 2017. https://doi.org/10.2514/6.2017-0993.

[11] Belligoli, Z., Dwight, R. P., and Eitelberg, G., "Reconstruction of Turbulent Flows at High Reynolds Numbers Using Data Assimilation Techniques," *AIAA Journal*, Vol. 59, No. 3, 2021, pp. 855–867. https://doi.org/10.2514/1.j059474.

[12] 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. https://doi.org/10.1016/j.jcp.2015.11.012.

[13] 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. https://doi.org/10.2514/1.j055595.

[14] Ströfer, C. A. M., "DAFI: An Open-Source Framework for Ensemble-Based Data Assimilation and Field Inversion," *Communications in Computational Physics*, Vol. 29, No. 5, 2021, pp. 1583–1622.

[15] He, C., Liu, Y., and Gan, L., "A data assimilation model for turbulent flows using continuous adjoint formulation," *Physics of Fluids*, Vol. 30, No. 10, 2018, p. 105108. https://doi.org/10.1063/1.5048727.

[16] 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.

[17] Spalart, P., and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, 1992. https://doi.org/10.2514/6.1992-439.

[18] Wilcox, D. C., et al., *Turbulence modeling for CFD*, Vol. 2, DCW industries La Canada, CA, 1998.

[19] Jasak, H., Jemcov, A., Tukovic, Z., et al., "OpenFOAM: A C++ library for complex physics simulations," *International workshop on coupled methods in numerical dynamics*, Vol. 1000, IUC Dubrovnik Croatia, 2007, pp. 1–20.

[20] 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.

[21] Wu, N., Kenway, G., Mader, C. A., Jasa, J., and Martins, J. R. R. A., "pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems," *Journal of Open Source Software*, Vol. 5, No. 54, 2020, p. 2564. https://doi.org/10.21105/joss.02564.

[22] Somers, D. M., "Design and experimental results for the S809 airfoil," Tech. rep., Jan 1997. https://doi.org/10.2172/437668.

[23] Laval, J.-P., and Marquillie, M., "Direct Numerical Simulations of Converging–Diverging Channel Flow," *ERCOFTAC Series*, Springer Netherlands, 2011, pp. 203–209. https://doi.org/10.1007/978-90-481-9603-6_21.

[24] McConkey, R., Yee, E., and Lien, F.-S., "A curated dataset for data-driven turbulence modelling," *Scientific Data*, Vol. 8, No. 1, 2021. https://doi.org/10.1038/s41597-021-01034-2.

[25] Xiao, H., Wu, J.-L., Laizet, S., and Duan, L., "Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations," *Computers & Fluids*, Vol. 200, 2020, p. 104431. https://doi.org/10.1016/j.compfluid.2020.104431.