Machine Learning and Field Inversion approaches to Data-Driven Turbulence Modeling

Carlos Alejandro Michelén Ströfer

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

Heng Xiao, Chair
K. Todd Lowe
Eric G. Paterson
Christopher J. Roy

March 24, 2021
Blacksburg, Virginia

Keywords: Turbulence modeling, Deep learning, Bayesian inference, Variational methods, Ensemble methods
Copyright 2021, Carlos Alejandro Michelén Ströfer
Machine Learning and Field Inversion approaches to Data-Driven Turbulence Modeling

Carlos Alejandro Michelén Ströfer

(ABSTRACT)

There still is a practical need for improved closure models for the Reynolds-averaged Navier–Stokes (RANS) equations. This dissertation explores two different approaches for using experimental data to provide improved closure for the Reynolds stress tensor field. The first approach uses machine learning to learn a general closure model from data. A novel framework is developed to train deep neural networks using experimental velocity and pressure measurements. The sensitivity of the RANS equations to the Reynolds stress, required for gradient-based training, is obtained by means of both variational and ensemble methods. The second approach is to infer the Reynolds stress field for a flow of interest from limited velocity or pressure measurements of the same flow. Here, this field inversion is done using a Monte Carlo Bayesian procedure and the focus is on improving the inference by enforcing known physical constraints on the inferred Reynolds stress field. To this end, a method for enforcing boundary conditions on the inferred field is presented. The two data-driven approaches explored and improved upon here demonstrate the potential for improved practical RANS predictions.
Machine Learning and Field Inversion approaches to Data-Driven Turbulence Modeling

Carlos Alejandro Michelén Ströfer

(GENERAL AUDIENCE ABSTRACT)

The Reynolds-averaged Navier–Stokes (RANS) equations are widely used to simulate fluid flows in engineering applications despite their known inaccuracy in many flows of practical interest. The uncertainty in the RANS equations is known to stem from the Reynolds stress tensor for which no universally applicable turbulence model exists. The computational cost of more accurate methods for fluid flow simulation, however, means RANS simulations will likely continue to be a major tool in engineering applications and there is still a need for improved RANS turbulence modeling. This dissertation explores two different approaches to use available experimental data to improve RANS predictions by improving the uncertain Reynolds stress tensor field. The first approach is using machine learning to learn a data-driven turbulence model from a set of training data. This model can then be applied to predict new flows in place of traditional turbulence models. To this end, this dissertation presents a novel framework for training deep neural networks using experimental measurements of velocity and pressure. When using velocity and pressure data, gradient-based training of the neural network requires the sensitivity of the RANS equations to the learned Reynolds stress. Two different methods, the continuous adjoint and ensemble approximation, are used to obtain the required sensitivity. The second approach explored in this dissertation is field inversion, whereby available data for a flow of interest is used to infer a Reynolds stress field that leads to improved RANS solutions for that same flow. Here, the field inversion is done via the ensemble Kalman inversion (EKI), a Monte Carlo Bayesian procedure, and the focus is on improving the inference by enforcing known physical constraints on the inferred Reynolds stress field. To this end, a method for enforcing boundary conditions on the inferred field is presented. While further development is needed, the two data-driven approaches explored and improved upon here demonstrate the potential for improved practical RANS predictions.
Dedication

To my sisters, Melina and Nicole
Acknowledgments

In my experience research is most successful when done as a collaborative endeavour and I would like to first thank two fellow graduate students in my group: Dr. Jinlong Wu whom I learned so much from and Dr. Xinlei Zhang who I worked with closely, learning, coding, and writing together. I am of course grateful to my committee for their support and feedback. I can honestly say I learned a great deal from each one of them. I would also like to thank my fellow graduate students who made the graduate school experience a little more bearable. In this regard, I would like to specially acknowledge my friend Mark Parson, whom I shared an office with our first semester and has since gone with me through all the highs and lows of graduate school life. Finally, I would like to thank my family for their unwavering support throughout this journey.

Part of this material is based on research sponsored by the U.S. Air Force under agreement number FA865019-2-2204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Global Science Press has granted permission to include the published version of the paper “DAFI: An Open-Source Framework for Ensemble-Based Data Assimilation and Field Inversion” for which they hold the copyright.

Elsevier has granted permission to include the published version of the paper “Enforcing boundary conditions on physical fields in Bayesian inversion” for which they hold the copyright.
# Contents

1 Introduction .................................................. 1
   1.1 Data-Driven Turbulence Modeling .......................... 3
   1.2 Field Inversion ............................................. 4
   1.3 Structure and Content ...................................... 7
   1.4 Contributions .............................................. 8
   1.5 Publications ............................................... 9
   1.6 Attributions ............................................... 10
Bibliography .................................................... 11

I Machine Learning .............................................. 15

2 End-to-end differentiable learning of turbulence models from indirect observations 16
   2.1 Introduction ............................................... 17
   2.2 Differentiable Framework for Learning Turbulence Models 18
   2.3 Test Cases .................................................. 22
   2.4 Conclusion ................................................. 25
References ...................................................... 26
Supplemental Materials ......................................... 27

3 Ensemble Gradient for Training Data-Driven Turbulence Models 39
   3.1 Introduction ............................................... 40
   3.2 Ensemble Gradient ......................................... 43
   3.3 Results ..................................................... 46
   3.4 Conclusions ............................................... 49
Appendix A. Regularized Projection ............................. 50
## Field Inversion

4 **DAFI: An Open-Source Framework for Ensemble-Based Data Assimilation and Field Inversion**  
4.1 Introduction .......................................................... 57  
4.2 General Problem Formulation  ........................................ 63  
4.3 Ensemble-Based Analysis Methods  .................................. 65  
4.4 Implementation .......................................................... 71  
4.5 Fields  ........................................................................ 73  
4.6 Test Cases  ................................................................. 77  
4.7 Software Installation and Numerical Examples  ..................... 87  
4.8 Concluding remarks ...................................................... 88  
Appendix C. Kalman Gain Matrix for EnKF  ............................. 89  
Appendix D. Iterative EnKF for Field Inversion  ......................... 89  
Appendix E. Discrete KL Modes ........................................... 91  
Appendix F. Code example .................................................. 92  
References ...................................................................... 94  

5 **Enforcing Boundary Conditions on Physical Fields in Bayesian Inversion**  96  
5.1 Introduction ................................................................. 97  
5.2 Methodology ............................................................... 101  
5.3 Results ................................................................. 105  
5.4 Conclusions ................................................................. 113  
Appendix G. Framework for Bayesian field inversion .................... 115  
References ...................................................................... 116
6 Conclusions and Future Work

6.1 Deep learning with indirect observations ........................................... 119
6.2 Approaches to data-driven RANS turbulence modeling .......................... 121
Bibliography .............................................................................................. 122
Chapter 1

Introduction
Chapter 1. Introduction

The Reynolds-averaged Navier–Stokes (RANS) equations describe the mean velocity and pressure fields of fluid flows and are widely used in engineering applications due to their relatively low computational cost compared to direct numerical simulation (DNS) or large eddy simulations (LES). The RANS equations, however, are unclosed and require modeling of the Reynolds stress tensor. There is no known, generally applicable, turbulence model and the computation of Reynolds stress represents the main source of uncertainty in RANS simulations [1]. The focus of this dissertation is the use of experimental data to improve flow predictions from RANS simulations. Two different approaches are investigated. The first is using machine learning to learn closure models from a set of training data which can then be applied to predict new flows. The second approach is field inversion whereby sparse experimental measurements for a given flow are used to improve the RANS predictions for that flow.

The incompressible steady RANS equations for the mean velocity $u$ and pressure $p$ are a set of four partial differential equations (PDE) given as

$$u \cdot \nabla u - \nu \nabla^2 u + \nabla \cdot \tau + \nabla p - f = 0 \quad (1.1a)$$

$$\nabla \cdot u = 0, \quad (1.1b)$$

where $\tau$ is the unclosed Reynolds stress tensor, $\nu$ is the kinematic viscosity of the fluid, and $f$ is the external body force (momentum source) driving the flow. In order to solve the RANS equations the equations must be closed via a turbulence model which maps mean flow fields to Reynolds stress tensor. The eddy viscosity hypothesis presumes the Reynolds stress tensor to be a local function of the velocity gradient. That is, the Reynolds stress is given as $\tau = \tau(\nabla u, s_1, s_2, \ldots)$, where $s_i$ are turbulence scales with at least two required based on dimensional grounds [2]. The simplest and most widely used models are linear eddy viscosity models (LEVM) which further take the Reynolds stress to be proportional to the strain rate tensor $S$, the symmetric component of the velocity gradient. Commonly used two equations LEVM use an additional two PDEs to model the turbulent kinetic energy $k$ and a turbulent dissipation rate $\varepsilon$ and give the Reynolds stress $\tau = \tau(S, k, \varepsilon)$ as

$$\tau = -2\nu_t S \quad (1.2)$$

$$\nu_t = C_\mu \frac{k^2}{\varepsilon}, \quad (1.3)$$

where $\nu_t$ is the eddy viscosity and $C_\mu$ is a model coefficient. Although widely used, LEVM are inaccurate even in simple flows, including the inability to capture the secondary flow in non-circular pipes [3]. Higher fidelity non-linear eddy viscosity models (NLEVM), such as those obtained from algebraic stress models [2, 4], are not widely used due to inconsistent improvement over LEVM and poor convergence properties [5]. Similarly, models based on the Reynolds stress transport equations model non-local dynamics not captured by NLEVM but are not widely used because of the added complexity and poor numerical stability [6].

Recently, there has been a renewed interest in using data to improve RANS predictions [7]. The first approach explored here is using machine learning to learn a NLEVM based on
data from different flows. Such a model maps from mean velocity gradient and modeled turbulence scales to Reynolds stress and can be used to predict new flows. The second approach explored here uses sparse velocity or pressure data for a given flow to infer the Reynolds stress field using Bayesian inference. The inferred Reynolds stress can then be plugged into the RANS equations without a need for turbulence modeling. In this approach the Reynolds stress field predicted by a traditional turbulence model, such as LEVM, is used as the prior mean in the Bayesian inference. These two approaches are introduced in more detail in the next sections.

At their core, both approaches are optimization problems and there is a choice as to the nature of the optimization solution. In this dissertation both variational and ensemble approaches are used. For the field inversion problem, an iterative Bayesian approach is used with an ensemble of states used to approximate the distributions. Each state evolves through the Bayesian update in a gradient-free manner, and the converged ensemble mean becomes the inferred field. This approach is in contrast to variational methods where an explicit cost function is minimized, e.g. the maximum a posteriori estimate (MAP) of the Bayesian problem, and the problem is solved through gradient-based optimization using the model adjoint for the required sensitivity. For the machine learning approach, the network is trained with gradient-based optimization, and both variational and ensemble methods are used to obtain the sensitivity of the RANS equations. The variational method uses the continuous adjoint, which gives an explicit formulation for the sensitivities of the RANS equations. This requires deriving and implementing the adjoint equations but only requires one additional PDE solution to obtain the sensitivity. Alternatively, an ensemble of states can be used to estimate the sensitivity. This has the advantage of not being intrusive, treating the RANS equations as a black box, but requires multiple RANS solutions.

## 1.1 Data-Driven Turbulence Modeling

The goal of data-driven turbulence modeling is to learn a single-point mapping from modeled mean field quantities to Reynolds stress. These models have been developed and evolving over the last decade with some clear trends towards physical consistency. The necessity for coordinate invariant inputs and outputs was recognized early on. Most early efforts trained disparate functions from a set of ad-hoc invariant inputs to the components of the eigendecomposition of the Reynolds stress [8, 9, 10, 11]. For instance, Wang et al. [10] trained a random forest to map from 10 invariant inputs (e.g. Q-criterion) to the turbulence kinetic energy (magnitude), eigenvalues (shape), and eigenvectors (orientation) of the anisotropic Reynolds stress. The ad-hoc choice of invariant inputs was replaced in several works by systematically creating a basis of invariant inputs, e.g. Ling et al. [9] and Wu et al. [12]. Still, most efforts were based on learning independent scalar functions from invariant inputs to the invariant components (e.g. eigenvalues) of the Reynolds stress. A major shift towards physical compatibility happened when Ling et al. [13] replaced these disparate scalar func-
Chapter 1. Introduction

...ions with a single invariant tensor function using an integrity basis [2]. This limited the scope of data-driven models to learning NLEVM but gave the complete general formulation for learning such models. This formulation has since been used by several works, including efforts using neural networks ([14]), random forests ([15], gene expression programming ([16, 17]), and sparse symbolic regression ([18, 19]).

Another major shift was learning models that could be integrated into an iterative RANS solver. Earlier efforts, including Ling et al. [13] and Wu et al. [20], trained with RANS inputs and DNS outputs which limited the learned models to be used as an intermediate step to map baseline RANS results to corrected Reynolds stress. One approach to address this has been obtaining, from the high fidelity data, inputs that are consistent with RANS. The input to the NLEVM is the velocity gradient non-dimensionalized by a modeled turbulence timescale and the RANS modeled timescales are highly empirical. Consistent high fidelity turbulent timescales have been obtained by passively solving the RANS turbulence transport equations using the high fidelity mean velocity [16, 18, 21]. An alternative approach is to iteratively solve the RANS equations to re-generate the inputs and retrain the model [14].

Currently, there is a shift away from high fidelity Reynolds stress data to velocity and pressure data that can be obtained experimentally [17, 22]. Such data includes full field data (e.g. particle image velocimetry), sparse point measurements (e.g. pressure probes), and integral quantities like lift and drag. The efforts in this dissertation [23, 24] are a part of this shift to using derived measurable quantities. Specifically this dissertation presents a framework for training deep neural networks using gradient-based optimization and velocity and pressure data. The use of velocity and pressure data requires the solution of the RANS equations at every step as well as the sensitivity of the RANS equations. For the sensitivity two different approaches are explored, the continuous adjoint method and ensemble methods. Since the RANS equations must be solved at every step, the neural network inputs can be updated iteratively and converge together with the neural network’s parameters. The other two efforts in using measurable quantities to learn turbulence models differ from the approach here and can be considered parallel efforts. Zhao et al. [17] use gradient-free optimization and genetic algorithms to learn symbolic expressions for the turbulence closure. Holland et al. [22] keep a traditional LEVM and learn a scalar corrective field for the production of turbulence kinetic energy using neural networks, rather than learning the closure itself as done here and by Zhao et al. [17].

1.2 Field Inversion

The problem at hand is how to use sparse experimental data for a flow of interest to improve the RANS predictions for the full flow field. For this we use Bayesian inference where different sources of information and their uncertainties are combined to obtain a more accurate prediction. For a state $x$ we which to infer and a set of data $d$, Bayes’ theorem can be
expressed as

\[ P(x|d) \propto P(x)P(d|x), \]  \hspace{1cm} (1.4)

where \( P(x|d) \) is the posterior distribution of the state \( x \) after accounting for the observed data, \( P(x) \) is the prior distribution or belief prior to seeing the data, and \( P(d|x) \) is the likelihood that the data came from the state \( x \). The state to be inferred could be any field in the RANS equations and the prior distribution can be taken as a Gaussian process with mean taken as the RANS predicted field and some assumed covariance structure. The likelihood of the data can similarly be expressed using a normal distribution with mean given by the measured values and independent standard deviations given by the measurement’s uncertainties.

The fields of interest are the velocity and pressure and the data consists of measurements of velocity and pressure. Bayesian inference could be used directly to infer these fields [25], i.e. with the state \( x \) in Equation 1.4 consisting of velocity and pressure. This black box approach however treats the entire RANS equations as uncertain. It is known however that the uncertainty in the RANS predictions come from the the Reynolds stress tensor and one could treat the Reynolds stress as the uncertain field to be inferred. This embedded discrepancy [26] opens the model and corrects only the term that is known to be uncertain. This has the advantage of enforcing the known physics, i.e. the predicted velocity and pressure fields are solutions to the RANS equations which is not the case if one infers them directly. In this approach the RANS equations are seen as a mapping from Reynolds stress to velocity and pressure

\[ \mathbf{u}, p = \mathcal{F}(\tau), \]  \hspace{1cm} (1.5)

and the goal is to infer the input Reynolds stress from observations of the output velocity and pressure. For sparse observations however, this inverse problem is ill-posed since different input fields can result in the same observations. For this reason, the Bayesian formulation is used to regularize the inverse problem based on the prior solution.

The observation operator is the mapping from state space to observation space. With the Reynolds stress as the state, the observation operator includes solving the RANS equations and is therefore not linear. However, as will be shown later, the techniques used for the case of a linear observation operator can be adapted for non-linear operators and the linear case is considered first. For a linear observation operator, direct application of Bayes’s theorem results in the Kalman filter, a common technique in data assimilation. In this case the Kalman filter would be used once resulting in the posterior distribution. For problems where the state consists of discretized fields the state vector can be very large and direct application of the Kalman filter becomes unmanageable. Instead, a Monte Carlo approach can be used to represent the probability distributions by a finite set of samples. Each prior sample is updated to a posterior sample and the posterior distribution estimated from the posterior samples statistics. Direct application of the Kalman filter using an ensemble results in the ensemble Kalman filter (EnKF) [27], a common ensemble method. Problems with a nonlinear observation operator can be recast as an artificial dynamics problem with a linear
observation operator \cite{28} for an augmented state that includes the field in both the state and observation space. This method, referred to as the ensemble Kalman inversion (EKI), iteratively applies the ensemble Kalman filter using the same observation data but updating the state based on the nonlinear model, e.g. the RANS equations.

Xiao et al. \cite{29} used the EKI approach to infer the Reynolds stress field from sparse velocity measurements. For this they represented the Reynolds stress field based on its eigendecomposition and inferred scalar fields corresponding to the magnitude (turbulent kinetic energy), and eigenvalues and eigenvectors of the anisotropic component of Reynolds stress. Xiao et al. and similar works use DNS data to create synthetic sparse observations which allows comparison of the inferred Reynolds stress field and resulting velocity field with the full field truth. The results show significant improvement in velocity prediction over the baseline RANS, but the inferred Reynolds stress was worst than the original RANS predictions. This is possible if the flow velocity is not very sensitive to the Reynolds stress field in certain regions of the flow, e.g. regions with small velocity gradient.

The goal of the field inversion work presented in this dissertation is to get better inference of the Reynolds stress field not just better predictions of the velocity field. A simpler inversion problem, using a synthetic LEVM for the truth, is considered where only the scalar eddy viscosity field is inferred. In previous works the velocity and pressure fields satisfy the physics in the RANS equations but only a few physical constraints are enforced on the inferred Reynolds stress fields obtained through Bayesian inference. In Xiao et al. \cite{29} the turbulent kinetic energy is constrained to be positive by using a lognormal process rather than a Gaussian process to represent it and the same strategy has been used in this dissertation for the eddy viscosity field. The main improvement presented in this dissertation is enforcing the known boundary conditions on the inferred eddy viscosity fields. This is done by modifying the covariance structure such that any realization of the random field satisfies the boundary conditions. This was used to enforce the periodic, symmetry, and wall conditions for the same cases as in Xaio et al. It is also shown that the reason for the inferred field becoming orders of magnitude larger than the prior and truth was due to an instability caused by using observations for a lognormal process close to a boundary with zero-value Dirichlet condition, e.g. walls. Avoiding such measurements and enforcing boundary conditions resulted in improved posterior fields over the prior, as expected of Bayesian inference, and required much fewer measurements to obtain the desired improvements to the velocity field.

This research resulted in two additional efforts to improve the inferred field through enforcing known constraints that are not part of this dissertation but are now summarized briefly. The Reynolds stress transport equations can be considered an additional source of information in the form of physical constraint on the inferred fields. A method was developed to enforce such transport equations by modifying the covariance structure \cite{20}. This results in a covariance structure that clearly reflects the convection and diffusion in the transport equations. Another effort was the development of a regularized iterative ensemble method \cite{30} that can enforce arbitrary constraints during the ensemble Kalman inversion (EKI). Unlike the variational approach, when solving the inverse problem through EKI there is no explicit cost
function to which to add additional regularization. Instead the EKI optimizes an implicit cost function through the Kalman update of the samples. Starting from a regularized cost function, a modified update scheme was derived for the iterative EKI.

The ensemble inversion approach used here is in contrast to variational approach which is summarized here for completeness. In the variational approach a cost function is explicitly minimized through gradient-based optimization. The cost function can for instance be the maximum a posteriori estimate of the Bayesian problem, i.e.

\[ J(\tau) = \|H(\tau) - d\|^2_{C_d^{-1}} + \|\tau - \bar{\tau}\|^2_{C_{\tau}^{-1}} \]  

(1.6)

where \( H \) is the observation operator, \( \bar{\tau} \) is the prior mean, e.g. obtained from a baseline RANS solution, and \( C_d \) and \( C_{\tau} \) are the covariance matrix associated with the data and the prior solution. The gradient \( \nabla J \) requires the sensitivity of the nonlinear observation operator which includes the RANS equations. This is obtained via the adjoint method.

The focus of this dissertation is on the EKI approach for inferring Reynolds stress from sparse measurements as used by Xiao et al. [29], but there have been other efforts that use different approaches to incorporate case-specific data to improve RANS predictions using the embedded discrepancy formulation. Several works, including those by Edeling et al. [31, 32] and Ray et al. [33, 34, 35], infer model coefficients in existing turbulence models using Markov Chain Monte Carlo methods. Other works have used the variational approach to infer embedded fields. Dow and Wang [36] inferred the eddy viscosity from full field velocity observation, and Singh and Duraisamy [37] used limited data to infer an embedded multiplicative discrepancy in the production term of the Spalart–Allmaras model. A different approach taken by Meldi et al. [38, 39] embeds reduced order Kalman filtering techniques into CFD solvers to do the data assimilation simultaneously with the CFD solution. Direct application of Kalman filtering to RANS simulations results in a large, intractable problem. In the approach by Meldi et al. the Kalman filtering technique is not approximated using an ensemble as in the EKI but simplified by making other assumptions such as a diagonal covariance matrix for the prior distribution. Yet another approach, used by Edeling et al. [40] propose new transport equations for the eddy viscosity perturbations and use Bayesian inference to calibrate the case-specific model constants based on data. All these efforts use data to correct a source of uncertainty in the model rather than to correct the model predictions directly.

1.3 Structure and Content

This dissertation follows the *Manuscript* format. The dissertation is divided into two parts. Part I contains Chapters 2 and 3 and consists of efforts related to learning turbulence closures using machine learning. Part II contains Chapters 4 and 5 and consists of efforts related to inferring the Reynolds stress field through field inversion based on sparse measurements. The content of each chapter is as follows:
Chapter 1: Introduces the problem of using sparse data to improve RANS predictions, presents an overview and literature review of the two approaches explored in this dissertation, and summarizes the outline, content, and contributions of the dissertation.

Part I: Machine Learning

Chapter 2: Consists of a paper, entitled “End-to-end differentiable learning of turbulence models from indirect observations” and in preparation for submission to the Journal of Fluid Mechanics Rapids, or similar journal. This paper presents the framework whereby a neural network turbulence model is trained using velocity or pressure measurements. The continuous adjoint equations are derived and implemented to obtain the required sensitivity of the RANS equations. The trained neural network is incorporated into an iterative RANS solver as a custom turbulence model in OpenFOAM.

Chapter 3: Consists of a paper, entitled “Ensemble gradient for training data-driven turbulence models” and in preparation for submission to a peer reviewed journal. This paper presents ensemble methods as an alternative to the adjoint equations for obtaining the sensitivity of the RANS equations.

Part II: Field Inversion

Chapter 4: Consists of a peer reviewed paper, entitled “DAFI: An open-source framework for ensemble-based data assimilation and field inversion” and published in Communications in Computational Physics (reproduced here with permission from Global Science Press). This paper presents an overview of the theory of ensemble-based field inversion and presents the details of the software developed for the purpose of testing such methods.

Chapter 5: Consists of a peer reviewed paper, entitled “Enforcing boundary conditions on physical fields in Bayesian inversion” and published in Computer Methods in Applied Mechanics and Engineering (reproduced here with permission from Elsevier). This paper presents a methodology for enforcing boundary conditions on the inferred field during field inversion. The paper also identifies the source of the exploding magnitude of the inferred field seen in previous works as an instability associated with measurements near zero-valued boundaries.

1.4 Contributions

The main contributions of this dissertation towards the goal of learning a predictive turbulence closure from data are the following:
1. Development of a framework for training deep neural networks representations of the turbulence closure by using velocity and pressure measurements and implementation of the trained network into an iterative RANS turbulence model.

2. Derivation and use of both continuous adjoint and ensemble methods for obtaining the RANS sensitivity required during training.

The main contributions of this dissertation towards the goal of inferring the Reynolds stress field from sparse observations are the following:

1. The enforcement of boundary conditions on the inferred field during field inversion problems in a systematic way. Boundary conditions had been often ignored by previous related works or enforced only in simple cases in an ad-hoc manner. A code was developed and released to develop and test new field inversion methods including the method for enforcing boundary conditions presented in this dissertation as well as methods for enforcing prior physical or empirical knowledge presented in related collaborative publications.

2. The identification of a singularity—when using a lognormal process to represent the inferred field and making observations near a boundary with zero-value Dirichlet condition—as the cause of the exploding magnitude of the inferred field seen in previous related works. This issue was previously wrongly attributed to the ill-posedness of the problem.

1.5 Publications

Articles in preparation related to the machine learning effort:

- **Carlos A. Michelén Ströfer** and Heng Xiao. *End-to-end differentiable learning of turbulence models from indirect observations.* (In preparation).


Peer-reviewed journal publications related to the field inversion effort:


Other peer-reviewed journal publications published during my Ph.D. studies:


1.6 Attributions

The nature of technical research is collaborative and this is reflected in the number of co-authored papers produced during my Ph.D. studies. The full list of articles is presented in Section 1.5, and the present dissertation includes only articles for which I was the major contributor and writer of the manuscript. The individual contributions of the co-authors to this research are as follows:

• Dr. Heng Xiao is my Ph.D. advisor. He participated in conceptualization, planning, and discussions, provided advice and guidance, and reviewed and edited my writing.

• Dr. Xinlei Zhang was a fellow doctoral student. He worked closely with me deriving new methods, implementing code, and running simulations. Our collaboration produced several works, some of which he lead and some of which I lead and are presented here.

• Dr. Olivier Coutier-Delgosha was Xinlei’s Ph.D. advisor. He provided Xinlei with support and guidance.
Bibliography


Part I

Machine Learning
Chapter 2

End-to-end differentiable learning of turbulence models from indirect observations
End-to-end differentiable learning of turbulence models from indirect observations

Carlos A. Michelén Ströfer\textsuperscript{1} and Heng Xiao\textsuperscript{1}\textsuperscript{‡}

\textsuperscript{1}Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060, USA

(Received xx; revised xx; accepted xx)

The emerging push of the differentiable programming paradigm in scientific computing is conducive to training deep learning turbulence models using indirect observations. This paper demonstrates the viability of this approach and presents a framework for training deep neural networks to learn eddy viscosity models from observations derived from the velocity and pressure fields. For the sensitivities of the measured quantities to the Reynolds stress field we derive the continuous adjoint equations for the Reynolds-averaged Navier–Stokes (RANS) equations. We demonstrate the ability of this approach to learn the true underlying turbulence closure when one exists by training models using synthetic velocity data from linear and nonlinear closures. We also train a linear eddy viscosity model using synthetic velocity measurements from direct numerical simulations of the Navier–Stokes equations (DNS) for which no true underlying linear closure exists. The trained deep neural network turbulence model is then used to predict similar flows.

Key words: turbulence modelling, turbulence theory, variational methods

1. Introduction

There still is a practical need for improved closure models for the Reynolds-averaged Navier–Stokes (RANS) equations. Currently, the most widely used turbulence models are linear eddy viscosity models (LEVM), which presume the Reynolds stress is proportional to the mean strain rate. Although widely used, LEVM do not provide accurate predictions in many flows of practical interest, including the inability to predict secondary flows in non-circular ducts (Speziale 1982). Alternatively, non-linear eddy viscosity models (NLEVM) capture nonlinear relations from both strain and rotation rate tensors. NLEVM, however, do not result in consistent improvement over LEVM and can suffer from poor convergence (Gatski & Speziale 1993). Data-driven turbulence models are an emerging alternative to traditional single-point closures. Data-driven NLEVM use the integrity basis representation (Pope 1975; Ling \textit{et al.} 2016) to learn the mapping from the velocity gradient field to the normalized Reynolds stress anisotropy field, and retain the transport equations for turbulence quantities from a traditional model.

It has been natural to train such models using Reynolds stress data (e.g. Ling \textit{et al.} 2016; Weatheritt \& Sandberg 2017). However, Reynolds stress data from high-fidelity simulations, i.e. DNS or LES, is mostly limited to simple geometries and low Reynolds number. It is therefore desirable to train with \textit{indirect observations}, such as quantities

\textsuperscript{†} Email address for correspondence: cmich@vt.edu
\textsuperscript{‡} Email address for correspondence: hengxiao@vt.edu
based on the velocity or pressure fields, for which experimental data is available for a much wider range of complex flows. Such measurements include full field data, e.g. from particle image velocimetry, sparse point measurements such as from pressure probes, and integral quantities such as lift and drag. Training with indirect observations has the additional advantage of circumventing the need to extract turbulence scales that are consistent with the RANS modeled scales from the high fidelity data (e.g. Weatheritt & Sandberg 2017).

Recently, Zhao et al. (2020) learned a zonal turbulence model for the wake-mixing regions in turbomachines in symbolic form (e.g., polynomials and logarithms) from indirect observation data by using genetic algorithms. However, while symbolic models are easier to interpret, they may have limited expressive power as compared to, e.g., deep neural networks (Raghu et al. 2017), which are successive composition of nonlinear functions. Symbolic models may therefore not be generalizable and be limited to zonal approaches. More importantly, gradient-free optimization method such as genetic programming may not be as efficient as gradient-descent methods, and the latter should be used whenever available (Audet & Kokkolaras 2016). In particular, deep learning methods (LeCun et al. 2015) have achieved remarkable success in many fields and represent a promising approach for data-driven NLEVM (e.g. Ling et al. 2016).

A major obstacle for gradient-based learning from indirect observations is that a RANS solver must be involved in the training and its gradient is required to learn the model. While, such gradients can be obtained by using adjoint equations, which have long been used in aerodynamic shape optimization (Jameson 1988), these are not generally rapidly available or straightforward to implement. The emerging interest in differentiable programming is resulting in efficient methods to develop adjoint accompanying physical models, including modern programming languages that come with built-in automatic differentiation (Bezanson et al. 2019), or neural-network-based solutions of partial differential equations (Raissi et al. 2019). Recently, Holland et al. (2019) used the discrete adjoint to learn a corrective scalar multiplicative field in the production term of the Spalart–Allmaras transport model. This is based on an alternative approach to data-driven turbulence modeling (Parish & Duraisamy 2016) in which empirical correction terms for the turbulence transport equations are learned while retaining a traditional linear closure (LEVM).

In this work we demonstrate the viability of training deep neural networks to learn general eddy viscosity closures (NLEVM) using indirect observations. We embed a neural-network-represented turbulence model into a RANS solver using the integrity basis representation, and as a proof of concept use the continuous adjoint equations for the required RANS sensitivities. This leads to an end-to-end differentiable framework that provides the gradient information needed to learn turbulence models from indirect observations.

### 2. Differentiable framework for learning turbulence models

In this proposed framework a neural network is trained by optimizing an objective function that depends on quantities derived from the network’s outputs rather than on those outputs directly. The training framework is illustrated schematically in figure 1 and consists of two components: the turbulence model and the objective function. Each of these two components has a forward model that propagates inputs to outputs and a backwards model that provides the sensitivities of the outputs with respect to the inputs or parameters. The gradient of the objective function $J$ with respect to the network’s trainable parameters $\mathbf{w}$ is obtained by combining the sensitivities from the
Differentiable learning of turbulence models from indirect observations

\[ \mathcal{T}(k, t_\tau) = 0 \]

\[ \tau \]

\[ \theta \]

\[ g \]

\[ w \]

\[ \nabla \cdot \mathbf{u} = 0 \]

\[ \nabla \cdot \mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla \cdot \tau + \nabla p - s = 0 \]

\[ J = J(\mathbf{u}, p) \]

Figure 1. Schematic of the end-to-end differentiable training framework. The framework consists of two main components, the turbulence model and the observation operator, each of which has a forward and backwards (adjoint) model. For any value of the network’s parameters, \( w \), the gradient of the objective function, \( J \), can be obtained by solving these four problems. The turbulence model consists of a neural network representing the closure function using integrity basis representation, \( \theta \mapsto g \), and transport equations \( \mathcal{T}(k, t_\tau) = 0 \) for the turbulence quantities. The observation operator consists of solving the RANS equations with the proposed turbulence model and extracting the quantities of interest that are compared to the observations in the cost function \( J \).

Two components through the chain rule as

\[ \frac{\partial J}{\partial w} = \frac{\partial J}{\partial \tau} \frac{\partial \tau}{\partial w}, \]

where \( \tau \) is the Reynolds stress anisotropy (deviatoric component).

2.1. Forward model

For given values of the trainable parameters \( w \), the forward model evaluates the cost function \( J \) that is the discrepancy between predicted and observed quantities. The forward evaluation consists of two main components: (i) evaluating the neural network turbulence model and (ii) mapping the network’s outputs to observation space by first solving the RANS equations. The two components are detailed below.

The turbulence model, shown on the left box in figure 1, maps the velocity gradient field to the Reynolds stress field. The integrity basis representation for a general eddy viscosity model (Pope 1975) is given as

\[ \tau = 2k \sum_{i=1}^{10} g^{(i)} T^{(i)}, \quad g^{(i)} = g^{(i)}(\theta_1, \ldots, \theta_5), \]

where \( T \) and \( \theta \) are the basis tensor functions and scalar invariants of the input tensors, and \( g \) are the scalar coefficient functions to be learned. The input tensors are the symmetric and antisymmetric components of the normalized velocity gradient: \( S = \tfrac{1}{2} t_\tau (\nabla \mathbf{u} + \nabla \mathbf{u}^\top) \) and \( R = \tfrac{1}{2} t_\tau (\nabla \mathbf{u} - \nabla \mathbf{u}^\top) \), where \( t_\tau \) is the turbulence time-scale and \( \mathbf{u} \) is the mean velocity. The linear and quadratic terms in the integrity basis representation
are given as
\[ T(1) = S, \quad T(2) = SR - RS, \quad T(3) = S^2 - \frac{1}{3} \{ S^2 \} I, \quad T(4) = R^2 - \frac{1}{3} \{ R^2 \} I \]

where \( I \) is the second rank identity tensor, and curly braces \( \{ \cdot \} \) indicate the tensor trace.

Different eddy viscosity models differ in the form of the scalar coefficient functions, \( \theta \mapsto g \), and in the models for the two turbulence scale quantities \( k \) and \( \tau_r \). We represent the scalar coefficient functions using a deep neural network with 10 hidden layers with 10 neurons each and a rectified linear unit (ReLU) activation function. The turbulence scaling parameters are modelled using traditional transport equations, \( T(k, \tau) = 0 \), with the TKE production term \( P \) modified to account for the expanded formulation of Reynolds stress: \( P = \tau : \nabla u \), where : denotes double contraction of tensors.

The RANS solver along with its post-processing serves as an observation operator that maps the neural network’s outputs (Reynolds stress anisotropy \( \tau \)) to the observation quantities (e.g., sparse velocity measurement, drag). This is shown in the right box in figure 1. The first step in this operation is to solve the RANS equations with the given Reynolds stress closure to obtain velocity and pressure fields. This is followed by post-processing to obtain the observation quantities (e.g., sampling velocities at certain locations or integrating surface pressure to obtain drag). When solving the RANS equations, explicit treatment of the divergence of Reynolds stress can make the RANS equations ill-conditioned (Wu et al. 2019; Brener et al. 2021). We treat part of the linear term implicitly by use of an effective viscosity, \( \nu_{\text{eff}} \), which is easily obtained since with the integrity basis representation the linear term is learned independently. The incompressible RANS equations are then given as

\[
\begin{align*}
\mathbf{u} \cdot \nabla \mathbf{u} - \nabla \nu_{\text{eff}} \nabla \mathbf{u} - \nabla \mathbf{u} \cdot \nabla \nu_{\text{eff}} + \nabla \cdot \tau_{\text{NL}} + \nabla p &= s, \\
\nabla \cdot \mathbf{u} &= 0, \\
\nu_{\text{eff}} &= \nu - g^{(1)} k \tau_r, \\
\tau_{\text{NL}} &= 2k \sum_{i=2}^{N_T} g^{(i)} T^{(i)}
\end{align*}
\]

where the term \( \nabla \nu_{\text{eff}} \nabla \mathbf{u} \) is treated implicitly. The coefficients \( g^{(i)} \) are outputs of the neural network-based turbulence model that have the fields \( \theta \) as input.

2.2. Adjoint model

For a proposed value of the trainable parameters \( \mathbf{w} \) the backwards model (represented by left-pointing arrows in figure 1) provides the gradient \( \partial J / \partial \mathbf{w} \) of the cost function with respect to the trainable parameters. This is done by separately obtaining the two terms on the right hand side of equation (2.1): (i) the gradient of the Reynolds stress \( \partial \tau / \partial \mathbf{w} \) from the neural network turbulence model and (ii) the sensitivity \( \partial J / \partial \tau \) of the cost function to the Reynolds stress, using their respective adjoint models. Combining the two adjoint models results in an end-to-end differentiable framework, whereby the gradient of observation quantities (e.g. sparse velocity) with respect to the neural network’s parameters can be obtained. The two components of this adjoint model are detailed below.

The gradient of the Reynolds stress anisotropy with respect to the neural network’s parameters \( \mathbf{w} \) is obtained in two parts using the chain rule. The gradient of the neural network outputs with respect to its parameters, \( \partial g / \partial \mathbf{w} \), is efficiently obtained by backpropagation, which is a reverse accumulation automatic differentiation algorithm for deep neural networks that applies the chain rule on a per-layer basis. The entries
of the sensitivities of the Reynolds stress with respect to the coefficient functions are obtained as $\partial \tau / \partial g^{(i)} = 2k \hat{T}^{(i)}$ from differentiation of equation 2.2, which is a linear tensor equation.

For the sensitivity of the objective function with respect to the Reynolds stress we derived the appropriate continuous adjoint equations. Since the Reynolds stress must satisfy the RANS equations, this is a constrained optimization problem. The problem is reformulated as the optimization of an unconstrained Lagrangian function with the Lagrange multipliers described by the adjoint equations. The resulting adjoint equations are

\[
\begin{align*}
    u \cdot \nabla \hat{u} + \nabla \hat{u} \cdot u + \nu \nabla^2 \hat{u} - \nabla \hat{p} &= \frac{\partial J_{\Omega}}{\partial u}, \\
    \nabla \cdot \hat{u} &= -\frac{\partial J_{\Omega}}{\partial p},
\end{align*}
\]

(2.5)

where $\hat{u}$ and $\hat{p}$ are the Lagrange multipliers, referred to as adjoint velocity and adjoint pressure, respectively, and $J_{\Omega}$ is the scalar field such that the objective function is given as the integral $J = \int J_{\Omega} d\Omega$ over the solution domain $\Omega$. When the cost function is instead given as an integral over the domain boundary (e.g. drag) the cost function affects the boundary conditions of the adjoint equations instead (Othmer 2008). When solving the adjoint equation using a segregated approach such as the SIMPLE algorithm used here, the adjoint transpose convection term $\nabla \hat{u} \cdot u$ is treated explicitly and can result in instabilities (Oriani & Pierrot 2016). For this reason it is common to dampen or eliminate this term (Othmer 2014), and here we eliminate it. After solving the adjoint equations, the sensitivity of the function $J$ to the Reynolds stress is given as the gradient of the adjoint velocity

\[
\frac{\partial J}{\partial \tau} = -\nabla \hat{u}.
\]

(2.6)

The details of the derivation and validations of the continuous adjoint equations are shown in (Michelén Ströfer 2021) for interested readers.

### 2.3. Gradient descent procedure

The training is done using the Adam algorithm, a gradient descent algorithm with momentum and adaptive gradients commonly used in training deep neural networks. The default values for the Adam algorithm are used, including a learning rate of 0.001. The training requires solving the RANS equations at each training step. In a given training step the inputs $\theta_i$ are updated based on the previous RANS solution and scaled to the range 0-1, and the RANS equations are then solved with fixed values for the coefficient functions $g$. The inputs $\theta_i$ and their scaling parameters are fixed at a given training step and converge alongside the main optimization of the trainable parameters $w$.

Initialization of the neural network’s parameters requires special consideration. The usual practice of random initialization of the weights is not suitable in this case since it leads to divergence of the RANS solution. We use existing closures (e.g. a laminar model with $g^{(i)} = 0$ or a linear model with $g^{(1)} = -0.09$) to generate data for pre-training the neural network and thus provide a suitable initialization. This has the additional benefit of embedding existing insight into the training by choosing an informed initial point in the parameter space. When pre-training to constant values (e.g. $g^{(1)} = -0.09$) we add noise to the pre-training data, since starting from very accurate constant values can make the network difficult to train.
3. Test cases

The viability of the proposed framework is demonstrated by testing on three test cases. The first two cases use synthetic velocity data obtained from a linear and a non-linear closure, respectively, to train the neural network. The use of synthetic data allows us to evaluate the ability of the training framework to learn the true underlying turbulence closure when one exists. In the final test case realistic velocity measurements, obtained from a DNS solution and for which no known true underlying closure exists, are used to learn a linear eddy viscosity model. The trained LEVM is then used to predict similar flows and the predictions are compared to those from a traditional LEVM.

3.1. Learning a synthetic LEVM from channel flow

As a first test case we use a synthetic velocity measurement at a single point from the turbulent channel flow to learn the underlying linear model. The flow has a Reynolds number of 10,000 based on bulk velocity and half channel height. The turbulent equations used are the $k$–$\omega$ model of Wilcox (1998), and the synthetic model corresponds to a constant $g^{(1)} = 0.09$. For the channel flow there is only one independent scalar invariant and $T^{(1)}$ is the only linear tensor function in the basis. We therefore use a neural network with one input and one output which maps $\theta_1 \mapsto g^{(1)}$. The network has 1021 trainable parameters and is pre-trained to the laminar model $g^{(1)} = 0$. The sensitivity of the predicted point velocity to the Reynolds stress is obtained by solving the adjoint equations with $J_\Omega$ equal to the velocity field times a radial basis function. Figure 2 shows the results of the training. The trained model not only results in the correct velocity field, but the correct underlying model is learned.

3.2. Learning a synthetic NLEVM from flow through a square duct

As a second test case we use a synthetic full field velocity measurement from flow in a square duct to learn the underlying nonlinear model. The flow has a Reynolds number of 3,500 based on bulk velocity and half duct side length. This flow contains a secondary in-plane flow that is not captured by LEVM (Speziale 1982). For the objective function, $J_\Omega$ is the difference between the measured and predicted fields, with the discrepancy of the in-plane velocity scaled by a factor of 1,000 as to have a similar weight to the axial velocity discrepancy. The NLEVM is the Shih quadratic model (Shih et al. 1993) which

---

**Figure 2.** Results of learning a LEVM from a single velocity measurement in a turbulent channel flow: (a) the learned coefficient function, (b) the Reynolds stress field, and (c) the velocity. The final (trained) results overlap with the truth and the two are visually indistinguishable.
Differentiable learning of turbulence models from indirect observations

Figure 3. Results of learning a NLEVM, the Shih quadratic model, from full field velocity measurements in flow through a square duct. The results shown are the two combinations of coefficient functions that have an effect on velocity plotted against the scalar invariant $\theta_1 \approx -\theta_2$.

\[
g_1(\theta_1, \theta_2) = \frac{-2/3}{1.25 + \sqrt{2\theta_1} + 0.9\sqrt{-2\theta_2}}, \quad g_2(\theta_1, \theta_2) = \frac{7.5}{1000 + (\sqrt{2\theta_1})^3}, \quad g_3(\theta_1, \theta_2) = \frac{1.5}{1000 + (\sqrt{2\theta_1})^3}, \quad g_4(\theta_1, \theta_2) = \frac{-9.5}{1000 + (\sqrt{2\theta_1})^3}. \tag{3.1}
\]

For the flow in a square duct only four combinations of the Reynolds stress components affect the predicted velocity with RANS equations: $\tau_{xy}$ and $\tau_{xz}$ in the axial equation and $\tau_{yz}$ and $(\tau_{zz} - \tau_{yy})$ in the in-plane equation. In this flow the in-plane velocity gradients are orders of magnitude smaller than the gradients of the axial velocity $u_x$. For these reasons only two combinations of coefficient functions can be learned, $g^{(1)}$ and the combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$, and there is only one independent scalar invariant with $\theta_1 \approx -\theta_2$.

Figure 4. (a): Velocity and Reynolds stress anisotropy results of learning a NLEVM from full field velocity measurements in flow through a square duct. The $u_z$ and $\tau_{xz}$ fields are the reflection of $u_y$ and $\tau_{xy}$ along the diagonal. (b): Schematic of flow through a square duct showing the secondary in-plane velocities ($y$-$z$ plane). The simulation domain (bottom left quadrant) is shown in blue.

The neural network has two inputs and four outputs and was pre-trained to the LEVM with $g^{(1)} = -0.09$. The turbulent equations used are those from the Shih quadratic $k$–$\varepsilon$ model. Figure 3 shows the learned model which shows improved agreement with the truth. The combination $g^2 - 0.5g^3 + 0.5g^4$ shows good agreement only for the higher range of scalar invariant $\theta_1$. This is due to the smaller scalar invariants corresponding to
smaller velocity gradients and smaller magnitudes of the tensors $\mathbf{T}$. The velocity field is therefore expected to be less sensitive to the value of the Reynolds stress in these regions. It was observed that the smaller range of the invariant, where the learned model fails to capture the truth, occurs mostly in the center channel. Figure 4 shows the ability of the learned model to capture the correct velocity, including predicting the in-plane velocities, and the Reynolds stress anisotropy. The trained model fails to predict the correct $\tau_{yz}$ in the center channel, but these do not propagate to the predicted velocities. Additionally, it was observed that obtaining significant improvement in the velocity field requires only a few tens of training steps and only requires the coefficients to have roughly the correct order of magnitude. On the other hand obtaining better agreement of the scalar coefficients took 1–2 orders of magnitude more training steps with diminishing returns in velocity improvement. This shows the importance of using synthetic data to evaluate the ability of a training framework to learn the true underlying model when one exists rather than only comparing the quantities of interest.

3.3. Learning a LEVM from realistic data of flow over periodic hills

As a final test case, a LEVM is trained using sparse velocity measurements from DNS of flow over periodic hills. The DNS data comes from Xiao et al. (2020) who performed DNS of flow over periodic hills of varying slopes. This flow is characterized by a recirculation region on the leeward side of the hill and scaling the hill width (scale factor $\alpha$) modifies the slope and the characteristics of the recirculation regions (e.g. from mild separation for $\alpha = 1.5$ to massive separation for $\alpha = 0.5$). For all flows, the Reynolds number based on hill height and bulk velocity through the vertical profile at the hill top is $Re = 5,600$. The training data consists of four point measurements of both velocity components in the flow over periodic hills with the baseline, $\alpha = 1$, geometry. The two components of velocity are scaled equally in the objective function. The training data and training results are shown in figure 5. The neural network in this case has one input and one output and is pre-trained to laminar flow, $g^{(1)} = 0$. The trained model is a spatially varying LEVM $g^{(1)} = g^{(1)}(\theta_1)$ that closely predicts the true velocity in most of the flow with the exception of the free shear layer in the leeward side of the hills.

To test the extrapolation performance of the trained LEVM, we use it to predict the flow over the other periodic hill geometries, $\alpha \in [0.5, 0.8, 1.2, 1.5]$, and compare them to results with the $k$–$\omega$ model ($g^{(1)} = -0.09$). The results for $\alpha = 0.5$ and $\alpha = 1.5$ are shown in figure 6. For the $\alpha > 1.0$ cases the trained linear model outperforms the $k$–$\omega$ model in the entire flow. For the $\alpha < 1.0$ cases the trained model results in better
Differentiable learning of turbulence models from indirect observations

Figure 6. Comparison of horizontal velocity $u_x$ predictions using the trained eddy viscosity model ($g^{(1)} = g^{(1)}(\theta_1)$) and the $k$–ω model ($g^{(1)} = -0.09$) on two periodic hills geometries, $\alpha = 0.5$ ((a)) and $\alpha = 1.5$ ((b)).

velocity predictions in some regions, particularly the upper channel, while the $k$–ω model results in better velocities in the lower channel.

4. Conclusions

In this paper we present a framework to train deep learning turbulence models using quantities derived from velocity and pressure that are readily available for a wide range of flows. The method was first tested using synthetic data obtained from two traditional closure models: the linear $k$–ω and the Shih quadratic models. These two cases demonstrate the ability to learn the true underlying turbulence closure from measurement data when one exists. The method was then demonstrated using synthetic sparse velocity data derived from DNS simulations of a periodic hill. The trained linear model was used to predict flow over periodic hills of different geometries.

This work demonstrates that deep learning turbulence models can be trained from indirect observations when the relevant sensitivities of the RANS equations are available. With the growing interest in differentiable programming for scientific simulations, it is expected that the availability of gradient information will become more commonplace in scientific and engineering computations, making it more seamless to couple scientific computations with novel deep learning methods.

Acknowledgements

This material is based on research sponsored by the U.S. Air Force under agreement number FA865019-2-2204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Declaration of interests

The authors report no conflict of interest.
REFERENCES


Supplementary material for: End-to-end differentiable learning of turbulence models from indirect observations

Carlos A. Michélén Ströfer† and Heng Xiao‡

(Received xx; revised xx; accepted xx)

1. Adjoint equations

The adjoint equations derived here provide the sensitivity of a cost function $J = J(u, p)$ with respect to the Reynolds stress $\tau$ subject to the constraint that $\tau$, $u$, and $p$ must satisfy the RANS equations. The procedure and notation used here is similar to that in Othmer (2008), who derived the adjoint equations for the sensitivity of the Navier–Stokes equations with Darcy term to the scalar porosity field. For detailed derivation of the boundary conditions, which are identical for both cases, the reader is referred to Othmer (2008).

The RANS equations can be written as $\mathcal{R}(u, p, \tau) = 0$ where $\mathcal{R}(= (R_u, R_p)\top$ and

$$R_u = u \cdot \nabla u - \nu \nabla^2 u + \nabla \cdot \tau + \nabla p - g$$

$$R_p = \nabla \cdot u.$$  

The minimization of the cost function $J$ subject to the RANS constraint $\mathcal{R} = 0$ can be formulated as the minimization of the unconstrained Lagrangian function

$$L = J + \int_{\Omega} (\Psi \mathcal{R}) \, d\Omega,$$

where $\Omega$ is the flow domain and $\Psi = (\hat{u}, -\hat{p})$ consists of four Lagrange multipliers with $\hat{u}$ and $\hat{p}$ referred to as adjoint velocity and adjoint pressure. The negative of the adjoint pressure is used as a Lagrange multiplier so that in the resulting adjoint equations the adjoint pressure plays an analogous role to that of the physical pressure in the RANS equations. Since the velocity and pressure depend on the Reynolds stress, the total variation of $L$ is

$$\delta L = \delta_\tau L + \delta_u L + \delta_p L.$$  

Here we have ignored the variation of the momentum source term $g$, which is correct only when the source term is constant, i.e. it does not depend on the other flow variables. If the momentum source term depends on the other flow variables, e.g. to achieve a prescribed bulk velocity as in the test cases used here, ignoring its variation constitutes an approximation. This is similar to the frozen turbulence approximation common in many adjoint-based optimization works (Dwight & Brezillon 2006) where the variations of the turbulence quantities are ignored. Since the constraint $\mathcal{R}(u, p, \tau)$ is zero everywhere, the Lagrange multipliers can be chosen freely and are chosen such that

$$\delta_u L + \delta_p L = 0,$$  

† Email address for correspondence: cmich@vt.edu
‡ Email address for correspondence: hengxiao@vt.edu
to avoid calculating the sensitivities of the other flow variables \((u, p)\) with respect to the Reynolds stress. Equation 1.4 is the adjoint condition which leads to the adjoint equations and boundary conditions. Equation 1.3 becomes

\[ \delta L = \delta_\tau L, \] (1.5)

which leads to an expression for the desired sensitivity \(\partial L/\partial \tau\) in terms of the adjoint variables.

The variations of \(L\) are obtained as

\[
\begin{align*}
\delta_\tau L &= \int_\Omega \Psi \delta_\tau R d\Omega \\
\delta_u L &= \delta_u J + \int_\Omega \Psi \delta_u R d\Omega \\
\delta_p L &= \delta_p J + \int_\Omega \Psi \delta_p R d\Omega,
\end{align*}
(1.6)
\]

with the variations of the RANS equations \(R\) as

\[
\begin{align*}
\delta_\tau R &= R(u, p, \tau + \delta \tau) - R(u, p, \tau) = (\nabla \cdot \delta \tau, 0) \\
\delta_u R &= R(u + \delta u, p, \tau) - R(u, p, \tau) \\
&= ((u \cdot \nabla)\delta u + (\delta u \cdot \nabla)u - \nu \nabla^2 \delta u, \nabla \cdot \delta u) \\
\delta_p R &= R(u, p + \delta p, \tau) - R(u, p, \tau) = (\nabla \delta p, 0),
\end{align*}
\]

where the higher order term \((\delta u \cdot \nabla)\delta u\) in \(\delta_u R\) was ignored. Using these results, Equation 1.4 becomes

\[
\begin{align*}
\delta_u J + \int_\Omega (\Psi \delta_u R) d\Omega + \delta_p J + \int_\Omega (\Psi \delta_p R) d\Omega \\
&= \delta_u J + \int_\Omega \hat{u} \cdot ((u \cdot \nabla)\delta u + (\delta u \cdot \nabla)u - \nu \nabla^2 \delta u) d\Omega - \int_\Omega \hat{p} (\nabla \cdot \delta u) d\Omega \\
&\quad + \delta_p J + \int_\Omega \hat{u} \cdot \nabla \delta p d\Omega \\
&= 0.
\end{align*}
(1.8)
\]

Integration by parts is used to eliminate the sensitivity of gradients (e.g. \(\nabla \cdot \delta u\)) from the expression. The results of integration by parts are summarized in Table 1, where \(\Gamma = \partial \Omega\) is the boundary of the domain \(\Omega\). Integration by parts is done twice on the term with second order derivative \(\nabla^2 \delta u\) which leaves a first order derivative in the boundary integral. The term \(\int_\Omega \hat{u} \cdot ((\delta u \cdot \nabla)u) d\Omega\) does not require integration by parts, but doing so leads to a more convenient adjoint equation that requires only the primal velocity and not its gradient. Finally, the cost function \(J\) is written in terms of integrals over the interior domain and the boundary as

\[
J(u, p) = \int_\Omega J_\Omega (u, p) d\Omega + \int_\Gamma J_\Gamma (u, p) d\Gamma,
\]

(1.9)

and Equation 1.8 can be written as

\[
\begin{align*}
&\int_\Omega (-u \cdot \nabla \hat{u} - \nabla \hat{u} \cdot u - \nu \nabla^2 \hat{u} + \nabla \hat{p} + \frac{\partial J_\Omega}{\partial u}) \cdot \delta u d\Omega \\
&+ \int_\Omega (-\nabla \cdot \hat{u} + \frac{\partial J_\Omega}{\partial p}) \delta p d\Omega \\
&+ \int_\Gamma ((u \cdot n)\hat{u} + n(\hat{u} \cdot u) + \nu (n \cdot \nabla \hat{u}) - \hat{p} n + \frac{\partial J_\Gamma}{\partial u}) \cdot \delta u d\Gamma \\
&+ \int_\Gamma (-\hat{u} \cdot n + \frac{\partial J_\Gamma}{\partial p}) \delta p d\Gamma \\
&+ \int_\Gamma -\nu (n \cdot \nabla \delta u) \cdot \hat{u} d\Gamma \\
&= 0.
\end{align*}
(1.10)
\]

The volume and boundary integrals must vanish separately and the volume integrals will lead to the adjoint equations while the boundary integrals lead to the boundary conditions.
Differentiable learning of turbulence models from indirect observations

... of the adjoint velocity and zero gradient for the adjoint pressure. On these boundaries, including walls boundaries, the...}

The boundary conditions are obtained from setting the boundary integrals in Equation 1.10 to zero and requiring terms that involve \( \delta u \) and \( \delta p \) to vanish independently as

\[
\int_{\Omega} \left( (u \cdot n) \hat{u} + n(\hat{u} \cdot u) + \nu (n \cdot \nabla \hat{u}) + \hat{\nu} n + \frac{\partial J_{\Omega}}{\partial u} \right) \cdot \delta u \, d\Omega = 0
\]

(1.13a)

\[
\int_{\Gamma} \left( -\hat{u} \cdot n + \frac{\partial J_{\Gamma}}{\partial p} \right) \delta p d\Gamma = 0.
\]

(1.13b)

This can be used to derive the adjoint boundary conditions corresponding to different primal boundary conditions. The resulting adjoint boundary conditions for two typical primal boundary conditions are derived in Othmer (2008) and summarized here. The first is a primal boundary with constant velocity and zero pressure gradient, such as a wall or inlet. The corresponding adjoint boundary conditions are

\[
\hat{u}_t = 0
\]

(1.14a)

\[
\hat{u}_n = -\frac{\partial J_{\Gamma}}{\partial p}.
\]

(1.14b)

for the normal (\( \hat{u}_n \)) and tangential (\( \hat{u}_t \)) components of the adjoint velocity and zero gradient for the adjoint pressure. On these boundaries, including walls boundaries, the
adjoint velocity has a no-slip condition and normal flow dependent on the sensitivity of the cost function to the pressure field. The second is a primal boundary with zero velocity gradient and zero pressure, typical of outlet conditions. The corresponding adjoint boundary conditions are

\[ u_n \hat{u}_t + \nu (n \cdot \nabla) \hat{u}_t + \frac{\partial J}{\partial u_t} = 0 \quad (1.15a) \]

\[ \hat{p} = 2\hat{u}_n u_n + \hat{u}_t \cdot u_t - \nu \nabla_t \cdot \hat{u}_t + \frac{\partial J}{\partial u_n}, \quad (1.15b) \]

where \( \nabla_t \) is the tangential, in-plane, component of the gradient. Equation 1.15a is used to determine the boundary values of the tangential component of the adjoint velocity. Equation 1.15b then gives a relation between the normal component of the adjoint velocity and the adjoint pressure that can be satisfied by enforcing the resulting adjoint pressure.

1.3. Sensitivity

We now derive an expression for the sensitivity of the Laplace function with respect to the Reynolds stress tensor in terms of the adjoint variables. Using the variations in Equation 1.6 and 1.7 and integration by parts (Table 1), Equation 1.5 becomes

\[ \delta L = \int_{\Omega} \hat{u} \cdot (\nabla \cdot \delta \tau) \, d\Omega = \int_{\Omega} -\nabla \hat{u} : \delta \tau \, d\Omega + \int_{\Gamma} \left( \hat{u} \cdot n^\top \right) : \delta \tau \, d\Gamma. \quad (1.16) \]

Based on Equation 1.16 the sensitivity of \( L \) with respect to the Reynolds stress at a point \( x_i \) in the domain is

\[ \frac{\partial L}{\partial \tau_i} = -\nabla \hat{u}_i, \quad (1.17) \]

and with respect to a point \( x_i \) in the boundary it is

\[ \frac{\partial L}{\partial \tau_i} = \hat{u}_i \cdot n_i^\top. \quad (1.18) \]

2. Objective functions

The adjoint method requires the function for which the gradient is sought to be expressible as integrals over the domain \( \Omega \) and boundary \( \Gamma = \partial \Omega \) as

\[ J(u, p) = \int_{\Omega} J_\Omega(u, p) \, d\Omega + \int_{\Gamma} J_\Gamma(u, p) \, d\Gamma. \quad (2.1) \]

This appendix presents how to express the objective function in this manner for some typical experimental measurements.

2.1. Full field or boundary measurements

First, as a straightforward case, we consider full field or boundary measurements such as full field velocity or pressure distribution along a wall boundary. In these cases the objective function, i.e. the discrepancy between predicted and measured quantities, can be directly expressed in the form in equation 2.1. For instance, for full field velocity discrepancy the objective function based on mean squared error is given by

\[ J_\Omega = \frac{1}{2} \| u - u^* \|^2, \quad J_\Gamma = 0, \quad (2.2) \]
where the superscripts * indicates experimental values, and its derivatives are
\[
\frac{\partial J_{\Omega}}{\partial u} = (u - u^*) \quad \frac{\partial J_{\Omega}}{\partial p} = \frac{\partial J_{\Gamma}}{\partial u} = \frac{\partial J_{\Gamma}}{\partial p} = 0. \quad (2.3)
\]
Similarly, for the pressure distribution along a wall the objective function is given by
\[J_{\Omega} = 0 \quad \text{and} \quad J_{\Gamma} = \frac{1}{2}(p - p^*)^2\] for points on the wall boundary and \(J_{\Gamma} = 0\) for points on any other boundaries.

2.2. Sparse and integral measurements

Next, we consider an objective function for the mean squared error discrepancy of more general experimental measurements including sparse measurements such as point velocity measurements or integral quantities such as drag on a solid boundary. The objective function is written as
\[
J = \frac{1}{2} \|d(\tau) - d^*\|_{R^{-1}}^2, \quad (2.4)
\]
where \(d^*\) is a vector of measurements, \(R\) is the measurement’s covariance matrix that captures the measurement uncertainties, and \(\|x\|_W^2 = x^TWx\) indicates the L2-norm of a vector \(x\) with a weight matrix \(W\). The RANS predicted measurements can be written as
\[
d(\tau) = \begin{bmatrix} J^{(1)}, \ldots, J^{(N_d)} \end{bmatrix}^T, \quad (2.5)
\]
with each \(J^{(i)}\) written in the form in equation 2.1. The required sensitivity for the objective function in Equation 2.4 is given by
\[
\frac{\partial J}{\partial \tau} = \left[ R^{-1}(d(\tau) - d^*) \right]^T d'(\tau), \quad (2.6)
\]
where \(i\)th row of the sensitivity matrix \(d'(\tau)\) is given by \(\partial J^{(i)}/\partial \tau\) obtained by solving the adjoint equations with function \(J^{(i)}\). This requires one adjoint solve for each measurement. For the common situation where the measurements are independent, \(R\) is diagonal with entries corresponding to the variance of each measurement \(\sigma_i^2\) and Equation 2.6 can be written as
\[
\frac{\partial J}{\partial \tau} = \sum_{i=1}^{N_d} \frac{1}{\sigma_i^2} \left( J^{(i)} - d_i^* \right) \frac{\partial J^{(i)}}{\partial \tau}, \quad (2.7)
\]
where the discrepancy for each measurement is weighted by the measurement’s variance.

Alternatively, for the purpose of learning a turbulence model one might want to weight different types of measurements differently, e.g. to manually give more weight to regions of rapid change such as in boundary and shear layers or use a more formal importance sampling approach. The weighting in equation 2.7 becomes \(\lambda_i^2/\sigma_i^2\) and the weight matrix in the norm in equation 2.4 becomes \(W = D_\lambda R^{-1}D_\lambda = D_\lambda^2 R^{-1}\), where the vector of weights \(\lambda = [\lambda_1, \ldots, \lambda_{N_d}]^T\) and \(D_\lambda\) is the diagonal matrix with entries \(\lambda\).

We now provide two concrete examples of expressing measurement discrepancies in the form of equation 2.4 and equation 2.5. First, for a single point measurement of \(u_x\),
\[
J^{(i)} = \int_{\Omega} M u_x d\Omega, \quad M(x) = \delta_i(x) \approx \frac{r(x)}{\int_{\Omega} r(x) d\Omega}, \quad (2.8)
\]
where the mask field \(M\) is the Dirac delta function \(\delta_i\) for the measurement location. For the discretized problem the mask can be approximated using a radial function \(r\) centered at the measurement location, and normalized such that the integral of the mask field is
one. As a second example, for the drag on a wall boundary,
\[ J^{(i)} = \int_{\Gamma} M p n \cdot x_d d\Gamma, \] (2.9)
where \(x_D\) is the unit vector in the direction of drag and the mask is one for points on the wall boundary and zero for points on any other boundaries.

3. Simple Cases

3.1. Two-dimensional flows

For statistically two-dimensional flows \((u_z = 0\) and \(\partial / \partial z = 0\)) the first three functions in Equation 2.3 and first two scalars in Equation 2.4 form a functional and a scalar basis.

Furthermore the third tensor basis can be incorporated into the pressure term in the RANS equations, leaving only two tensor functions and two scalar invariants. The non-dimensionalized velocity gradient for the statistically two dimensional case can be written as
\[ t_\tau \nabla u = \begin{bmatrix} a & b & 0 \\ c & -a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \] (3.1)
which results in only two independent scalar invariants
\[ \theta_1 = 2a^2 + \frac{1}{2}(b+c)^2 \]
\[ \theta_2 = -\frac{1}{2}(b-c)^2. \] (3.2)

The other three invariants are \(\theta_3 = \theta_4 = 0\) and \(\theta_5 = \frac{1}{2}\theta_1\theta_2\). This results in the following symmetric and deviatoric tensor basis
\[ T^{(i)} = \begin{bmatrix} T_{11}^{(i)} & T_{12}^{(i)} & \cdots \\ T_{12}^{(i)} & T_{22}^{(i)} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, \]
\[ T^{(1)} = \begin{bmatrix} a & \frac{1}{2}(b+c) & -a \\ \frac{1}{2}(b+c) & \frac{1}{2}b^2 - c^2 & a(b-c) \\ -a & a(b-c) & \frac{1}{2}b^2 - c^2 \end{bmatrix}, \]
\[ T^{(2)} = \begin{bmatrix} \frac{1}{3}a^2 + \frac{1}{12}(b+c)^2 & 0 & 0 \\ 0 & \frac{1}{3}a^2 + \frac{1}{12}(b+c)^2 \end{bmatrix}, \]
with \(T_{13}^{(i)} = T_{23}^{(i)} = 0\). The other tensors in the full 3 dimensional basis become \(T^{(4)} = \frac{\theta_2}{\theta_1} T^{(3)}, T^{(5)} = T^{(10)} = 0, T^{(6)} = \theta_2 T^{(1)}, T^{(7)} = -\frac{1}{2}\theta_2 T^{(2)}, T^{(8)} = \frac{1}{2}\theta_1 T^{(2)}, \) and \(T^{(9)} = \theta_2 T^{(3)}\). Since the problem is statistically two-dimensional, for a quantity \(\phi\) of the form
\[ \phi = \begin{bmatrix} \phi & 0 & 0 \\ 0 & \phi & 0 \\ 0 & 0 & 0 \end{bmatrix}, \] (3.4)
its divergence can be incorporated into the gradient of pressure term as \(\nabla \cdot \phi = \nabla \phi\).

This is the case with the third tensor in the basis, which results in the pressure term
\[ p = \frac{1}{\rho}(p - p_0) + \frac{2}{3}k + \frac{1}{3}k\theta_1 g^{(3)}, \] (3.5)
since \(2k(\frac{1}{3}a^2 + \frac{1}{12}(b+c)^2) = \frac{1}{3}k\theta_1\). For data from synthetic models, when learning the functions \(g^{(i)}\) for \(i \in 1, 2, 3\) what is learned is
\[ \tilde{g}^{(1)} = g^{(1)} + \theta_2 g^{(6)} \]
\[ \tilde{g}^{(2)} = g^{(2)} - \frac{1}{2}\theta_2 g^{(7)} + \frac{1}{2}\theta_1 g^{(8)} \]
\[ \tilde{g}^{(3)} = g^{(3)} + \frac{\theta_2}{32} g^{(4)} + \theta_2 g^{(9)}, \] (3.6)
and similarly the term $\tilde{g}^{(3)}$ can be incorporated into the pressure term.

For two-dimensional flows the RANS equations become

$$u_x \partial_x u_x + u_y \partial_y u_x + \partial_x p - \nu \partial_x^2 u_x - \nu \partial_y^2 u_x + \partial_x \tau_{xx} + \partial_y \tau_{xy} - g_x = 0$$

$$u_x \partial_x u_y + u_y \partial_y u_y + \partial_y p - \nu \partial_x^2 u_y - \nu \partial_y^2 u_y + \partial_x \tau_{xy} + \partial_y \tau_{yy} - g_y = 0$$

$$\partial_x u_x + \partial_y u_y = 0$$

(3.7)

3.2. Turbulent channel flow

The turbulent channel flow (figure 1) is a two-dimensional flow but with $a = b = 0$ and only $c$ is non-zero. Then $\theta_2 = -\theta_1$ and there is only one scalar invariant $\theta_1 = \frac{1}{2} c^2$. The three tensors in the basis become

$$T^{(1)} = [0, \frac{1}{2} c, 0, 0, 0, 0]$$

$$T^{(2)} = [\frac{1}{2} c^2, 0, 0, 0, -\frac{1}{2} c^2, 0]$$

$$T^{(3)} = [\frac{1}{12} c^2, 0, 0, 0, \frac{1}{12} c^2, 0].$$

(3.8)

The RANS equations become

$$-\nu \partial_y^2 u_x + \partial_y \tau_{xy} - g_x = 0$$

$$\partial_y p + \partial_y \tau_{yy} = 0.$$  

(3.9)

Since $\partial_x \tau_{xx} = 0$ the $xx$ term in the basis tensors is inconsequential and any basis with a $yy$ term and zero $xy$ term can be incorporated into the pressure. In this case tensors $T^{(2)}$ and $T^{(3)}$ can be incorporated into the pressure as

$$p = \frac{1}{\rho} (p - p_0) + \frac{2}{3} k + k \theta_1 \tilde{g}^{(2)},$$

(3.10)

with $\tilde{g}^{(2)} = -2g^{(2)} + \frac{1}{3} g^{(3)}$. Although the mapping from non-dimensionlized velocity gradient to Reynolds stress has three functional basis, when using Reynolds stress in the RANS equations there are effectively only two. One, $g^{(1)}$, impacts the velocity while the other, $\tilde{g}^{(2)}$, affects the pressure.

3.3. Flow through a square pipe

Flow through a square pipe (figure 2) is three dimensional, but the problem still has some important symmetries that lead to simplifications. The driving mechanisms for secondary flow are clear when using the mean axial voriticity transport equation instead
Figure 2. Flow through a square duct. Streamlines of the in-plane flow are shown on a cross-section.

of the two in-plane momentum equations (Speziale 1982). For fully developed flow the velocity has no dependence on the axial $x$ direction and the continuity equation becomes $\partial_y u_y + \partial_z u_z = 0$ and therefor a stream function exists such that

$$u_y = -\frac{\partial \Phi}{\partial z}, \quad u_z = \frac{\partial \Phi}{\partial y}, \quad \omega_x = \nabla^2 \Phi, \quad (3.11)$$

where $\omega_x$ is the mean axial vorticity. The RANS equations become these three equations plus the axial momentum and vorticity equations (from the curl of the momentum equations)

$$u_y \frac{\partial u_x}{\partial y} + u_z \frac{\partial u_x}{\partial z} - \nu \nabla^2 u_x - \frac{\partial}{\partial y} \tau_{xy} - \frac{\partial}{\partial z} \tau_{xz} - g_x = 0$$

$$u_y \frac{\partial \omega_x}{\partial y} + u_z \frac{\partial \omega_x}{\partial z} - \nu \nabla^2 \omega_x + \frac{\partial^2}{\partial y \partial z} (\tau_{zz} - \tau_{yy}) + \left(\frac{\partial^2}{\partial y^2} - \frac{\partial^2}{\partial z^2}\right) \tau_{yz}. \quad (3.12)$$

Only four combinations of Reynolds stress components affect the RANS equations: $\tau_{xy}$ and $\tau_{xz}$ in the axial equation and $\tau_{yz}$ and $(\tau_{zz} - \tau_{yy})$ in the in-plane equation (required to produce in-plane velocities).

The velocity gradient for this problem can be written as

$$t \nabla u = \begin{bmatrix} 0 & 0 & 0 \\ a & \alpha & \gamma \\ b & \beta & -\alpha \end{bmatrix}, \quad (3.13)$$

with $a, b \gg \alpha, \beta, \gamma$ since the in-plane velocities are orders of magnitude smaller than the axial velocity. With the simplification $\alpha = \beta = \gamma = 0$ there is only one independent scalar invariant, $\tilde{\theta}_1 = \frac{1}{2} (a^2 + b^2)$ where the tilde indicates the approximation was used.

The other invariants become $\tilde{\theta}_2 = -\tilde{\theta}_1$, $\tilde{\theta}_4 = -\frac{1}{3} \tilde{\theta}_3$ and $\tilde{\theta}_4 = \tilde{\theta}_3 = 0$, and $\tilde{\theta}_5 = \frac{1}{2} \tilde{\theta}_1^2$.

Similarly $\tilde{T}^{(5)} = \tilde{T}^{(10)} = 0$, $\tilde{T}^{(1)}$ and $\tilde{T}^{(6)}$ have $xy$ and $xz$ components but zero $yz$ or $zz-yy$ components and therefor affect only the axial momentum equation, and the other 6 tensors have $yz$ and $zz-yy$ components but no $xy$ or $xz$ and therfor affect only the axial vorticity equation. Denoting $\tilde{T}_{\text{axial}} = [\tilde{T}_{xy}, \tilde{T}_{xz}]$ and $\tilde{T}_{\text{in-plane}} = [\tilde{T}_{yz}, (\tilde{T}_{zz} - \tilde{T}_{yy})]$, ...
we can write
\[
\tilde{T}^{(6)}_{\text{axial}} = -\tilde{\theta}_1 \tilde{T}^{(1)}_{\text{axial}} \\
\tilde{T}^{(4)} = -\tilde{T}^{(3)} \\
\tilde{T}^{(4)}_{\text{in-plane}} = -\tilde{T}^{(3)}_{\text{in-plane}} = 2\tilde{T}^{(2)}_{\text{in-plane}} \\
\tilde{T}^{(9)}_{\text{in-plane}} = \tilde{T}^{(8)}_{\text{in-plane}} = \tilde{T}^{(7)}_{\text{in-plane}} = \tilde{\theta}_1 \tilde{T}^{(4)}_{\text{in-plane}}.
\]

If training a turbulence model with only flows through square pipes and \(a, b >> \alpha, \beta, \gamma\) only two functions can be learned
\[
\tilde{g}^{(1)} = g^{(1)} - \tilde{\theta}_1 g^{(6)} \\
\tilde{g}^{(2)} = g^{(2)} - 2g^{(3)} + 2g^{(4)} + 2\tilde{\theta}_1 g^{(7)} + 2\tilde{\theta}_1 g^{(8)} + 2\tilde{\theta}_1 g^{(9)}
\]

Without that approximation, the only simplification is \(\theta_4 = -\frac{1}{3}\theta_3\) and in general all ten functions can be learned.

4. Shih’s quadratic model

The Shih quadratic \(k-\varepsilon\) model (Shih et al. 1993) for the normalized Reynolds stress anisotropy is given as
\[
b = \sum_{i=1}^{4} \alpha_i B_i \\
\alpha_1 = -\frac{\nu_t}{2k} \\
\alpha_2 = \frac{1}{2} \frac{C_{\mu}}{A_2 + \eta \varepsilon} \frac{k^2}{\varepsilon} \\
\alpha_3 = \frac{1}{2} \frac{C_{\mu}}{A_2 + \eta \varepsilon} \frac{k^2}{\varepsilon} \\
\alpha_4 = \frac{1}{2} \frac{C_{\mu}}{A_2 + \eta \varepsilon} \frac{k^2}{\varepsilon}
\]
\[
B_1 = \partial_j u_i + \partial_i u_j \\
B_2 = (\partial_k u_i)(\partial_j u_k) + (\partial_k u_j)(\partial_i u_k) - \frac{2}{3} \delta_{ij} \\
B_3 = (\partial_k u_i)(\partial_j u_j) - \frac{1}{3} \tilde{\Pi} \delta_{ij} \\
B_4 = (\partial_i u_k)(\partial_j u_k) - \frac{1}{3} \tilde{\Pi} \delta_{ij}
\]
with
\[
A_1 = 1.25 \quad A_2 = 1000 \quad \alpha = 0.9 \\
C_{r1} = -4 \quad C_{r2} = 13 \quad C_{r3} = -2 \quad \nu_t = C_{\mu} \frac{k^2}{\varepsilon} \quad \frac{1}{\sqrt{A_1 + \eta + \alpha \xi}}
\]

and the scalar quantities derived from the velocity gradient are defined as follows:
\[
\eta = \frac{k}{\varepsilon} \sqrt{2S_{mn} S_{mn}} \\
\xi = \frac{k}{\varepsilon} \sqrt{2R_{mn} R_{mn}} \\
\Pi = (\partial_j u_i)(\partial_i u_j) \\
\tilde{\Pi} = (\partial_j u_i)(\partial_j u_i).
\]

To obtain the coefficient functions \(g_i\) that correspond to Shih’s model we seek a change
of basis matrix $C$ such that

$$g = C\alpha. \quad (4.4)$$

This is obtained by first writing one set of basis in terms of the other. The change of basis matrix is (algebra omitted)

$$C = \frac{\varepsilon^2}{k^2} \begin{bmatrix}
\frac{2}{k} & 0 & 0 & 0 \\
0 & 0 & 1 & -1 \\
0 & 2 & 1 & 1 \\
0 & 2 & -1 & -1
\end{bmatrix} \quad (4.5)$$

Each column gives one of Shih’s basis tensor in terms of Pope’s basis tensors, e.g. the second column says that $B_2 = (2\varepsilon^2_k)T_3 + (2\varepsilon^2_k)T_4$.

Carrying out the matrix multiplication $g = C\alpha$ the resulting equations for the scalar coefficient functions are

$$g_1 = 2\varepsilon_k(\alpha_1)$$
$$g_2 = \varepsilon^2_k(\alpha_3 - \alpha_4)$$
$$g_3 = \varepsilon^2_k(2\alpha_2 + \alpha_3 + \alpha_4)$$
$$g_4 = \varepsilon^2_k(2\alpha_2 - \alpha_3 - \alpha_4) \quad (4.6)$$

The parameters $\eta$ and $\xi$ can be expressed as

$$\eta = \sqrt{2\theta_1} \quad \xi = \sqrt{-2\theta_2} \quad (4.7)$$

The scalar function $\theta_2 = \{\tilde{R}^2\}$ is always $\theta_2 \leq 0$. This is because it is the sum of products of anti-symmetric components, i.e. $R_{ij}R_{ji} = -(R_{ij})^2$. Similarly $\theta_1 = \{\tilde{S}^2\}$ is always $\theta_1 \geq 0$ since it is the sum of products of symmetric components. Using the definitions of the coefficients $\alpha_i$ the final expression for the coefficients $g_i(\theta_1, \theta_2)$ is

$$g_1(\theta_1, \theta_2) = -\frac{2}{3} \frac{1}{A_1 + \sqrt{2\theta_1 + \alpha \sqrt{-2\theta_2}}}$$
$$g_2(\theta_1, \theta_2) = \frac{1}{2} \frac{C_{c_2} - C_{c_3}}{A_2 + (2\theta_1)^{3/4}}$$
$$g_3(\theta_1, \theta_2) = \frac{1}{2} \frac{2C_{c_1} + C_{c_2} + C_{c_3}}{A_2 + (2\theta_1)^{3/4}}$$
$$g_4(\theta_1, \theta_2) = \frac{1}{2} \frac{2C_{c_1} - C_{c_2} - C_{c_3}}{A_2 + (2\theta_1)^{3/4}} \quad (4.8)$$

or replacing in the coefficients

$$g_1(\theta_1, \theta_2) = -\frac{2/3}{1.25 + \sqrt{2\theta_1 + 0.9\sqrt{-2\theta_2}}}$$
$$g_2(\theta_1, \theta_2) = \frac{7.5}{1000 + (\sqrt{2\theta_1})^3}$$
$$g_3(\theta_1, \theta_2) = \frac{1.5}{1000 + (\sqrt{2\theta_1})^3}$$
$$g_4(\theta_1, \theta_2) = -\frac{9.5}{1000 + (\sqrt{2\theta_1})^3} \quad (4.9)$$

5. Validation

Before testing the full training framework the adjoint equations and the framework were validated using a simple case for which the expected qualitative behaviour is known. This validation test uses the turbulent channel flow with a Reynolds number of 10,000 based on bulk velocity and half distance $h$. The flow is driven by a constant pressure gradient $g_x$ such that the specified bulk velocity Reynolds number is achieved. The RANS
discretization uses 50 equally-spaced cells along the \( y \)-direction from the bottom wall to the mid-channel with wall and symmetry boundary conditions at the bottom and top, respectively, and cyclic boundary conditions in the \( x \)-direction. For this case only the \( x \)-velocity is non-zero and only the \( y \)-derivatives are non-zero and the RANS equations reduce to

\[
\begin{align*}
-\nu \frac{\partial^2}{\partial y^2} u_x + \frac{\partial}{\partial y} \tau_{xy} - g_x &= 0 \\
\frac{\partial}{\partial y} p + \frac{\partial}{\partial y} \tau_{yy} &= 0,
\end{align*}
\]  

(5.1)

two independent equations for the \( x \)-velocity and pressure. Only the \( xy \)-component of Reynolds stress has an effect on the velocity distribution. For the channel simulation we use the standard \( k-\omega \) equations for the turbulence scales and linear closures, i.e. \( g^{(1)} \) is the only non-zero coefficient function.

As a validation test for the adjoint equations we look at the sensitivity of the velocity at a point to the Reynolds stress field. For this, the single-point objective function is used with radial basis function

\[
r(x) = \exp \left( - \left( \frac{x - x_0}{l} \right)^2 \right),
\]  

(5.2)

with length scale \( l = 0.04h \) or two cell height. Integrating equation 5.1 twice and using the fact that both velocity and Reynolds stress are zero at the wall the velocity distribution is given by

\[
\nu u_x = -\frac{1}{2} g_x y^2 + u_\ast y + \int_0^y \tau_{xy},
\]  

(5.3)

where \( u_\ast \) is the friction velocity. From this we see that the velocity at a point should depend on the \( xy \)-component of Reynolds stress from the wall to that point and have no dependence on Reynolds stress at locations between the point and the mid-channel. The RANS equations were solved for this case using the standard closure \( g^{(1)} = -0.09 \) and the results were used to solve the adjoint equations for the sensitivities of the velocity at different points. Figure 3 shows the sensitivity of the velocity at three points with respect to the Reynolds stress in the entire domain. As expected the velocity depends on the Reynolds stress from the wall up to the point and has zero sensitivity on the
Reynolds stress from the point to the mid-channel. Figure 3 also shows the sensitivity of the full field velocity magnitude, i.e. $J = \int_{\Omega} u_x d\Omega$, which as expected depends on the full Reynolds stress field with the sensitivity being largest at the wall and smoothly decreasing to zero at the mid-channel.

REFERENCES


Othmer, Carsten 2008 A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International journal for numerical methods in fluids* 58 (8), 861–877.


Chapter 3

Ensemble Gradient for Training Data-Driven Turbulence Models
Ensemble Gradient for Training Data-Driven Turbulence Models

Carlos A. Michelén Ströfera,∗, Xin-Lei Zhanga,b,∗, Heng Xiaoa,∗∗

aKevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060, USA
bUniv. Lille, CNRS, ONERA, Arts et Metiers Institute of Technology, Centrale Lille, UMR 9014 - LMFL - Laboratoire de
Mécanique des fluides de Lille - Kampé de Fériet, F-59000 Lille, France

Abstract

Training data-driven turbulence models with high fidelity Reynolds stress can be impractical and recently such models have been trained with velocity and pressure measurements. For gradient-based optimization, such as training deep learning models, this requires evaluating the sensitivities of the RANS equations. This paper explores the use of an ensemble approximation of the gradient of the RANS equations in training data-driven turbulence models with indirect observations. A deep neural network representing the turbulence model is trained using the network’s gradients obtained by backpropagation and the ensemble approximation of the RANS sensitivities. Different ensemble gradient approximations are explored and a method based on explicit projection onto the sample space is presented. As validation, the gradient approximations from the different methods are compared to that from the continuous adjoint. The ensemble approximation is then used to learn different turbulence models from velocity observations. In all cases, the learned model predicts improved velocities. However, it was observed that once the sensitivity of the velocity to the underlying model becomes small, the approximate nature of the ensemble gradient hinders further optimization of the underlying model. The benefits and limition of the ensemble gradient approximation are discussed, in particular as compared to the adjoint equations.

Keywords: ensemble methods, turbulence modeling, deep learning

1. Introduction

The Navier–Stokes equations fully describe the instantaneous velocity and pressure fields in fluid flows. However the resolution required to capture the range of turbulence scales makes solving the Navier-Stokes equations computationally inaccessible for flows with high Reynolds number or complex geometries. Instead, the Reynolds-averaged Navier–Stokes equations (RANS) are widely used in practice thanks to the relatively inexpensive computation required for their solution. The RANS equations are a set of coupled partial differential equations (PDE) that describe the mean velocity \( \mathbf{u} \) and mean pressure \( p \) fields. However, the RANS equations contain the unclosed Reynolds stress tensor \( \mathbf{\tau} \) which captures the effects of turbulence on the mean flow and requires modeling. The incompressible steady RANS equations are

\[
\mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla \cdot \mathbf{\tau} + \nabla p - \mathbf{s} = 0, \\
\nabla \cdot \mathbf{u} = 0,
\]

(1)

where \( \mathbf{s} \) are the external body forces. Compactly, this can be written as \( \mathcal{M} (\mathbf{u}, p; \mathbf{\tau}) = 0 \) where the Reynolds stress \( \mathbf{\tau} \) requires turbulence modeling.

Although widely used, RANS predictions are known to be inaccurate due to the lack of an accurate general turbulence model. In particular, the widely used linear eddy viscosity models (LEVM) are known to

*These authors contributed equally to this work.
**Corresponding author

Email address: hengxiao@vt.edu (Heng Xiao)
URL: https://www.aoe.vt.edu/people/faculty/xiaoheng.html (Heng Xiao)
be inaccurate even in simple flows of practical interest, including the inability to predict secondary flows in non-circular ducts [1]. Eddy viscosity models are single-point closures that represent the Reynolds stress as a local function of the velocity gradient. Non-linear eddy viscosity models can capture more complex non-linear relations between the velocity gradient and the Reynolds stress but existing NLEVM have not resulted in consistent improvement over LEVM. This has led to an interest in developing data-driven turbulence models [2]. Particularly, data-driven NLEVM [3] have recently gained much attention.

Data-driven NLEVM have been typically trained with full field Reynolds stress data from high fidelity simulations. It has been recently recognized, however, that the use of high fidelity Reynolds stress data for training is impractical, which has led to the use of velocity and pressure measurements as training data [4, 5]. This allows for the use of more complex flows for which solutions of the Navier–Stokes equations are not feasible but for which experimental data is available. Training the model using such data has the added complexity of requiring solving the RANS equations at each training step and, for gradient-based optimization, obtaining the gradient of the RANS equations. In this work we explore the use of ensemble-based gradient approximations as an alternative to adjoint models for gradient-based training of data-driven turbulence models from indirect observations.

1.1. Data-Driven eddy viscosity models trained with indirect observations

The representation of the turbulence model and the training framework used here are the same as in MichélonStröfer and Xiao [4] except for the use of the ensemble gradient in place of the adjoint and the reader is referred there for more details than summarized here. A non-linear eddy viscosity turbulence model is an invariant mapping from the mean flow velocity gradient to Reynolds stress anisotropy tensor, \( \nabla u \mapsto \tau \). Any such mapping can be written as

\[
\tau = 2k \sum_{i=1}^{10} g^{(i)} T^{(i)}
\]

where \( T^{(i)} \), \( g^{(i)} \), and \( \theta_j \) are the basis tensor functions, scalar coefficient functions, and scalar input invariants, and \( k \) is the turbulent kinetic energy. The basis tensors and input invariants are

\[
\begin{align*}
T^{(1)} &= S, & T^{(6)} &= R^2 S + S R^2 - \frac{2}{3} \{ S R^2 \} I \\
T^{(2)} &= S R - R S, & T^{(7)} &= R S R^2 - R^2 S R \\
T^{(3)} &= S^2 - \frac{1}{3} \{ S^2 \} I, & T^{(8)} &= S R S^2 - S^2 R S \\
T^{(4)} &= R^2 - \frac{1}{3} \{ R S \} I, & T^{(9)} &= R S^2 R^2 + S^2 R^2 - \frac{2}{3} \{ S^2 R^2 \} I \\
T^{(5)} &= R S^2 - S^2 R, & T^{(10)} &= R S^2 S^2 - R^2 S^2 R,
\end{align*}
\]

and

\[
\begin{align*}
\theta_1 &= \{ S^2 \}, & \theta_2 &= \{ R^2 \}, & \theta_3 &= \{ S^3 \}, & \theta_4 &= \{ R^2 S \}, & \theta_5 &= \{ R^2 S^2 \},
\end{align*}
\]

where \( S \) and \( R \) are the non-dimensionalized symmetric and antisymmetric components of the velocity gradient tensor, \( I \) is the second rank identity tensor. Learning a turbulence model consists of learning the scalar functions \( g^{(i)}(\theta_1, \ldots, \theta_5) \). A deep neural network is used to represent this mapping, \( \theta \mapsto g \).

Training the neural network with velocity and pressure quantities requires the gradient of both the neural network and the RANS equations. The overall training framework is summarized in Figure 1. The gradient of the neural network outputs with respect to its inputs are obtained using backpropagation, an efficient reverse mode automatic differentiation algorithm for deep neural networks. The cost function \( J \) compares the predicted and measured quantities and requires solving the RANS equations. The gradient of the cost function with respect to the trainable parameters \( w \) is given by the chain rule as

\[
\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \tau} \frac{\partial \tau}{\partial \tau} \frac{\partial w}{\partial \tau}.
\]

Here the gradient of cost function with respect to the Reynolds stress is approximated with the ensemble gradient rather than obtained by solving the adjoint equations.
1.2. Differentiation of physical models

Training data-driven models with indirect observations poses a challenge to computing the gradient of the cost function as this now requires obtaining the gradients of the RANS equations. Three options are (i) using gradient-free optimization to avoid the gradient calculation, (ii) using adjoint methods to obtain the exact gradient, or (iii) approximating the gradient from multiple evaluations of the RANS model. Zhao et al. [5] use gene expression programming to train a model in a gradient-free manner. Gradient-free optimization updates an ensemble of states based on a heuristic other than gradient-descent, such as natural selection in genetic algorithms. Michelén Ströfer and Xiao [4] use the continuous adjoint equations to obtain the sensitivities of the RANS equations and train a deep neural network using gradient-based optimization. This requires solving only one additional set of equations at each training step. The adjoint equations provide a method for obtaining the exact gradient but can be difficult to converge and require expressing the cost function as a volume integral which makes it cumbersome to work with diverse types of measurements. Another option for the exact gradient is the discrete adjoint [6]. Like the continuous adjoint, this represents a good choice but it requires a major effort to implement into an existing code. In this work we explore the use of approximate gradient using ensemble gradient methods in place of the exact gradient from the continuous or discrete adjoint. Like the gradient-free approach this requires multiple evaluations of the RANS equations and treats the model as a black box but like the adjoint methods it provides a gradient that can be used in gradient-based optimization.

Different types of approximate gradients have been devised, including the finite difference [7], simplex gradient [8], simultaneous perturbation stochastic approximation (SPSA) [8], and ensemble-based optimization (EnOpt) [9]. These methods differ on the number and selection of samples and how these are used to estimate the gradient. Specifically, finite difference uses the same number of samples, $N$, as dimensions, $D$, and perturbs one orthogonal direction at a time with a fixed perturbation. The simplex gradient method uses a set of $N = D + 1$ samples that are affinely independent and estimates the gradient information based on the relationship between the samples and the centroid. Simultaneous perturbation stochastic approximation uses $N = 2$ samples by simultaneously perturbing all model parameters to estimate the gradient. Further, Li and Reynolds [10] developed a modified SPSA or stochastic Gaussian search direction (SGSD) method which draws multiple ($N < D$) samples from Gaussian distribution and uses the expectation of estimated gradient as the downhill direction. The ensemble gradient method used here also uses $N < D$ random samples from a Gaussian distribution and has the benefit of significantly reducing the number of samples.
model evaluations as compared to finite difference or simplex gradient. The ensemble gradient is more robust and efficient for finding the steepest direction than other approximate gradient methods and can provide conditioned realizations based on random maximum likelihood [10]. Do and Reynolds [11] discuss the connection of ensemble-based optimization to the simplex gradient method and SPSA method. They showed that ensemble-based optimization is equivalent to a preconditioned simplex gradient and to the second-order SPSA with Gaussian sampling. The ensemble-based gradient is also employed implicitly in ensemble-based data assimilation techniques such as ensemble Kalman filtering [12] and ensemble maximum randomized likelihood method [13]. Ensemble-based data assimilation does an implicit optimization, but it essentially uses an explicit approximate stochastic gradient based on the ensemble realizations [14, 11].

1.3. Contribution of present work

Here we demonstrate the use of ensemble gradient to train a data-driven turbulence model. While we use it for turbulence modeling we point out that it could be used for any cost function involving the solutions of PDEs. The ensemble gradient represents a non-intrusive method for obtaining gradients that can treat the PDEs as a black box model and is more flexible with the types of measurement data and cost function. Additionally the ensemble gradient can be robust to non-differentiable or noisy problems as long as there is a well behaved larger trend. This can be important, for example, for the sensitivity of high fidelity turbulent flows whose chaotic nature makes the adjoint method ineffective [15]. Here, we use both the EnOpt method [9] and a method derived here which is based on explicit projection onto the samples. The new method performs just as well but is based on different heuristic.

The rest of the paper is organized as follows. Section 2 presents the different ensemble-based methods, their interpretations, and specialization to the turbulence modeling problem. Section 3 presents the results, starting with a comparison of the computed gradients between different methods, including the adjoint. Then the use of the ensemble gradient for model learning is demonstrated by learning turbulence models from velocity observations. The velocity observations come from synthetic flow solutions with known turbulence model, which provide a ground truth to evaluate the model learning. Two different synthetic models are learned, a linear and a quadratic model. Finally, Section 4 concludes the paper.

2. Ensemble Gradient

The ensemble gradient approximation is characterized by the use of \( N \) random samples, with \( N \) less than the dimensions of the problem. The samples are chosen from a Gaussian process with assumed covariance kernel. This section present different approaches for using ensemble approximation for the gradient in the context of optimization. The direct implementation of gradient descent with ensemble gradient is presented first and the reasons why it fails are discussed. Next, the common approach of preconditioning the gradient descent with the state covariance is presented and the reasons it works are discussed. Lastly, the method used in this work, which is based on direct projection onto the subspace spanned by the samples, is presented. Numerical comparison of these methods will be shown in Section 3.1. This section concludes with a summary of the application of these different methods to the problem of training data-driven NLEVM with indirect observations.

2.1. Direct ensemble gradient

The cost function to be minimized is the least squares discrepancy between the observations and predictions weighted by the observation uncertainty, given as

\[
J(x) = \frac{1}{2} \|z - y\|_{C^{-1}}^2, \tag{6}
\]

\[
z = H(x), \tag{7}
\]

where \( x \) is the state vector, \( y \) is the vector of measurement data, \( z \) is the state mapped to observation space by means of the non-linear observation operator \( H \), \( \| \cdot \|_W^2 \) indicates the L-2 norm weighted by the matrix \( W \),
and $C_y$ is the measurement covariance matrix representing measurement uncertainties. For gradient descent optimization the state is updated as

$$x_{n+1} = x_n - \alpha_n \nabla J(x_n),$$

where $\alpha$ controls the step size and can be a fixed scalar or determined by line search at each step $n$. The gradient of the cost function in Equation 6 is

$$\nabla J(x) = (H'(x))^\top C_y^{-1} (H(x) - y),$$

where $H'(x)$ is the sensitivity matrix of the vector $H(x)$ with respect to the state $x$. The matrix $C_y$ and vector $y$ are fixed while the vector $H(x)$ is obtained by solving the forward (RANS) model. The sensitivity matrix $H'(x)$ is estimated using an ensemble gradient approximation as

$$H'(x) \approx [H(x^{(i)}) - \bar{H}(x)][x^{(i)} - \bar{x}]^+ = \Delta_x \Delta_x^+,\quad (10)$$

where for a quantity $\phi$ the matrix $\Delta \phi = [\phi^{(1)} - \bar{\phi}, \ldots, \phi^{(N)} - \bar{\phi}]$ consists of the mean-subtracted samples, $\phi^{(i)}$ indicates the $i$th sample, $\bar{\phi}$ is the ensemble mean, and the superscript + indicates the pseudoinverse. However, the inversion problem described by Equation (6) is ill-posed in that multiple states $x$ can result in $H(x) = y$ or at least up to high accuracy of the latter. This inherent ill-posedness can result in cost function and sensitivity matrix with frequent, abrupt changes for small changes in the state $x$. Even for well-posed problems the use of the pseudoinverse of the ensemble matrix can lead to noisy approximate gradients, as will be demonstrated later. For this reason the direct ensemble gradient is not a suitable option for gradient descent optimization or model learning as pursued in this work.

### 2.2. Preconditioning with state covariance

Pre-multiplying the gradient $\nabla J(x)$ by the state covariance $C_x$ is a common method for preconditioning of the gradient descent optimization, as

$$x_{n+1} = x_n - \alpha_n C_x \nabla J(x_n),$$

referred to as EnOpt [9]. This correspond to steepest descent rather than gradient descent on a discrete vector space with inner-product defined by the covariance $C_x$ [16]. The use of the state covariance as a preconditioner for the gradient descent is discussed in more detail in Appendix B. The ensemble approximation of the state covariance is given as

$$C_x \approx \frac{1}{N} \Delta_x \Delta_x^T, \quad (12)$$

and the product of the state covariance and the forward model sensitivity can be approximated using the ensemble as

$$C_x H'(x) \approx \frac{1}{N} \Delta_x \Delta_x^T (\Delta_y \Delta_y^+)^T = \frac{1}{N} \Delta_x \Delta_x^T (\Delta_y^+)^+ \Delta_y^T = \frac{1}{N} \Delta_x \Delta_y^T \approx C_{xz}, \quad (13)$$

where $C_{xy}$ is the cross-covariance between the state and model outputs. This formulation avoids taking the pseudoinverse of an ensemble matrix, which in general can be ill-conditioned. Empirical results show that while the ensemble gradient in Equation (10) is noisy, the ensemble cross-covariance $C_{xy}$ captures the correct qualitative correlations [17]. Finally, the update scheme in Equation (11) becomes

$$x_{n+1} = x_n - \alpha_n C_{xz} C_y^{-1} (H(x_n) - y),$$

where $C_{xz}$ is approximated with the ensemble, based on Equation (13) as

$$C_{xz} \approx \frac{1}{N} \Delta_x \Delta_x^T. \quad (15)$$
2.3. Projection to subspace

The method used here is based on explicit projection of the state onto the subspace spanned by the samples in the ensemble, similar to the formulation in the ensemble-variational (EnVAR) data assimilation method [18]. The ill-posedness of the inverse problem in Equation (6) is alleviated by means of dimensionality reduction and regularized projection. The state is expressed as \( x = \bar{x} + \Delta_x \beta \) where \( \Delta_x \) is the matrix of mean subtracted samples and the vector \( \beta \in \mathbb{R}^N \) is the new state with reduced dimensions. That is, a state \( x \) is expressed as the ensemble mean plus a linear combination of mean-subtracted samples. The cost function in Equation (6) now has gradient, with respect to the new state \( \beta \), given as

\[
\nabla_{\beta} J(x(\beta)) = \Delta_x^T (H'(x))^{-1} C_y^{-1} (H(x(\beta)) - y)
\]

\[\approx \Delta_x^T (\Delta_x \Delta_x^+)^T C_y^{-1} (H(x(\beta)) - y) \]

\[= \Delta_x^T C_y^{-1} (H(x(\beta)) - y), \tag{16}\]

where the model discrepancies \( \Delta_y \) are obtained from the ensemble. While this gradient could be used to optimize the new state \( \beta \), for training the turbulence model the gradient of the cost function with respect to the original state \( x \) is required. For this, we seek the gradient of the new state \( \beta \) with respect to the original state \( x \). For a state \( x \), the new state \( \beta \) can be obtained by projecting \( x - \bar{x} \) onto the mean-subtracted samples as

\[
\beta_i = \frac{\langle x - \bar{x}, \Delta_i^{(i)} \rangle_w}{\langle \Delta_i^{(i)}, \Delta_i^{(i)} \rangle_w} = (\Delta_x^T W \Delta_x + \lambda I)^{-1} \Delta_x^T W (x - \bar{x}), \tag{17}\]

where \( W \) is the weight matrix in the definition of vector inner product \( \langle \cdot, \cdot \rangle_w \). The gradient is then

\[
\nabla \beta = (\Delta_x^T W) \Delta_x^{-1} \nabla \beta = (\Delta_x^T W \Delta_x + \lambda I)^{-1} \Delta_x^T W, \tag{18}\]

It is noted that the matrix \( \Delta_x^T W \Delta_x \) is not related to the state covariance \( C_x \approx \frac{1}{N} \Delta_x \Delta_x^T \) and is often ill-conditioned as the samples are randomly drawn and not orthogonal. A convenient approach to converting the matrix to well-conditioned is Tikhonov regularization which results in

\[
\beta_i = (\Delta_x^T W \Delta_x + \lambda I)^{-1} \Delta_x^T W (x - \bar{x}), \tag{19}\]

and

\[
\nabla \beta = (\Delta_x^T W \Delta_x + \lambda I)^{-1} \Delta_x^T W, \tag{20}\]

where \( I \) is the identity matrix and \( \lambda \) is the Lagrange multiplier of the constraint which can typically be very small. The details are presented in Appendix A.

Equation (16) and Equation (20) can be combined by the chain rule to obtain the desired gradient as

\[
\nabla J(x) = ((\Delta_x^T W \Delta_x + \lambda I)^{-1} \Delta_x^T W)^T (\Delta_x^T C_y^{-1} (H(x) - y)), \tag{21}\]

where \( \Delta_y \) and \( \Delta_x \) are both obtained from the ensemble. The ensemble state discrepancy \( \Delta_x \) could be fixed throughout the optimization, and only the forward model ensemble be updated \( \Delta_x \) using the same ensemble discrepancy but new mean \( \bar{x} = x_n \). Here, however, we follow the common practice in EnVar and resample the ensemble state at each iteration. Similarly, we use the true mean \( \bar{x} = x_n \) and \( H(x) = H(\bar{x}) \) rather than the ensemble mean for the ensemble discrepancies and denote these by \( \Delta \) and \( \Delta_H \). Finally the gradient is given by

\[
\nabla J = \left( (\Delta_x^T W \Delta_x + \lambda I)^{-1} \Delta_x^T W \right)^T \left( \Delta_x^T C_y^{-1} (H(x) - y) \right), \tag{22}\]

\[45\]
2.4. Ensemble gradient for RANS equations

In the cost function is given by Equation (6), the state is the discretized values of the Reynolds stress \( x = \tau \) at each cell, and the forward model \( \mathcal{H} \) is the composition of the RANS equations which map Reynolds stress to velocity and pressure \( \tau \mapsto u, p \) and an observation operator that maps velocity and pressure fields to observations (e.g. sparse sampling, surface integration). Regardless of which method is used, the ensemble gradient requires solving the RANS equations for the neural network predicted Reynolds stress \( \mathcal{H}(\tau) \) as well as creating an ensemble of Reynolds stress fields \( (\Delta \tau) \) and solving the RANS equations for each sample in the ensemble \( (\Delta \tau) \). The gradient used for each of the three methods are summarized in Table 1. For the projection method, since the state is a discretization of a field, the matrix \( W \) defining the inner product is the diagonal matrix with cell volumes, \( D_V \).

<table>
<thead>
<tr>
<th>method</th>
<th>( \partial J / \partial \tau )</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct</td>
<td>((\Delta \tau \Delta \tau^\top)^\top \mathcal{C}_\tau^{-1} (\mathcal{H}(\tau) - y))</td>
</tr>
<tr>
<td>preconditioned</td>
<td>(\frac{1}{N} \Delta \tau \Delta \tau^\top \mathcal{C}_\tau^{-1} (\mathcal{H}(\tau) - y))</td>
</tr>
<tr>
<td>projection</td>
<td>((\tilde{\Delta} \tau^\top D_V \tilde{\Delta} \tau + \lambda I)^{-1} \tilde{\Delta} \tau^\top D_V \mathcal{C}_\tau^{-1} (\mathcal{H}(\tau) - y))</td>
</tr>
</tbody>
</table>

Table 1: Ensemble gradients used for the different methods. All quantities of the form \( \Delta \phi \) or \( \tilde{\Delta} \phi \) are ensemble matrices.

Creating the ensemble of states requires perturbing the Reynolds stress tensor, which poses the question of how to best perturb a tensor field. This has been done in previous works by perturbing the turbulence kinetic energy (magnitude), eigenvalues (shape), and eigenvectors (direction) of the Reynolds stress tensor [19]. The formulation of the Reynolds stress in Equation (2), however, provides a convenient way of creating how to best perturb a tensor field. This has been done in previous works by perturbing the turbulence kinetic energy (magnitude), eigenvalues (shape), and eigenvectors (direction) of the Reynolds stress tensor \( \mathcal{C}_\tau \).

\[ K(x_i, x_j) = \exp \left(-\frac{1}{2} \frac{||x_i - x_j||^2}{l^2} \right) \]

where \( x_i \) and \( x_j \) are the spatial coordinates of cell \( i \) and \( j \), \( l \) is the correlation length, and \( \alpha \) is the ratio of the standard deviation to the magnitude of \( g^{(i)} \).

3. Results

The ensemble gradient is first validated by comparing the ensemble gradient, using different methods, to the gradient obtained from the adjoint equations for a turbulent channel flow. The ensemble gradient is then used in place of the adjoint to reproduce the results in the two synthetic cases in Michelen Ströfer and Xiao [4]. The use of synthetic data provides a ground truth to which to compare the learned models. The first case is a turbulent channel flow with a linear closure model. The second case is flow through a square duct with a quadratic model. For all cases, a neural network with 10 hidden layers and 10 neurons per hidden layer is used. A ReLU activation is used for hidden layers and linear for the output layer. The ADAM algorithm is used for the gradient-based training.
3.1. Validation

For validation, the fully developed turbulent channel flow is used. The RANS simulation domain includes the bottom half of the channel, with a symmetry boundary condition at the mid-channel, and a discretization of 50 cells of equal sizes. The Reynolds number, based on bulk velocity and channel half height \( H \), is 10,000, and the linear \( k-\omega \) model is used for the truth. First, the estimated sensitivity of the different ensemble methods are compared to that obtained from the adjoint equations. In this case the cost function is the full field discrepancy of the velocity \( u_x \), i.e.

\[
J(\tau) = \| u_x(\tau) - u_x^* \|_{D_V}^2, \tag{25}
\]

where \( u_x^* \) is the true velocity field and \( D_V \) contains the cell volumes in the diagonal. Figure 2 shows the sensitivity of the cost function with respect to the Reynolds stress field for \( \tau = 0 \) corresponding to laminar velocity. It can be seen that the direct ensemble method gives a noisy estimate on the sensitivity. In contrast, the ensemble method with either the projection method or the precondition method can provide smooth estimation of the sensitivity and give the same gradient direction as the adjoint method. This suggests that either the projection or preconditioned ensemble methods can be used for gradient-based training. It is noted that the exact values of the gradient are not expected to be the same, but as long as it has the right sign and has the correct zero any estimate of the gradient can be used for training. One reason for the discrepancy is that the adjoint equations give the gradient of the Laplacian function that includes the cost function and the RANS constraints [4]. This is in addition to the other approximations in both the adjoint and ensemble methods.

Before using the ensemble gradient to train a neural network, which has 1021 parameters, it is validated using a simplified one parameter turbulence model. That is, \( g^{(1)} = -C_\mu \) is treated as a constant and the true value is taken as \( C_\mu = 0.09 \). The gradient is evaluated for a range of values of \( C_\mu \) using both the ensemble and adjoint methods. The results are shown in Figure 3, and it can be seen that although they are noisy and have different magnitude the ensemble gradients results in the same sign and same zero as from the adjoint in the search region near the true value. The ensemble gradients with only 20 samples can be seen to be noisy but still were sufficient for correctly training the channel case (see Section 3.2). This observation further confirms that the ensemble gradient can be used for training turbulence models.

3.2. Learning a linear model

The channel case is used to learn the linear turbulence model. The neural network has one input \( \theta_1 \), one output \( g^{(1)} \), and 1,021 parameters. The true solution is simply \( g^{(1)} = -0.09 \) which is constant and has no
dependence on the input, but this is not enforced and must be learned. The neural network is pre-trained to a constant \(g^{(1)} = -0.05\) rather than to \(g^{(1)} = 0.0\) since the later would result in samples with \(g^{(1)} > 0\) which are nonphysical and tend to result in diverging RANS. The network is then trained using the ADAM algorithm with default parameters. Figure 4 shows the results of the training including the initial and final samples used to estimate the gradient. The trained model not only results in the correct velocity but learns the true underlying model for \(g^{(1)}\).

Figure 4: Results of learning a linear model from velocity data of the channel flow. The samples are used to estimate the gradient. The wall normal coordinate is indicated by \(y\) with \(y = 0.5\) the center of the channel and \(y = 0\) and 1 the bottom and upper walls, respectively.
3.3. Learning a quadratic model

The second test case consists of using the full velocity field in a flow in a square duct to learn a quadratic turbulence model. The non-linear model is the Shih quadratic $k-\varepsilon$ [20] given by

$$
\begin{align*}
g_1(\theta_1, \theta_2) &= -\frac{2/3}{1.25 + \sqrt{2}\theta_1 + 0.9\sqrt{-2\theta_2}} \\
g_2(\theta_1, \theta_2) &= \frac{1.5}{1000 + (\sqrt{2}\theta_1)^3} \\
g_3(\theta_1, \theta_2) &= \frac{1.5}{1000 + (\sqrt{2}\theta_1)^3} \\
g_4(\theta_1, \theta_2) &= \frac{-0.5}{1000 + (\sqrt{2}\theta_1)^3}
\end{align*}
$$

However, for the square duct case only $g^{(1)}$ and the combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$ affect the velocity, and $\theta_1 \approx -\theta_2$. Therefore a neural network with one input and two outputs is used. The neural network is pre-trained to the linear model, $g^{(1)} = -0.09$ and $g^{(2)} = 0$. The training is first done with a small learning rate of $10^{-5}$ for initial stability of the optimization scheme, and later increased to the default value of $0.001$.

Figure 5 shows the velocity prediction of the trained model. The trained model predicts the in-plane velocity, which the linear models fail to predict.

The trained model did not learn the underlying Shih quadratic model. This is because only a small modification of the coefficients is needed to achieve improved velocity predictions. Consequently, the sensitivity of the velocity to the underlying model becomes very small. Michelén Ströfer and Xiao [4] noted that learning the correct velocity with the adjoint gradient required only a few tens of training steps, while getting a better agreement in the underlying model took 1–2 order of magnitude more steps. Unlike the adjoint, the ensemble gradient here represents only an approximation of the gradient, which is not accurate enough to continue training once the sensitivity of velocity to the underlying model becomes small.

4. Conclusion

In this paper an ensemble approximation of the gradient was used to obtain the sensitivities of the RANS equations during training of a data-driven turbulence model with indirect observations. Through the use
of simple one parameter validation case it was shown that although the different ensemble approximations and the adjoint produce different gradients, they share the same zero-gradient location, sign, and qualitative behavior. This case consisted of learning the scalar $C_\mu$ coefficient in a linear turbulence model from observations of synthetic velocity in a turbulent channel flow. The same channel flow velocity was used to successfully train a deep neural network to learn a linear model. A second test case consisted of using synthetic velocity observations from flow in square duct using non-linear model. The learned model was able to predict the velocities, including the in-plane secondary velocity, but did not learn the correct underlying turbulence model. This is due to the gradient approximation not being accurate enough to continue training once the sensitivity of velocity becomes small. This is not expected to be an issue, however, when training more realistic models with a wide range of flows rather than with a single flow where the peculiarities of a single flow would have less weight.

The present methodology can be used for any case where a neural network (or other differential machine learning models that provides its own gradients) is trained using quantities that require propagating the output of the network through another forward model. The use of the ensemble gradient requires multiple evaluations of the forward model (e.g. RANS equations) at each step, but allows for it to be treated as a black box model. On the other hand deriving and implementing the continuous or discrete adjoint can require significant overhead. Additionally, the ensemble gradient can handle diverse types of measurement data more easily than the adjoint, where the cost function must be expressed as an integral over the domain and different types of measurements (e.g. line integral of pressure) cannot be easily combined. With the ensemble gradient these different measurements can all be combined into a single linear observation operator $\mathcal{H}$. Finally, it is noted that the ensemble gradient does not require the cost function to be locally differentiable as long there are clear global trends. For these reasons, in some cases the approximate ensemble gradient might be preferable to the exact gradient from the adjoint method.

Acknowledgements

Michelén Ströfer is supported by the U.S. Air Force under agreement number FA865019-2-2204 while performing this research. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Appendix A. Regularized Projection

The projection of the state $x$ onto the space spanned by the sample discrepancies, described in Section 2.3, is equivalent to minimization of the following objective function on $\beta$:

$$J_b = \frac{1}{2} \| \Delta x \beta - (X - \bar{X}) \|_W^2$$  \hspace{1cm} (A.1)

We regularize the problem with Tikhonov regularization as

$$\arg \min_{\beta} J_b(\beta) = \frac{1}{2} \| \Delta x \beta - (X - \bar{X}) \|_W^2 + \lambda \frac{1}{2} \| \beta \|^2$$  \hspace{1cm} (A.2)

where $\lambda$ is the regularization strength parameter and can be chosen as small as possible while still providing the desired regularization. The derivative of the objective function with respect to $\beta$ can be formulated as

$$\nabla_{\beta} J_b = \Delta_x^\top W (\Delta_x \beta - (X - \bar{X})) + \lambda \beta$$  \hspace{1cm} (A.3)

By setting the derivative equals to zero and solving for $\beta$, we have

$$\beta = (\Delta_x^\top W \Delta_x + \lambda I)^{-1} \Delta_x^\top W (X - \bar{X}).$$  \hspace{1cm} (A.4)

The effectiveness of the regularization is demonstrated by looking at the condition number of the matrix to be inverted. The condition number of $\Delta_x^\top W \Delta_x$ with and without regularization is presented in Fig. A.6.
for the channel case in Section 3.2. The condition number if shown for ensembles with different number of samples and a regularization parameter of $\lambda = 10^{-8}$ is used. It can be seen that without regularization the matrix $\Delta_x^\top W \Delta_x$ has very large condition number and is ill-conditioned even with only 2 samples. With regularization, the matrix can be well conditioned with small condition numbers.

**Appendix B. State Covariance as a Preconditioner**

The preconditioned gradient descent in Equation (11) represents a steepest descent on a discrete state space with inner product defined by the state covariance matrix. That is, the direction of steepest descent depends on the choice of norm in the state space [16]. The norm defines an infinitesimal circle and the steepest descent direction is towards the point in the circle that takes the minimum value of the objective function. In general for L2 norms the steepest direction is only aligned with the gradient direction when the matrix defining the inner product is the identity matrix. Otherwise, for an inner product $\langle a, b \rangle = a^\top P b$ the steepest direction is simply $P$ times the gradient direction, as in Equation (11). The use of the steepest direction can be seen as preconditioner or a regularization that results in smoother gradient approximation.

Alternatively, the use of the state covariance as a preconditioner can be derived from the quasi-Newton method, with some rough approximation, and a regularized cost function. The regularized cost function is

$$\tilde{J}(x_n) = \frac{1}{2} \| H(x_n) - y \|_{C_y^{-1}}^2 + \frac{1}{2} \| x_n - x_{n-1} \|_{C_x^{-1}}^2. \quad (B.1)$$

It has also been shown that, for the same regularized cost function, the continuous-time limit of the ensemble Kalman inversion behaves as the preconditioned gradient descent in Equation (11). These two cases are summarized next.

**Appendix B.1. From quasi-Newton method**

A typical way to derive a linear preconditioner for a gradient descent problem is using an approximation of the operator appearing in the quasi-newton algorithm. Often, even a crude approximation works well [16]. The state covariance as the preconditioner can be obtained in this manner with some approximations [21]. Using the regularized cost function in Equation (B.1), the gradient and Hessian are

$$\nabla \tilde{J}(x_n) = (H'(x_n))^\top C_y^{-1} (H(x_n) - y) + C_x^{-1} (x_n - x_{n-1}) \quad (B.2)$$

$$\nabla^2 \tilde{J}(x_n) = C_x^{-1} + (H'(x_n))^\top C_y^{-1} H'(x_n) + (H''(x_n))^\top C_y^{-1} (H(x_n) - y), \quad (B.3)$$
where $\nabla^2$ indicates the outer product ($\nabla \otimes \nabla$) indicating the Hessian. For the quasi-Newton method the second derivative term $\mathcal{H}''$ is dropped,

$$\nabla^2 \tilde{J}(x_n) \approx C_x^{-1} + (\mathcal{H}'(x_n))^\top C_y^{-1} \mathcal{H}'(x_n)$$  \hfill (B.4)

and the update scheme is

$$x_{n+1} = x_n + \alpha_n \left( \nabla^2 \tilde{J}(x_n) \right)^{-1} \nabla \tilde{J}(x_n).$$ \hfill (B.5)

A crude approximation is obtained by treating the term $G(x) = \|\mathcal{H}(x) - y\|_C^{-2}$ without consideration to its form. The Hessian then becomes

$$\nabla^2 \tilde{J}(x_n) \approx C_x^{-1} + G'' \approx C_x^{-1}.$$ \hfill (B.6)

The update scheme becomes

$$x_{n+1} = x_n + \alpha_n (x_n - x_{n-1}) + \alpha_n C_x (\mathcal{H}'(x_n))^\top C_y^{-1} (\mathcal{H}(x_n) - y),$$ \hfill (B.7)

which when ignoring the previous step term $(x_n - x_{n-1})$ becomes the original update scheme preconditioned by the state covariance (Equation (11)).

### Appendix B.2. From ensemble Kalman inversion

The ensemble Kalman inversion (EKI) is a method for general problem inversion based on iterative application of the ensemble Kalman filter (EnKF). The EKI is commonly used in non-linear data-assimilation problems. The state is augmented to include the observations $\mathcal{H}(x)$ and the problem is reformulated as an artificial dynamics problem where all non-linearity's are moved to the dynamic model. The problem can be formulated as a linear EnKF problem on the augmented state, but can also be re-expressed in terms of the non-linear operators and the original state [22]. The EKI solves the regularized inverse problem in Equation (B.1) implicitly. The gradient-free update for each sample is given as

$$x_{n+1}^{(i)} = x_n^{(i)} + C_{xz} (C_{z} + C_y)^{-1} (\mathcal{H}(x_n) - y)$$

$$= x_n^{(i)} + \frac{1}{N} \Delta x_n \Delta z_n^\top \left( \frac{1}{N} \Delta z_n \Delta z_n^\top + C_y \right)^{-1} (\mathcal{H}(x_n) - y)$$ \hfill (B.8)

where $j$ indicates the sample index and the superscript $n$ indicates the iteration step. Schillings and Stuart [23] show that in the continuous time limit ($\frac{1}{N} \to 0$) the evolution of a sample in the EKI is

$$\frac{dx^{(i)}(t)}{dt} = -C_{xz} C_y^{-1} (\mathcal{H}(x) - y).$$  \hfill (B.9)

That is, the continuous time limit of the EKI has an update direction consistent with preconditioning the original gradient with the state covariance (Equation (11)).

### References


14

53
Part II

Field Inversion
Chapter 4

DAFI: An Open-Source Framework for Ensemble-Based Data Assimilation and Field Inversion
COMPUTATIONAL SOFTWARE

DAFI: An Open-Source Framework for Ensemble-Based Data Assimilation and Field Inversion

Carlos A. Michelén Ströfer, Xin-Lei Zhang, Heng Xiao

Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, USA.

Received 9 September 2020; Accepted (in revised version) 7 December 2020

Abstract. In many areas of science and engineering, it is a common task to infer physical fields from sparse observations. This paper presents the DAFI code intended as a flexible framework for two broad classes of such inverse problems: data assimilation and field inversion. DAFI generalizes these diverse problems into a general formulation and solves it with ensemble Kalman filters, a family of ensemble-based, derivative-free, Bayesian methods. This Bayesian approach has the added advantage of providing built-in uncertainty quantification. Moreover, the code provides tools for performing common tasks related to random fields, as well as I/O utilities for integration with the open-source finite volume tool OpenFOAM. The code capabilities are showcased through several test cases including state and parameter estimation for the Lorenz dynamic system, field inversion for the diffusion equations, and uncertainty quantification. The object-oriented nature of the code allows for easily interchanging different solution methods and different physics problems. It provides a simple interface for the users to supply their domain-specific physics models. Finally, the code can be used as a test-bed for new ensemble-based data assimilation and field inversion methods.

AMS subject classifications: 35R30, 76M21, 60-04

Key words: Data assimilation, inverse modeling, random fields, ensemble Kalman filter, Bayesian inference.

Program summary

Program title: DAFI

Nature of problem: This software performs ensemble-based, derivative-free, Bayesian inference of physical fields from sparse observations.

*Corresponding author. Email addresses: cmich@vt.edu (C. Michelén Ströfer), xinleizhang@vt.edu (X.-L. Zhang), hengxiao@vt.edu (H. Xiao)
1 Introduction

Inverse problems in physical systems take many forms, and two broad classes—data assimilation and field inversion—are considered here. Data assimilation [1] refers to a class of inverse problems where a dynamic model is available and time-dependent observations are used to infer some property of a dynamic system. An example of a data assimilation problem is inferring the temperature field of a heated solid at the current time by using both sparse observations of the temperature (i.e. at a few locations) and a model forecast of the entire field. The model forecast could be obtained from propagating the temperature field at an earlier time using the diffusion equation. Here, Field inversion problems refer to a class of inverse problems where two sets of fields are related to each other through a forward model and observations of the output fields are used to infer the input fields. With the heat diffusivity example, a field inversion problem is inferring the material diffusivity from steady-state temperature measurements, where the heat diffusion equation is the forward model relating a diffusivity field to a temperature field. The approach taken here is to formulate both data assimilation and field inversion problems within a general framework of inverse problems and solve them by using ensemble Kalman filtering methods [2]. This is possible since field inversion problems can be recast as artificial dynamics problem and solved iteratively by using data assimilation procedures [3]. The main intended application for our code, which we named DAFI, is solving field inversion problems described by partial differential equations (PDE), a common type of problem in science and engineering. The code has several features that reflect this emphasis on PDE-based inversion problems, including: (i) ensemble-based solution approaches which are non-intrusive, requiring no gradients from and no code
modification to the physics solver, which can therefore be treated as a black-box model, (ii) separation of statistical inference and physical modeling via object-oriented programming that allows the user to focus only on the physics problem at hand, and (iii) modules that facilitate working with discretized random physical fields.

DAFI is particularly geared to inverse problems involving fields, which are continuous-valued quantities over some domain. For example, the authors have used DAFI to infer the Reynolds stress tensor field from sparse observations of the velocity field in the Reynolds-averaged Navier–Stokes equations [4, 5], and to solve for the porosity field in problems involving fluid flow through porous media [6]. To facilitate working with fields, DAFI includes modules that perform common field operations, such as calculating the norm of a discretized field, generating covariance matrices, or performing a modal decomposition of a random field. The code also includes a module for reading and writing fields for OpenFOAM, a widely-used open-source finite volume PDE-solver.

All methods in DAFI are ensemble Kalman filtering approaches, which are particularly suited for large scientific and engineering problems due to their derivative-free and Bayesian nature. A derivative free approach is advantageous, since obtaining the gradient of a cost function with respect to input fields in complex science and engineering models is usually a non-trivial task. It would necessitate either deriving the adjoint equations and implementing an adjoint solver or implementing an intrusive discrete adjoint into the main code. This is even infeasible for commercial solvers where the source code is not accessible. Another benefit of ensemble-based methods is the use of a Monte Carlo representation for all probability distributions. This becomes a necessity for complex fields with large discretization since manipulating and propagating large covariance matrices through physical models quickly becomes intractable. Finally, ensemble Kalman filters are fully Bayesian and result in an estimate of the posterior distribution, which is particularly useful in applications that require quantifying uncertainty in the inferred quantities or fields.

DAFI is implemented in an object-oriented manner with every problem requiring two objects: a statistical method for the inference and a physics model describing the particular problem being solved. This separation allows the code to be useful to both physics domain experts trying to solve a particular problem and to algorithm developers allowing them to test their statistical methods. For the physics domain experts DAFI has a simple mechanism to provide the problem-specific physics model. This physics model could be as simple as a non-intrusive wrapper around a third party or commercial solver. Such a user should have domain-specific knowledge but little knowledge is required in the statistical methods used for field inversion. Unlike the models describing the physics, the statistical methods are problem agnostic, and the code provides a library of such methods. For the algorithm developer DAFI can act as a test-bed for testing new ensemble-based inversion methods. For example, the authors have used DAFI as an algorithmic test-bed to incorporate soft constraints from prior or physical knowledge through a novel regularized ensemble Kalman filter [7], to enforce boundary conditions during field inversion [8], and to enforce additional PDEs through physics-informed co-
The remainder of this introduction provides more details on data assimilation and field inversion, introduces uncertainty quantification problems, and provides a review of other existing codes for data assimilation. Section 2 presents the general problem formulation and shows how each of the problems presented here can be formulated in this framework. In Section 3 the statistical methods used to solve the general problem are presented. The code implementation is described in Section 4. The treatment of random fields in the DAFI code is described in Section 5. Section 6 presents several test cases to illustrate the use of the DAFI code. Finally, Section 8 concludes the paper.

1.1 Data assimilation in dynamic systems

In DAFI, field inversion problems are solved by using data-assimilation procedures iteratively and data assimilation techniques are therefore important for solving both data assimilation and field inversion problems. This subsection introduces data assimilation problems and relevant nomenclature. Fundamentally, data assimilation consists of inferring the state and/or parameters of a dynamic system by using a dynamic model and observations. Using the heat diffusion example, the observations can be spatially sparse temperature measurements (i.e. only at a few locations) at different times while the dynamic model is the diffusion equation. The observations and model can be used to infer the state (e.g. temperature) at a given time, the initial or boundary conditions, or some constant model parameter (e.g. diffusivity). Data assimilation problems are usually classified as either state estimation or parameter estimation, where the state is dynamically varying (e.g. temperature) and parameters are for instance model constants or material properties (e.g. diffusivity). However, the same solution techniques work for both state estimation and parameter estimation problems. In the case where both the state and parameters are being inferred, the approach is to combine both into an augmented state [10]. Hereafter, state shall refer to all quantities being inferred, which may include the true state, model parameters, or both. In problems involving fields, the continuous fields must be discretized to be included in the state vector to be inferred. Alternatively, reduced order modeling, e.g. based on modal decomposition, can be used to represent the field with a finite set of mode coefficients. Section 5 describes the treatment of fields in the DAFI code.

The state estimation problem can be solved in different ways, two of which are filtering and smoothing. In the filtering approach the state vector is corrected each time an observation is made by using the observations and the model forecast. Starting from the initial condition, the dynamic model is used to propagate the state to the first time at which there are observations. The observations are then used to correct the forecast state to an analysis state using a linear filter. Starting with the analysis state as the new initial condition, the dynamic model is used to propagate the state to the next time at which there are observations. This is illustrated in Fig. 1. The smoothing approach is characterized by the use of data at future times to infer an earlier state. There are differ-
ent smoothing approaches and the smoothing problem can be formulated as a sequential method similar to the filtering problem [11]. Both types of problems are common in data assimilation and in principle the DAFI code is well suited for solving smoothing problems but the code has not yet been used for such problems. The rest of this paper focuses on the filtering problem, whose formulation is also used to solve the field inversion problem iteratively.

1.2 Field inversion

Field inversion problems consist of inferring some input fields from (possibly sparse) observations of output fields, where the input and output fields are related through a non-linear \textit{forward} model. Like the dynamic model earlier, the forward model generally consists of a system of coupled PDEs. Field inversion problems can be solved using data assimilation techniques iteratively. The iterations are required to account for the non-linearity of the forward model since data assimilation filtering techniques assume a linear mapping between the state and observations. This is presented in more detail in Section 3.1.1.

The main advantage of inferring the input fields over inferring the output fields directly is that the physics described by the forward model is enforced on the output field. Additionally, often times the forward model captures the correct physics but there are uncertainties in some input fields. As an example, Xiao et al. [4] infer the Reynolds stress field from sparse observations of the velocity field in computational fluid dynamics simulations. The forward model consists of the Reynolds-averaged Navier–Stokes (RANS) equations which propagate a Reynolds stress field to a velocity field. While the RANS
simulations capture the correct physics, the Reynolds stress field requires modeling and
is the largest source of uncertainty in RANS simulations. Inferring Reynolds stress tack-
les the real source of uncertainty and ensures the output fields (velocity, pressure) satisfy
the physics described by the RANS equations.

1.3 Uncertainty quantification

In some cases simply updating the state or parameters based on observations is not
enough, and an estimate of the confidence in these new estimates is also sought. The
methods implemented in DAFI are ensemble-based Bayesian approaches and hence al-
ways result in a full posterior distribution. Therefore, in DAFI, all data assimilation and
field inversion problems are also uncertainty quantification problems. For uncertainty
quantification, one could for example use the mean of the posterior distribution as the
updated state and the covariance as a measure of its uncertainty.

Bayesian updating provides a posterior probability distribution from a combination
of a prior distribution and a likelihood derived from the observations. As an example, if
the diffusivity parameter in the heat diffusion problem is being inferred, we could specify
the prior distribution—our belief and confidence on the value of diffusivity prior to see-
ing the data—based on the mean and standard deviation of the material (e.g. steel). The
measured values (e.g. point-measurements of temperature) and instrumentation’s un-
certainty determine a likelihood distribution for the state, that is, the probability that the
observed measurements would result from a given diffusivity parameter. The Bayesian
update then produces a posterior distribution giving the probability of any value of dif-
fusion given the observations. The mean of the posterior distribution can be used as the
updated diffusivity and the variance as a measure of its uncertainty.

1.4 Review of existing data assimilation codes

A number of free, open-source codes have been developed for data assimilation and are
reviewed here. The two common approaches to solving data assimilation problems are
ensemble data assimilation, based on the ensemble Kalman filter, and variational data
assimilation, a gradient-based approach. The National Center for Atmospheric Research
(NCAR) created the DART [12] code as a community testbed to try out different data
assimilation methods. This platform is the most developed and widely used platform for
ensemble data assimilation. It is used both in research and in deployed operational cases
and includes good parallelisation options. The code is frequently updated to include
the latest methods in the literature. The main downside of DART is the steep learning
required from the user. In many science and engineering applications a light-weight code
would be more accessible to physics domain experts who are not necessarily experts in
data assimilation or statistical inference.

OpenDA [13] is another open-source toolkit for data assimilation. It has both en-
semble and variational data assimilation methods, as well as methods for uncertainty
analysis and model calibration. One of the driving philosophies of the project is to be a platform for researchers to share and exchange code and ideas. As such, the code is object-oriented and users can share both new models and new data-assimilation techniques for use by the wider community. OpenDA has found many applications in ocean engineering, with the code coupled to popular codes such as OpenFOAM and SWAN. OpenDA is similar to DART in that it is very general, powerful, and well-suited for large deployment scale problems. As a result, it shares the same drawback as DART, the steep learning cost associated with it. For instance, OpenDA has language interfaces with C/C++, Java, and Fortran. While these are good options for large scale deployment, usually engineering research and exploration happens in more user-friendly interpreted languages such as Python or Matlab. The use of compiled languages can be a major overhead to many researchers.

DATEs [14] is another code developed at Virginia Tech as a testbed for new data assimilation algorithms. Its implementation shares many similar philosophies with DAFI, including using Python object-oriented programming to easily switch between models or data assimilation techniques. DATEs is capable of using both ensemble and variational methods. In addition to different data assimilation methods, DATEs has the ability to switch between different linear algebra and time integration routines. Many of these features are incorporated because of the focus on testing and evaluating the performance of different algorithms. DATEs is more accessible to a naive user than DART but still includes some complexity due to its generality and focus on algorithm testing.

The main distinctive attributes of our code are its specialization to ensemble methods, its ease of use, and its field operations. These attributes reflect the focus on engineering and physics applications and make DAFI an accessible code for solving diverse problems by physics domain experts who are less versed in statistical inference. Specific examples of this focus include the modal decomposition of random fields for reduced order modeling and the ability to account for unstructured meshes when performing field calculations. These features of the code were chosen based on the authors’ experience with field inversion in diverse problems, including turbulence modeling, tsunami-induced sediment transport, vegetation modeling in coastal engineering, and flow through porous media. Admittedly, the other codes reviewed here are capable of using more general solution approaches than DAFI, but this generality, however, comes at the cost of simplicity and ease of use. For instance, by specialising in ensemble methods, DAFI always requires the same small set of functions from any physics model. This requires little understanding of data assimilation techniques from the part of the user, and makes it simple to couple their physics model. Similarly, because of the specialization to ensemble methods, adding new ensemble methods is straightforward. Ensemble methods are generalized in DAFI to consists of two loops, a time marching outer loop and an inner loop that iterates at the same time, a structure that can encompass all ensemble methods the authors are aware of. This structure and functionalities such as checking convergence and saving intermediate results are already implemented in a general class and adding a new method requires only overriding the update scheme.
2 General problem formulation

A general problem is formulated here to encompass all the different classes of problems described above. The problem is implemented as a filtering approach but with two distinguishing features: (1) the observation operator is separated from the filter and allowed to be non-linear, and (2) the analysis step is done iteratively. This means the solver consists of two loops: an outer time-marching loop and an inner iteration loop. The inner loop is used to deal with non-linearity either in the observation operator (e.g. field inversion problem), in the dynamic model (e.g. the ensemble randomized maximum likelihood method for highly nonlinear models), or both. The process is summarized in Fig. 2 and Algorithm 1, which are further described below. The details of the ensemble-based Bayesian solution approach for this general problem are deferred until Section 3.

Figure 2: Schematic of the general problem formulation. Starting with the forecast state at the first data observation time $x_f^1$, the state is first updated via data assimilation (grey box) to the analysis state at the same time $x_a^1$. The analysis state is then forecast to the next observation time $x_f^2$ via the dynamic model $M$ (white box). This sequential data assimilation and forecasting process continues for each observation time. In the data assimilation step (grey box), the forecast state $x_f^i$ is updated to the analysis state $x_a^i$ through the iterative update scheme that includes the nonlinear observation operator $H$ and the linear filter $A$. The linear filter $A$ takes three inputs: the forecast state $x_f^i$, the same state mapped to observation space $z_f^i$, and the observations at the current data assimilation time $y$ (not shown). The dynamic model and observation operators shown in blue are implemented in the user-defined physics model while the filter shown in red is chosen as one from the library contained in the DAFI Inverse module. The same color scheme is used in Algorithm 1.
mapped to observation space is given by
\[ z = \mathcal{H}(x) \in Y. \]  (2.1)

The analysis step is given by
\[ x^a = A(x^f, z^f, y), \]  (2.2)

where \( x^f \) and \( x^a \) are the forecast and analysis states, and \( A : (X \times Y \times Y) \to X \) is the linear filter. An iteration is added where the analysis state becomes the forecast step. This is indicated by the dashed lines in Fig. 2, and the loop is repeated until some convergence criteria are met. Different filtering techniques and how they fit within this general formulation will be discussed in Section 3. After the filtering loop is complete, the dynamic model is used to propagate the state to the next observation time as
\[ x_{i+1} = M(x_i). \]  (2.3)

Here, \( x_i \) indicates the state at the time corresponding to the \( i \)th observation and the dynamic model has its own time-stepping scheme to take the state from \( x_i \) to \( x_{i+1} \). This procedure is summarized in Algorithm 1.

**Algorithm 1 General Problem Formulation**

1: procedure DATAASSIMILATION\( (x^f_1, \{(t_i, y_i)\}_{i=1}^{N_t}) \)  \( \triangleright \) outer loop
2: for \( i = 1 \) to \( N_t \) do
3: \( x^a_i \leftarrow \text{Analysis}(x^f_i, y_i) \)  \( \triangleright \) analysis state
4: \( x^f_{i+1} \leftarrow \text{DynamicModel}(x^a_i, t_i, t_{i+1}) \)  \( \triangleright \) forecast state
5: end for
6: end procedure
7: 
8: procedure ANALYSIS\( (x_i, y_i) \)  \( \triangleright \) inner loop
9: while not converged do
10: \( z_i \leftarrow \text{ObservationOperator}(x_i) \)
11: \( x_i \leftarrow \text{Filter}(x_i, z_i, y_i) \)
12: end while
13: return \( x_i \)
14: end procedure

The different problems described in Section 1 can be described using this general formulation. These are summarized in Table 1 and described in the following list:

**Filtering.** The inner loop is not used and the outer loop is advanced by the dynamic model.

**Field Inversion.** The outer loop is not used and the inner loop iterations are used to account for the non-linearity of the problem. In this case the state to be inferred
is the input field, and the observation operator consists of two components: the forward model that maps from input field (state) to output field (observable state) and an observation operator on the observable state. For observable state \( u \in U \), forward model \( \mathcal{F} : X \mapsto U \), and observation operator on the observable state \( \mathcal{H}_u \), the observation operator is given by \( \mathcal{H}(x) = \mathcal{H}_u(\mathcal{F}(x)) = \mathcal{H}_u(u) \).

While it is useful to consider each type of problem separately, they all fit into the general formulation where the outer loop deals with time marching and data-assimilation steps, the inner loop is used to address any non-linearity, and the general observation operator maps from state space to observation space. In previous works, the authors have used the DAFI code for classical filtering, field inversion, and uncertainty quantification problems, and these are showcased in Section 6.

3 Ensemble-based analysis methods

In the general problem above, the analysis step consists of combining the model prediction (forecast state) and the observations to obtain an improved prediction (analysis state). The approach taken is a Bayesian implementation, where both the model prediction and the observation are considered random vectors. The state vector is now a random vector possibly consisting of discretized random fields. The goal is then to obtain the probability distribution of the analysis state. In the language of Bayesian inference, the forecast state is the prior probability distribution \( P(x) \), the observations imply a likelihood distribution \( P(y | x) \) for any proposed state vector, and the analysis state is the posterior probability distribution \( P(x | y) \). The resulting posterior distribution is given by Bayes’ formula

\[
P(x | y) \propto P(x)P(y | x),
\]

where the constant of proportionality ensures a total probability of one, i.e. \( \int_x P(x | y) = 1 \). The derivation of the ensemble Kalman filter assumes a Gaussian distribution for all random vectors (Gaussian process for random fields), even though admittedly the distributions do not remain Gaussian after propagation through a non-linear dynamic model.

Random vectors with Gaussian distributions are completely characterized by a mean vector and a covariance matrix. For our general problem, the Bayesian formulation re-
quires the forecast state to consist of a mean value and a covariance matrix. For steady-state problems this means defining a prior distribution, that is a prior mean and prior covariance matrix. For dynamic systems this can be obtained by defining the initial condition as a Gaussian distribution, and propagating this distribution to the first data assimilation time using the dynamic model. The analysis step then modifies this propagated (forecast) distribution using Bayes’ theorem. This updated (analysis) distribution is then propagated again to the next data assimilation time, and so on. The observations at each time are also considered Gaussian distributions, with mean equal to the measurement values and variances obtained from the accuracy of the measuring instruments. Observations are typically considered independent of each other resulting in a diagonal covariance matrix.

With a Gaussian assumption and linear observation operator $H$, Bayes’ formula results in a Gaussian posterior distribution given by the following mean and covariance [15]

$$\mathbf{x}_a = \mathbf{x}_f + K(\mathbf{y} - H\mathbf{x}_f), \quad \text{(3.2a)}$$
$$\mathbf{C}_a = (\mathbf{I} - KH)\mathbf{C}_f, \quad \text{(3.2b)}$$

where the posterior is relabeled as the analysis state $\mathbf{x}_a = \mathbf{x} | \mathbf{y}$ and the prior is relabeled as the forecast state $\mathbf{x}_f = \mathbf{x}$. Overlines denote the mean value of the distribution, $\mathbf{C}_x$ denotes the covariance matrix of the state vector, and $K$ is the Kalman gain matrix given by

$$K = \mathbf{C}_f H^\top \left( HC_f H^\top + \mathbf{C}_y \right)^{-1}, \quad \text{(3.3)}$$

where $\mathbf{C}_y$ is the covariance matrix of the observations. This procedure is the Kalman filter [15], a common data assimilation technique. The Kalman filter is also the basis for more complex methods, e.g., the extended Kalman filter (EKF) [16] where the mean state is propagated with the full nonlinear model instead of with the tangent linear model, and the unscented Kalman filter (UKF) [17] where the mean state and covariance are estimated from a set of propagated samples which are selected based on the unscented transform. In the Kalman filter, the posterior mean takes into account not only the mean of the prior and observations but also the confidence in each as defined by their covariance.

Problems involving fields have very large state vectors, proportional to the discretization size. Because of this, the state covariance matrix becomes computationally unmanageable and a direct application of the Kalman filter is unfeasible. Ensemble data assimilation methods are Monte Carlo approaches where all probability distributions are represented with a finite set of samples. The modified distributions (e.g. propagated in time, or after the analysis step) are then described by the sample mean and sample covariance of the modified samples. This procedure is illustrated in Fig. 3. A direct application of the Kalman filter using an ensemble results in the ensemble Kalman filter (EnKF) [10], a common ensemble method and the basis for more complex ensemble methods [18–20].
ensemble methods it is common to perturb the observations for each sample as well [21]. For the \( j \)th sample this is given as

\[
y^{(j)} \sim N(y, C_y).
\] (3.4)

DAFI has a library of different ensemble-based Bayesian methods. The subsections in this section present a brief theoretical background for the different methods which are currently implemented.

Field inversion problems which involve a non-linear operator can be solved by iterative use of the EnKF. Even for data assimilation problems, if the dynamic model is highly non-linear, each data assimilation step can be solved iteratively. Iterative methods require a convergence or stopping criteria and the correct choice of criteria can affect the method's performance [3, 22]. Currently, DAFI implements two convergence criteria in addition to a specified maximum number of iterations. Both convergence criteria are based on the norm of the average (over all samples) misfit, which at iteration \( l \) is given as

\[
g(l) = \frac{1}{N} \left\| \mathcal{H}(x) - y \right\|_l.
\] (3.5)

The first criteria is based on the iterative residual of the norm of the misfit. At the first iteration that this value decreases by less than a specified relative amount \( \varepsilon \) the iterations are stopped, that is, iterations are stopped when

\[
g(l-1) - g(l) \leq \varepsilon g(0).
\] (3.6)

The second stopping criteria is based on the discrepancy principle [3, 22, 23]. The idea behind the discrepancy principle is that once the noise for the misfit is within the same
order of magnitude as the noise level of the observation data, the data provides no more information. Continuing with data assimilation could lead to fitting the noise of the data (over-fitting). The discrepancy principle implemented in DAFI is based on the form used by Schillings and Stuart [23], and consists of stopping the iterations when

$$g(t) \leq \tau \sqrt{\text{trace}(C_y)},$$

(3.7)

for a $\tau \geq 1$. Here the square root of the trace of the observation covariance is the expected value of the norm of the observation noise [23].

### 3.1 Ensemble Kalman filter (EnKF)

The ensemble Kalman filter (EnKF) is a Monte Carlo implementation of the Kalman filter described above [15]. For the filtering problem in Fig. 1, the initial forecast state is replaced with an ensemble of $N_s$ samples of the state vectors. This ensemble is created, for instance, by choosing a mean and covariance for the initial conditions at $t_0$ and sampling the initial condition using a Gaussian distribution. Each sample is then propagated with the dynamic model to the first data assimilation time $t_1$. The analysis step then consists of updating each sample individually as

$$x_a^{(j)} = x^{(j)} + K(y^{(j)} - z^{(j)}),$$

(3.8a)

$$z = Hx,$$  

(3.8b)

where the Kalman gain is as in Eq. (3.3) but with the sample covariance matrix as an estimate for the true covariance matrix. Using the definition of sample covariance, the Kalman gain matrix can be written as (see Appendix A)

$$K = C_{xz}(C_x + C_y)^{-1},$$

(3.9)

where $C_{xz}$ is the covariance matrix between vectors $x$ and $z$. This avoids constructing the large covariance matrix $C_x$ for the state vector. This formulation also avoids explicitly constructing the matrix $H$ by using the vector $z = Hx$ directly. Each updated sample $x_a^{(j)}$ is then propagated with the dynamic model to the next data assimilation time step ($t_2$) and so on. As presented here, the EnKF solves the filtering data assimilation problem. The next subsection shows how to use the EnKF for field inversion problems.

#### 3.1.1 Iterative EnKF

Problems involving a nonlinear observation operator can be recast as a linear problem using an augmented state and artificial dynamics. Iglesias et al. [3] used this approach for field inversion problems by recasting the problem as an artificial dynamics problem. The augmented state vector $\tilde{x}$ consists of the state and the state mapped to observation space

$$\tilde{x} = \begin{bmatrix} x \\ H(x) \end{bmatrix} = \begin{bmatrix} x \\ z \end{bmatrix}.$$  

(3.10)
The artificial dynamics model is given by

\[ \tilde{x}_{i+1} = \tilde{M}(\tilde{x}_i) = \begin{bmatrix} x_i \\ H_i(\tilde{x}_i) \end{bmatrix}. \] (3.11)

The observation operator is given by \( H = [0 \ I] \). The problem is then solved iteratively as shown in Fig. 2 using the EnKF as in Eqs. (3.8)-(3.9) with the iterations acting as pseudo-time. The problem has been recast as an artificial dynamic problem with non-linear dynamic model and linear observation operator which allows for the direct use of the Kalman filter. However, with these definitions for the dynamic model, augmented state vector, and observation operator, the resulting Kalman update can be rewritten in terms of the original state vector (see Appendix B). The resulting equations are exactly Eq. (3.8a) and Eq. (3.9) but with

\[ z = H(x). \] (3.12)

The iterative EnKF is not implemented as a separate method from the EnKF, rather the user simply specifies a number of inner loop iterations or convergence criteria. The standard EnKF corresponds to a single inner loop iteration. For all methods the user-provided physics model is queried for the values of \( z \) which can be obtained by first running a non-linear forward model. When used for non-linear inversion this iterative EnKF approach is sometimes referred to as ensemble Kalman inversion (EKI).

### 3.2 Regularized EnKF (REnKF)

Inverse problems are often ill-posed with many different states resulting in satisfactory agreement with the observations. This necessitates the introduction of regularization techniques into ensemble methods to further constrain the inference process. This is a common technique in variational approaches, but one the traditional EnKF lacks. Zhang et al. [7] address this complication by augmenting the EnKF to allow general regularization terms in its implied cost function, resulting in the regularized EnKF (REnKF). This method is capable of regularizing the inference with additional constraints to enforce a-priori knowledge or preferred behavior.

A general constraint function on the state \( x \) enforcing some desired property can be expressed as

\[ \| \mathcal{G}(x) \|_W = 0, \] (3.13)

where \( \mathcal{G} \) is a constraint function and \( W \) is weight matrix defining the norm to minimize. The update scheme of the regularized EnKF is given as

\[ \hat{x}^{f(i)} = x^{f(i)} - C_x \mathcal{G}'(x^{f(i)}) W \mathcal{G}(x^{f(i)}), \] (3.14a)

\[ \hat{z}^{f(i)} = H \hat{x}^{f(i)}, \] (3.14b)

\[ x^{a(i)} = \hat{x}^{f(i)} + K \left( y^{(i)} - \hat{z}^{f(i)} \right), \] (3.14c)
where $G'(x)$ is the derivative of the constraint function $G$ with respect to the state $x$. Eq. (3.14a) represents a correction based on enforcing the constraint while Eq. (3.14c) is the standard Kalman update correction based on matching the observations. The regularized EnKF can also be used for field inversion problems in the same manner as the EnKF, by recasting the problem as an artificial dynamics and solving iteratively.

### 3.3 EnKF-MDA

In certain scenarios EnKF leads to overcorrection on the state variables in early iterations. This is the case for instance in schemes where the dynamic model (or artificial dynamics) is highly nonlinear and the prior is far from the truth. This is due to the EnKF performing the Gauss–Newton update with a full step where the system model is linearized and the averaged sensitivity matrix is used [24]. To address this issue, a common approach is to damp the update at early iterations. Motivated by this, the ensemble Kalman filter with multiple data assimilation (EnKF-MDA) [19] was proposed to reduce the update amount in each iteration by inflating the observation error covariance. EnKF-MDA performs a single Kalman update but in multiple steps where the same data is used in each step. This corresponds to re-expressing Bayes’ formula in Eq. (3.1) as

$$P(x \mid y) \propto P(x) \prod_{l=1}^{N_{\text{mda}}} P(y \mid x_l)^{\frac{1}{\alpha_l}}. \tag{3.15}$$

The EnKF-MDA has been shown to outperform EnKF for nonlinear applications [25, 26]. The EnKF-MDA is an iterative method with update scheme given as

$$\begin{align*}
  x_{l+1}^{(j)} &= x_l^{(j)} + \hat{K} \left( \hat{y}^{(j)} - z_l^{(j)} \right), \tag{3.16a} \\
  \hat{y} &= y + \sqrt{\alpha_l} \epsilon, \tag{3.16b} \\
  \hat{K} &= C_{xz} (C_z + \alpha_l C_y)^{-1}, \tag{3.16c} \\
  z_l^{(j)} &= H x_l^{(j)}, \tag{3.16d}
\end{align*}$$

where $l$ denotes the sub-iteration index in one data assimilation window, $\alpha_l$ is the inflation parameter at the $l^{th}$ iteration step, $\epsilon \sim \mathcal{N}(0, C_\epsilon)$ is the observation noise, and $\hat{K}$ is the Kalman gain matrix with inflated observation error. The inflation parameters are chosen such that

$$\sum_{l=1}^{N_{\text{mda}}} \alpha_l = 1, \tag{3.17}$$

where $N_{\text{mda}}$ is a specified number of data assimilation iterations. One option is to make the inflation parameters constant, i.e. $\alpha_l = \alpha$ which results in $\alpha = N_{\text{mda}}$. Alternatively, the inflation parameters $\alpha_l$ can be self-adaptive based on user-defined criteria [27]. The EnKF-MDA is currently implemented in DAFI using a constant inflation parameter. For linear models, results with different inflation parameters converge to the same solution.
and only differ in the number of iterations. For nonlinear cases, a large inflation parameter is suggested to damp the correction and alleviate the effects of the model nonlinearity. Like for the iterative EnKF, this iterative procedure is done in the inner DAFI loop.

3.4 EnRML

The ensemble randomized maximum likelihood (EnRML) [18, 28] method was initially proposed by Gu et al. [18] for strongly nonlinear systems. Randomized maximum likelihood (RML) is a technique that randomizes the likelihood function and converts the maximum a posteriori estimate to a minimization of an objective function which is solved with an optimization technique, e.g., Gaussian-Newton algorithm. EnRML introduces the ensemble technique into RML to estimate the complete posterior distribution. Similar to the EnKF-MDA, the EnRML is an iterative method that can damp the change of the nonlinear model with the Gaussian-Newton algorithm and thus alleviate the effects of the nonlinearity. It introduces an iteration which is done in the inner loop of DAFI. The update scheme of the EnRML method can be formulated as

\[
x^{(j+1)}_l = \gamma x^{(j)}_0 + (1-\gamma)x^{(j)}_l - \gamma C^{x,0}_l \left( R + (Z'_l)^\top C^{x,0}_l Z'_l \right)^{-1} (z^{(j)}_l - y^{(i)} - Z'_l (x^{(j)}_l - x^{(j)}_0)),
\]

(3.18a)

\[
z^{(j)}_l = H x^{(j)}_l,
\]

(3.18b)

where \(\gamma\) controls the step length of the Gaussian-Newton update, \(x_0\) and \(C^{x,0}\) are the initial state covariance in one data assimilation window, and \(Z'_l\) is the sensitivity matrix. The sensitivity matrix is defined by the relationship

\[
[z_l - \bar{z}_l] = Z'_l [x_l - \bar{x}_l],
\]

(3.19)

where \([z_l - \bar{z}_l]\) and \([x_l - \bar{x}_l]\) denote the matrices of mean subtracted samples. The matrix \([x - \bar{x}]\) is non-full, and its inverse is estimated based on singular value decomposition. The step length parameter \(0 < \gamma \leq 1\) can be determined by a standard line search. In this code, it is a user-specified constant. If \(\gamma = 1\) the iteration performs a full Gaussian-Newton update, while if \(\gamma = 0\) no update is performed. For values between \(0 - 1\) the Gaussian-Newton update is damped. For strongly nonlinear systems, a small \(\gamma\) is recommended to prevent the overcorrections at early iterations. The update scheme assimilates observation data to optimize the state \(x\) iteratively, starting from the initial prior distribution \(x_0\). The iterations are stopped once the convergence criteria or maximum iteration number is reached.

4 Implementation

DAFI is implemented in Python, is available for download from PyPI [29] (through the pip command) and has online documentation through Read the Docs [30]. The active
development repository is hosted in GitHub [31]. DAFI consists of a Python module and an executable. The DAFI module can be loaded with the Python command `import dafi` and run with the `dafi.run(<inputs>)` method. The inputs include the name of the inverse method implemented in `dafi.inverse`, the path to the physics model file, the number of samples, the data assimilation times (outer loop), the maximum number of iterations at each data assimilation time (inner loop), and two dictionaries containing the required inputs for the chosen inverse method and physics model. Alternatively the executable can be used to run DAFI from the command line using an input file as `dafi <inputfile>`. DAFI is implemented in an object-oriented manner with two main classes: one corresponding to the physics model and another to the inversion method (e.g. EnKF). Running DAFI requires an instance of each. The inverse method object can be selected as one from the provided library of methods, while the user is required to provide a case specific physics model. As such DAFI serves as a robust framework for diverse inverse problems involving fields, with the solution method or problem physics easily exchanged. The only user requirement is a physics model that follows a prescribed structure. This serves as an API connecting the DAFI solver and the user-specific problem. The physics model can be entirely implemented in Python or it can be simply a wrapper for existing solvers such as commercial tools. A simple physics model, implemented completely in Python, is shown in Appendix D as an example of writing such a code for a user-specific problem. Fig. 4 shows an overview of DAFI’s structure. The following two subsection describe the two classes in more detail.

4.1 PhysicsModel class

The physics model is problem-specific and is provided by the user as a class called `Model` which should be a subclass of the provided `dafi.PhysicsModel`. The required inputs, attributes, and methods for the `Model` class are briefly described here. Several example models and tutorials are provided within the GitHub repository and in the documenta-
tion for reference. The model is responsible for creating the initial distribution of state samples, propagating a state in time, mapping a state to observation space, and providing the observations and observation errors at each data assimilation time step. As such it is an API through which the user provides all the problem specific information. The required methods are the following:

- **generate_ensemble** - Creates the initial ensemble of state samples. This is a Monte Carlo representation of the prior distribution as discussed in Section 3.

- **forecast_to_time** - Forecasts the states to the next data assimilation time. This corresponds to the dynamic model in Eq. (2.3).

- **state_to_observation** - Maps the state to observation space. This implements the nonlinear observation operator in Eq. (2.1).

- **get_obs** - Provides the observations and observation error at the current data assimilation time step. These are discussed in Section 2.

These methods correspond to different tasks described in Section 2. Particularly, the dynamic model **forecast_to_time** and the nonlinear observation operator **state_to_observation** are shown in blue in Fig. 2 and Algorithm 1. The example code in Appendix D shows the physics model for the scalar problem presented in Section 6.5, showing a simple implementation of each of the methods described above.

### 4.2 InverseMethod class

The **InverseMethod** class represents a specific ensemble-based inverse method (e.g. EnKF). The different methods discussed in Section 3 are implemented as child classes of the **InverseMethod** class and are available through the `dafi.inverse` module. For most applications the user simply chooses one of the implemented methods. The **InverseMethod** class implements all the details of the general problem described in Section 2. This includes the outer time loop, inner iteration loop, and stopping criteria. Missing from the **InverseMethod** class however is any specific analysis scheme (i.e. filter A), which is implemented through the `analysis()` method of child classes. The specific methods (e.g. EnKF, EnRML) are implemented as child classes of **InverseMethod** and inherit the `solve` method but implement their own `analysis` method. Creating such a child class is also the recommended approach for creating new inverse methods. The object-oriented and open source nature of DAFI makes it straightforward to implement new inverse methods.

### 5 Fields

DAFI was built with fields in mind. Its core implementation however has no concept of fields, just state and observation vectors. When a problem has a state vector that includes
discretized fields it is up to the user to account for this when creating the problem-specific physics model. DAFI includes two python modules to aid in creating the physics model: dafi.random_fields and dafi.random_fields.covariance. The dafi.random_fields module includes functions for tasks such as calculating the Karhunen-Loève decomposition, calculating the norm of a field, and calculating the projections of a field onto a set of basis. It also includes classes representing stochastic processes with classes with methods to do tasks such as generating samples. Currently two such classes for Gaussian and lognormal processes are implemented. The dafi.random_fields.covariance submodule contains functions to perform tasks related to covariance and correlation matrices, including creating covariance matrices based on some implemented covariance kernels such as the square exponential kernel. The relevant theory is presented in this section and the physics models for the tutorials serve as simple examples on how to use these modules when creating a user-specific physics model.

When dealing with fields, solving the dynamic and forward models requires discretizing the domain. All fields are then represented by the finite vector of their values at each cell. These discretized values would then be included in the state vector. From the point of view of the DAFI code, there is no distinction between state vectors that include discretized fields and ones that do not. The notion of a physically continuous field, however, does come into play in several user specified inputs. Specifically, the appropriate norm for a field quantity at time \( t_i \) is defined by the inner product between two fields \( f_1 \) and \( f_2 \) as

\[
\langle f_1(\xi), f_2(\xi) \rangle = \int_{\Omega} f_1(\xi)f_2(\xi)d\Omega \approx \sum_{j=1}^{N_\Omega} f_1(\xi_j)f_2(\xi_j)\Delta\Omega_j, 
\]

\[
\|f(\xi)\|^2 = \langle f(\xi), f(\xi) \rangle, 
\]

where \( \Omega \) is the spatial domain, \( N_\Omega \) is the number of cells in the discretization, \( \xi_j \) is the coordinate of the \( j \)th cell’s center, and \( \Delta\Omega_j \) is the \( j \)th cell’s volume. For simplicity of notation the constant \( t_i \) argument for all fields was omitted. Note that when the field is discretized, calculating the inner product and norm requires taking into account the volume of each cell. In practice this is enforced by an appropriate choice of the weight matrix in the L2 norm. The norm of a vector \( x \) with given weight matrix \( W \) is given by

\[
\|x\|^2_W = x^TWx. 
\]

For the case where the vector consists of only a discretized field the weight matrix is a diagonal matrix with \( W_{jj} = \Delta\Omega_j \).

Another place where the concept of a continuous field comes into play is in the specification of a covariance matrix for the prior state distribution. This is required in the \texttt{PhysicsModel.generate_ensemble} method. If the state vector includes discretized fields, the covariance matrix should reflect the physical correlation of these fields. For example, a common covariance kernel for spatial correlation is the squared exponential kernel.
kernel, where the covariance between the values of the field at two spatial locations is given as a function of the distance between the two locations. Using the squared exponential kernel the covariance matrix $C$ for a vector $x$ consisting of a single discretized field is given by

$$C_{j,k} = C(\xi_j, \xi_k) = \sigma^2 \exp \left( -\frac{1}{2} \frac{||\xi_j - \xi_k||^2}{l^2} \right),$$

(5.3)

where the parameters $\sigma^2$ and $l$ are the variance and length scale, respectively, and $C$ is the continuous covariance kernel. More complex examples might use different kernels for the spatial covariance, such as a periodic kernel. Wu et al. [9] proposed a method of incorporating known physical constraints for input fields through choice of covariance matrix. Similarly Michelén Ströfer et al. [8] used the choice of covariance matrix to enforce boundary conditions on the input fields.

Using the Karhunen-Loève (KL) decomposition, a random field can be represented as an infinite linear combination of orthogonal basis functions, referred to as modes, where the coefficients of the linear combinations are random variables. In the case of a Gaussian process the coefficients are identically distributed independent random variables with standard normal distribution. That is, the random field at a given time $t_i$ can be written as

$$f(\xi) - \mathbb{E}(f) = \sum_{j=1}^{\infty} \omega_j \phi_j(\xi) \approx \sum_{j=1}^{N_m} \omega_j \phi_j(\xi) \approx \sum_{j=1}^{N_m} \omega_j \phi_j(\xi),$$

(5.4)

where $\mathbb{E}(f)$ is the mean field, $\omega_j$ are the random coefficients and $\phi_j(\xi)$ are the modes. Here again the constant $t_i$ argument for all fields was omitted and it is understood that the modes and coefficients correspond to the field at that time. The first approximation corresponds to the discretization of the domain and the second corresponds to an approximation using only $N_m < N_\Omega$ modes. The first few modes contain most of the variance and it is typical to represent the field with a subset of $N_m$ modes chosen to cover some percentage (e.g. 99%) of the variance. For a Gaussian process each coefficient has a standard normal distribution $\omega_j \sim \mathcal{N}(0,1)$. The user can use the KL decomposition to create the initial set of samples as an alternative to the Cholesky decomposition. Additionally, the KL decomposition can be used as a reduced order model where the coefficients $\{\omega_j\}_{j=1}^{N_m}$ are the state rather than the discretized values of the field [4].

The KL modes are given by the eigendecomposition of the covariance kernel. For a continuous covariance kernel $C(\xi_1, \xi_2)$ the eigenvalues $\lambda_k$ and eigenfunctions $e_k(\xi)$ are obtained by solving the associated Fredholm integral equation

$$\int_{\Omega} C(\xi_1, \xi_2) e_k(\xi_1) d\xi_1 = \lambda_k e_k(\xi_2),$$

(5.5)

and sorting the eigenpairs based on largest eigenvalues with $\lambda_1$ corresponding to the largest eigenvalue. With unit eigenfunctions $\hat{e}_k(\xi)$, the modes are given by

$$\phi_k(\xi) = \sqrt{\lambda_k} \hat{e}_k(\xi),$$

(5.6)
where the normalized eigenfunctions describe the mode shapes and the eigenvalues are the variance associated with each mode. For the discretized case the Fredholm integral equation in Eq. (5.5) becomes (see Appendix C)

\[(CW)(e_k) = \lambda_k e_k, \quad (5.7)\]

where \(C\) and \(e_k\) are now the covariance matrix and an eigenvector, respectively. The weight matrix \(W\) is diagonal with entries \(W_{ii} = \Delta\Omega_i\), where \(\Delta\Omega_i\) is the volume of the \(i\)th cell in the discretization. The modes are then given by

\[\phi_k = \sqrt{\lambda_k} \hat{e}_k. \quad (5.8)\]

Note that normalizing the eigenfunctions and eigenvectors is done with the L2 norm for a field given in Eq. (5.1) and Eq. (5.2). An alternative representation of the discrete problem is also presented in Appendix C.

5.1 Coupling to third party software: OpenFOAM example

The physics solver for most applications is likely to be either an in-house or commercial third party software. For instance DAFI has mostly been used to solve the RANS equations [4, 7, 8] using the open source finite volume tool OpenFOAM [32]. DAFI includes a tutorial that couples OpenFOAM and DAFI to solve the RANS equations. This serves as a general example on how to couple DAFI with other third party physics solvers, and as a specific example for developing other OpenFOAM-based physics models. The tutorial solves the RANS field inversion problem where the eddy viscosity is inferred from observations of the velocity for a two-dimensional flow over infinite periodic hills (see Fig. 5 later). The physics model nutfoam.py uses OpenFOAM for solving the RANS equations (forward problem). It serves as a more complex example of writing a physics model, using the fields and covariance modules, and coupling to third party solvers. To facilitate developing other OpenFOAM-based solvers, DAFI also includes the module dafi.foam for OpenFOAM file input/output (I/O) operations.

In addition to OpenFOAM DAFI is being coupled to the open-source NHWave [33] software by the developers at Old Dominion University. As an open source project, third party coupling with other software by the community are welcomed and could be included in future releases of the code.

5.2 Example: Working with fields

DAFI handles fields with arbitrary boundary shapes and discretization, i.e. it is not limited to constant-spacing structured meshes. This section shows an example of performing some field operations using DAFI. The case corresponds to the infinite periodic hills [34] example available in the tutorials, where a single hill is modeled using periodic boundary conditions. The top boundary is a wall. The mesh used is shown in Fig. 5(a) and consists
of 3000 cells. An important task is creating a covariance matrix for the input field (state to be inferred). The square exponential kernel is chosen for the vertical direction, and a periodic kernel is used for the periodic direction. The covariance kernel is then modified [8] to enforce zero covariance at the wall boundaries which have fixed-value Dirichlet boundary condition. The covariance between the field value at a single cell the field value at all other cells in the domain is shown in Fig. 5(b). This corresponds to a single row (or column) of the covariance matrix. It can be seen that the point is highly correlated to its immediate neighbors (even across the periodic boundary) and then quickly becomes uncorrelated with the rest of the domain. The covariance also goes to zero towards the wall.

A follow on, optional, task is calculating the discrete modes for the KL decomposition of the covariance matrix. This decomposition can be used to generate samples from the random field or truncated and used as a reduced order model, as shown in Eq. (5.4). The mode shapes (normalized eigenvectors) for a few modes, and the variance (eigenvalues) of each mode are shown in Fig. 6(a) and Fig. 6(b) respectively. The initial modes are seen to capture variance at large length scales, while the latter modes account for smaller length scales. To generate the prior samples, the modes would be weighted by the coefficients $\omega_i$ and added to the mean field following Eq. (5.4). For a Gaussian process, which the prior distribution is typically assumed to be, the coefficients are sampled from independent standard normal distributions (i.i.d.), i.e. $\omega_i \sim \mathcal{N}(0,1)$. For reduced order modeling, it can be seen in Fig. 6(b) that the first 200 modes capture virtually all the variance. Using this reduced set of modes to represent the state reduces the dimensionality of the state space from 3000, corresponding to the number of cells, to 200, corresponding to the number of retained KL modes. The state would now consist of the 200 coefficients $\omega_i$.

6 Test cases

Several test cases are presented in this section to showcase the range of problems DAFI is meant to solve. The first test case solves the classical filtering problem for a chaotic
dynamic system governed by the Lorenz equations using the EnKF. In this case the state consists of three scalar quantities related through three governing equations. This example, while simple, serves as an introduction to data assimilation and the filtering procedure. The second test case is a field inversion problem where the diffusivity (input) field is inferred from observations of the output field of the diffusion equation. This is first solved with the iterative EnKF which leads to improvement in the output field but to an input field with undesirable characteristics. This is possible because of the ill-posed nature of field inversion problems, where typically many input fields can result in similar output fields. The third test case uses the same diffusion field inversion problem but uses the regularized EnKF to enforce smoothness in the input field. This leads to similar improvements in the output field but with better behaved input field. The fourth test case shows the results, using DAFI and reproduced with permission from Michelén Ströfer et al. [8], of field inversion problem using the RANS equations where the eddy viscosity field is inferred from velocity observations. This constitutes an example of a problem of practical importance and of using a more complex, non-trivial geometry. Finally the fifth test case solves a simple scalar inverse problem described by algebraic equations, but focuses on the ability of doing uncertainty quantification with DAFI. Different methods are used and the advantage of either the EnKF-MDA or the EnRML over the EnKF for uncertainty quantification is demonstrated. These test cases collectively give an overview of DAFI’s capabilities and are available in the GitHub repository.

6.1 Data assimilation: Lorenz equations

The Lorenz equations, first introduced by Edward Lorenz in 1963 [35], are a well known system of chaotic ordinary differential equations. The system can be used to model at-
mospheric convection and contains three states governed by the following three coupled PDE

\[
\frac{dx_1}{dt} = \sigma(x_2 - x_1), \quad \frac{dx_2}{dt} = \rho x_1 - x_2 - x_1 x_3, \quad \frac{dx_3}{dt} = x_1 x_2 - \beta x_3,
\]

where \(\sigma, \rho,\) and \(\beta\) are real positive model parameters. These equations constitute the dynamic model, which is used to propagate a given state in time. The state \(x_1\) is proportional to the rate of convection, and the states \(x_2\) and \(x_3\) are proportional to the horizontal and vertical temperature differences, respectively [35]. The Lorenz equations are chaotic and small deviations in initial conditions can quickly result in large differences in the state. The chaotic behavior can be controlled by using data assimilation to correct the states whenever observations are available.

The baseline initial condition is chosen as \(x_1(0) = -8.5, x_2(0) = -7.0, x_3(0) = 27.0\) which is slightly off from the true initial conditions \(x_1(0) = -8.0, x_2(0) = -9.0, x_3(0) = 28.0\). Additionally, the parameters \(\beta\) and \(\sigma\) are taken as correct, but the parameter \(\rho\) is considered uncertain. The state augmentation procedure is used and \(\rho\) is added to the state, which is now

\[
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \rho \end{bmatrix}.
\]

The parameter \(\rho\) is chosen as 29 whereas the true value is taken as 28. As can be seen in Fig. 7 the baseline solution quickly diverges from the truth. We use the true solution to generate synthetic observations, which consists of observations of state \(x_1\) and \(x_3\) every half second. This corresponds to the observation operator

\[
H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.
\]

These observations each have an uncorrelated observation error (variance) based on relative and absolute standard deviations of \(r = 0.1\) and \(a = 0.05\), as

\[
C_{y,i,i} = (y^*_i r + a)^2,
\]

for \(i \in 1,2\), where \(y^*_i\) is the synthetic truth. Finally, the confidence in our initial values needs to be specified to create the prior distribution. The prior is taken as a multivariate Gaussian distribution with mean equal to the baseline values listed above and uncorrelated covariance matrix

\[
C_x = \text{diag}(0.4, 2.0, 1.4, 4.0).
\]
With the prior distribution specified, the dynamic model available, and the observations and their error available, the EnKF is used to correct the state each time an observation is made. The results of EnKF with 100 samples are shown in Fig. 7. The figure shows two states, $x_1$ and $x_2$, at two different time windows, one at the beginning of the process and the other after the process has been going on for a while. Each sample in the ensemble is propagated with the dynamic model. When an observation becomes available each sample is updated using the observation and ensemble statistics. It can be seen that at the beginning the state distribution has a mean far from the truth with very large covariance. After the data assimilation process has gone on for a while the mean gets very close to the truth and the uncertainty is greatly reduced. This is true for all states,
even $x_2$ which is not directly observed. For reference the baseline solution, that is the solution with the initial values with no data assimilation, is also shown. Fig. 8 shows the inferred value of the parameter $\rho$ which in general gets closer to the truth as more data is assimilated. The inferred parameter $\rho$ initially varies rapidly but quickly stabilizes as more data is assimilated.

6.2 Field inversion: Diffusion equation

The one-dimensional diffusion equation with homogeneous boundary conditions is used in this test case. The diffusion equation is given as

$$- \frac{d}{d\xi_1} \left( \mu \frac{du}{d\xi_1} \right) = f(\xi_1),$$  
$$u(0) = u(L) = 0,$$

where $u$ is the quantity being diffused (e.g. heat), $\mu$ is the diffusivity, $\xi_1$ is the spatial coordinate along the domain (e.g. a finite rod), $L$ is the domain length, and $f$ is a source term. The problem consists of inferring the non-constant diffusivity field $\mu$ from sparse observations of the output field $u$. The diffusion equations can be seen as a forward model that propagates the state (input field $\mu$) to the observable field (output field) $u$. The forward model is solved using a central finite difference scheme. The source term considered is $f(\xi_1) = \sin(0.2\pi \xi_1)$. The synthetic truth is created using the first three modes each with coefficient $\omega = 1$. The observations are created from the synthetic truth output field $u$ at $\xi_1/L \in \{0.25, 0.5, 0.75\}$ each with uncorrelated variance based on relative and absolute standard deviations of 0.1 and 0.0001. The observation operator then consists of the forward model followed by selection of the three points.
The domain is discretized into 100 equally spaced cells and the state could consist of the discretized diffusivity field, however it will be modified to enforce positivity and to showcase dimensionality reduction using KL modes. The diffusivity field is physically constraint to be positive, and this is enforced by inferring the logarithm of diffusivity \( \log[\mu/\mu_0] \) rather than the diffusivity directly. The baseline solution \( \mu_0 \) corresponds to our initial guess or prior belief (that is, before seeing the observation data) and was chosen as a constant field. The prior distribution is taken as a Gaussian process with zero mean and specified covariance \( C_x \), that is

\[
\log[\mu/\mu_0] \sim \mathcal{GP}(0, C_x). \tag{6.9}
\]

This formulation makes diffusivity have a lognormal distribution with median \( \mu_0 \), and the state being inferred corresponds to the exponent in the multiplicative term \( \mu = \mu_0 \exp(x) \). For the covariance a square exponential kernel is used with a standard deviation of \( \sigma_p = 5.0 \) and length scale \( 1/L = 0.02 \). While 100 states is manageable, for more complex cases involving fields the size of the discretization can become unmanageable. The KL modes can be used for dimensionality reduction as discussed in Section 5. With a choice of the \( m \) modes with largest variance, the field can be represented as

\[
\log[\mu/\mu_0] = \sum_{i=1}^{m} \omega_i \sqrt{\lambda_i} \phi_i, \tag{6.10}
\]

where the variance \( \lambda_i \) and mode shapes \( \phi_i \) are obtained from the eigendecomposition of the covariance matrix as discussed in Section 5. The state vector now consists of the \( m \) coefficients \( \omega_i \) and we use \( m = 15 \).

Results of the iterative EnKF using 100 iterations and 100 samples are shown in Fig. 9. The inferred output field \( u \) is greatly improved by incorporating the data, but the input field does not approach the true solution. This is because the problem is ill-posed, with many possible input fields resulting in good agreement with the observations. The next subsection solves the same problem using regularization to enforce known or desired properties of the input field.

### 6.3 Regularized field inversion

The field inversion problem in Section 6.2 did not result in an input field close to the truth. This is due to the ill-posedness of the field inversion problem. We will now show how to use the regularized EnKF discussed in Section 3.2 to enforce desired properties of the input field and alleviate the ill-posedness. Specifically, we notice that the inferred diffusivity field is very jagged and we have knowledge (e.g. expert domain-specific knowledge) that it should be more smooth. As such we will prefer to only use the first three modes if possible and will penalize the use of any additional modes, with the penalty function

\[
\mathcal{G}[\omega] = \omega_i, \tag{6.11}
\]
and weighting matrix

\[ W = \lambda \text{diag}(0,0,0,1,\cdots,1), \quad (6.12) \]

where \( \lambda \) is regularization strength parameter. This does not prevent the inference from utilizing higher modes, if using them does improve the agreement with observations, but among equally good solutions it prefers ones that use at most the first three modes.

The results of the field inversion using the regularized EnKF and a regularization coefficient of \( \lambda = 10^6 \) are shown in Fig. 10. The regularization coefficient used a ramp-up tanh function over the first 10 steps to decrease the effect of regularization during the first few iterations. The prior distributions are identical as before and the posterior of the output field displays the same improvement as with the EnKF (Fig. 9(d)), and therefore those results are omitted. The inferred input diffusivity field is still not close to the truth, but it satisfies the desired smoothness. This means that the problem is still ill-posed and
6.4 Field inversion: RANS equations

The Reynolds averaged Navier–Stokes equations describe the mean velocity and pressure of fluid flows. The RANS equations with eddy viscosity approximation of the Reynolds stress are given by

\[
\begin{align*}
\frac{\partial U_i}{\partial x_i} &= 0, \\
U_j \frac{\partial U_i}{\partial x_j} &= -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_i} \left( (\nu + \nu_t) \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \right),
\end{align*}
\]

where \( U_i \) are the velocity components, \( p \) is the pressure, \( x_i \) are the spatial coordinates, \( \nu \) is the fluid’s kinematic viscosity, and \( \nu_t \) is the unclosed eddy viscosity field. This subsection presents the results, reproduced with permission from Michelén Ströfer et al. [8], of inferring the eddy viscosity field from a single point observation of the velocity for the two-dimensional flow over periodic hills. The hill geometry is the same presented earlier in Fig. 5, the Reynolds number, based on hill height and bulk velocity, is 5,600, and the mesh consists of 3,000 cells. The mode decomposition and reduced order modeling described earlier is used and the state to be inferred consists of the coefficients for the first 192 modes. The prior and synthetic truth solutions are obtained from solving the problem with different turbulence models. For full details refer to Michelén Ströfer et al. [8]. The results using the EnKF are shown in Fig. 11. It can be seen that a velocity observation at a single point results in a posterior velocity field that is much closer to the truth than the prior. The inferred eddy viscosity field, while closer in magnitude to the
Figure 11: Results of the RANS field inversion test case, reproduced with permission from Michelén Ströfer et al. [8]. With a single observation of the velocity field the entire velocity field can be improved. While the inferred eddy viscosity field is closer to the truth than the prior it is still not correct due to the ill-posedness of the problem.}

truth, still deviates significantly from the ground truth. This is due to the ill-posedness of the problem and could be addressed with further regularization.

6.5 Uncertainty quantification

One advantage of ensemble-based Bayesian methods is that they result in an estimate of the full posterior distribution. This is in contrast to derivative based Bayesian methods where only the maximum a-posteriori estimate of the posterior is obtained. However, for field inversion problems using the iterative EnKF, repeated applications of the EnKF tends to make samples collapse. This results in the proper mean value but misrepresents the true uncertainty in the estimate. Zhang et al. [36] showed the better capabilities of EnKF-MDA and EnRML over EnKF to capture the uncertainty of the posterior. They compared the results of the three methods to more accurate estimate of the posterior using a Markov chain Monte Carlo (MCMC) approach.

This test case demonstrates the use of DAFI for uncertainty quantification and highlights the advantage of methods like EnKF-MDA and EnRML over EnKF. The test problem is a simple inversion problem consisting of two scalar states $x = [x_1, x_2]^{\top}$, and two observations related to the state by the observation operator

$$\mathcal{H}(x) = \begin{bmatrix} x_1 \\ x_1 + x_2^2 \end{bmatrix}. \quad (6.14)$$

The prior distribution is given as

$$x \sim \mathcal{N}\left([0.5,0.5]^{\top}, \text{diag}(0.1^2,0.1^2)\right). \quad (6.15)$$

There is a single observation with value $y = [0.8,2.0]^{\top}$ and independent variance of 0.05$^2$ for both observed quantities.
Figure 12: Results of the uncertainty quantification test case. The results are shown in both the state space (upper row) and observation space (lower row) for all three methods. It is clear that the EnKF collapses, underestimating the uncertainty in the posterior distribution. The other two methods, and the EnKF-MDA especially, can better capture this uncertainty.

The case was solved with the iterative EnKF, EnKF-MDA, and EnRML using 1000 samples. The iterative EnKF and the EnRML used the discrepancy principle for the stopping criteria with parameter \( \tau = 1.2 \), and the EnRML used a length scale parameter of \( \gamma = 0.5 \). The iterative EnKF and EnRML converged after 7 and 6 iterations, respectively. The EnKF-MDA was done with an inflation parameter of \( N_{mda} = 10 \). It was verified that larger values of the inflation parameter resulted in similar posterior distributions. The results are shown in Fig. 12 where the kernel density estimate (KDE) for samples is plotted using the Scott method for the bandwidths. All methods are successful in inferring mean closer to the observations. The EnKF results do indeed collapse, which results in a poor estimate of the uncertainty. Both the EnKF-MDA and EnRML are able to better capture the uncertainty. Zhang et al. [36] demonstrate that while the shape of the posterior for the EnKF-MDA and EnRML might not be accurate, the estimated level of uncertainty is comparable to that obtained from MCMC. The results for the EnKF-MDA in state space
are shown again in Fig. 13 both as the raw samples and the fitted KDE. This is done to elucidate the KDE and show more details on the marginal distributions.

7 Software installation and numerical examples

The software can be obtained from the GitHub repository [31] or installed from the PyPI repository [29] using the Python package manager pip. The code can run on any system with Python and NumPy and no compilation is required. The computational requirements and running times depend greatly on the problem being solved which is user-specific. The solution technique in this code requires running an ensemble of models and it is desirable to parallelize these evaluations. The parallelization task falls under the user-defined physics model in either the forecast to time or state to observation functions, depending on the problem. Through these functions the DAFI code provides the user’s physics model an ensemble of states and expects back an updated ensemble. The RANS field inversion tutorial for the periodic hill (Section 6.4) is also an example of how to parallelize this task using the standard Python library. This example is able to run in HPC machines but as currently implemented it is limited to multiple cores in a single node. More complex parallelization, such as parallelizing each of the individual model evaluations, are possible but fall under the user’s physics model.

Several numerical examples were presented in Section 6. The complete setup including input files are included in the tutorials directory of the software distribution. Detailed explanation of the input file are described in the documentation [30]. In addition
to the input file, the user needs to provide a physics model with predefined methods as detailed in Section 4.1. These files are included in the tutorials for all test cases presented and Appendix D provides some code example for such a file.

8 Concluding remarks

DAFI is an open-source, object-oriented Python package aimed at providing a robust framework for solving data assimilation and field inversion problems. It contains a library of ensemble-based, derivative-free, Bayesian methods based on the ensemble Kalman filter, and a straightforward way of coupling the user’s domain-specific model. As such the user is required to have domain-specific knowledge but little knowledge is required on the data assimilation and field inversion techniques. As an example, the authors have used DAFI to infer the Reynolds stress tensor in Reynolds-averaged Navier–Stokes (RANS) simulations of fluid flow [4]. Alternatively DAFI can be used as a testbed for research on new inversion methods, which is straightforward due to the object-oriented nature of the code. As examples, the authors have used DAFI to develop the regularized ensemble Kalman filter [7] and to enforce boundary conditions during field inversion [8]. One distinguishing feature of DAFI is that it was created with physical fields in mind. As such it includes several modules for field operations such as creating covariance matrices using different kernels, or performing the Karhunen-Loève decomposition. All this can be done in unstructured discretization of the fields.

This paper presents the theory behind ensemble-based methods and random fields, the approach taken to generalize diverse problems into a general problem, and the code implementation. It also presents several test cases that show the use of DAFI as well as the diversity in the types of problems it is meant to solve. The test cases include a scalar data assimilation problem using the Lorenz equations, a regularized field inversion problem using the diffusion equation, and an uncertainty quantification problem. The results from a different study, using DAFI, for a practical field inversion problem using the RANS equations are also shown, with permission. The code is actively maintained in GitHub [31], released through PyPI [29], and the documentation is available online through Read the Docs [30]. The GitHub repository and the documentation include several tutorials. The authors hope the code is beneficial to other researchers and in keeping with the open source nature of the project accept code improvements through GitHub pull requests.

Acknowledgements

Part of DAFI was adopted from previous code developed at Virginia Tech by Dr. Jianxun Wang, Dr. Jinlong Wu, and Dr. Rui Sun, whose contributions are gratefully acknowledged. The authors would like to thank Dr. Gangfeng Ma at Old Dominion University for his valuable feedback as one of the earliest external adopters of the DAFI code.
The authors would also like to thank Kristopher Olsheski for his help reviewing this manuscript.

A Kalman gain matrix for EnKF

This appendix shows how the Kalman gain matrix for the EnKF can be written in terms of the state mapped to observation space $z = Hx$ by using the definition of sample covariance. This avoids using, or having to explicitly construct, the observation operator $H$. It will also avoid having to construct, or perform matrix multiplications with, the full state covariance matrix $C_x$. The state $x$ mapped to observation space is

$$ z = Hx. \quad \text{(A.1)} $$

However the vector $z$ can typically be obtained directly from the physics model without explicitly constructing the matrix $H$ or carrying out the matrix multiplication. The Kalman gain matrix for the EnKF is given by

$$ K = C_x H^T \left( HC_x H^T + C_y \right)^{-1}, \quad \text{(A.2)} $$

where the sample covariance for the state $x$ is given by

$$ C_x = \frac{1}{N_s - 1} \sum_{j=1}^{N_s} (x^{(j)} - \bar{x})(x^{(j)} - \bar{x})^T, \quad \text{(A.3)} $$

and $N_s$ is the number of samples in the ensemble, $x^{(j)}$ is a particular sample, and $\bar{x}$ is the sample mean. Substituting Eq. (A.3) into Eq. (A.2) and rearranging, the Kalman gain matrix can be written as

$$ K = C_xz (C_z + C_y)^{-1}, \quad \text{(A.4)} $$

where $C_y$ is the observation error (covariance) matrix, $C_z$ is defined similarly to $C_x$, and the covariance matrix between the state and the state mapped to observation space is given by

$$ C_xz = \frac{1}{N_s - 1} \sum_{j=1}^{N_s} (x^{(j)} - \bar{x})(z^{(j)} - \bar{z})^T. \quad \text{(A.5)} $$

Note that the Kalman gain in Eq. (A.4) does not include the very large state covariance matrix $C_x$, rather it involves much smaller covariance matrices since the dimension of the observation space is generally much smaller than that of the state space.

B Iterative EnKF for field inversion

In field inversion problems one field is related to another through a nonlinear forward operator, and the input field is inferred from observations of the output field. This appendix
shows how a field inversion problem can be recast as an artificial dynamics problem using an augmented state and the EnKF used to solve it [3]. The augmented state vector written in terms of block matrices is

\[ \tilde{x} = \begin{bmatrix} x \\ \mathcal{H}(x) \end{bmatrix} = \begin{bmatrix} x \\ z \end{bmatrix}, \]  
(B.1)

and linear observation operator is

\[ \tilde{\mathcal{H}} = \begin{bmatrix} 0 & I \end{bmatrix}. \]  
(B.2)

Here the state \( x \) mapped to observation space is \( z \) and is given by a nonlinear observation operator \( \mathcal{H} \) that combines the forward model and the observation operator. The artificial dynamics is given by

\[ \tilde{x}_{i+1} = \tilde{\mathcal{M}}(\tilde{x}_i) = \begin{bmatrix} x_i \\ \mathcal{H}_i(x_i) \end{bmatrix}. \]  
(B.3)

The analysis state is then given by

\[ \tilde{x}^a(j) = \tilde{x}^f(j) + \tilde{K} \left( y^{(j)} - \tilde{\mathcal{H}}x^{f(j)} \right), \]  
(B.4)

with Kalman gain matrix

\[ \tilde{K} = C_x \tilde{\mathcal{H}}^T \left( \tilde{\mathcal{H}} C_x \tilde{\mathcal{H}}^T + C_y \right)^{-1}, \]  
(B.5)

where the covariance matrix is composed of four blocks as

\[ C_x = \begin{bmatrix} C_x & C_{xz} \\ C_{xz} & C_z \end{bmatrix}. \]  
(B.6)

The analysis step can be expressed in terms of the original state \( x \). Substituting Eq. (B.6) into Eq. (B.5) and carrying out the block operations the Kalman gain matrix becomes

\[ \tilde{K} = \begin{bmatrix} C_{xz} (C_z + C_y)^{-1} \\ C_z (C_z + C_y)^{-1} \end{bmatrix}, \]  
(B.7)

and Eq. (B.4) becomes

\[ \begin{bmatrix} x^{a(j)} \\ z^{a(j)} \end{bmatrix} = \begin{bmatrix} x^{f(j)} \\ z^{f(j)} \end{bmatrix} + \begin{bmatrix} C_{xz} (C_z + C_y)^{-1} \\ C_z (C_z + C_y)^{-1} \end{bmatrix} \left[ y^{(j)} - \mathcal{H}(x^{f(j)}) \right], \]  
(B.8)

or

\[ x^{a(j)} = x^{f(j)} + C_{xz} (C_z + C_y)^{-1} \left( y^{(j)} - \mathcal{H}(x^{f(j)}) \right), \]  
(B.9a)

\[ z^{a(j)} = z^{f(j)} + C_z (C_z + C_y)^{-1} \left( y^{(j)} - \mathcal{H}(x^{f(j)}) \right). \]  
(B.9b)

Eq. (B.9a) is the EnKF update scheme but with nonlinear observation operator:

\[ x^{a(j)} = x^{f(j)} + K \left( y^{(j)} - \mathcal{H}(x^{f(j)}) \right). \]  
(B.10)
That is, the artificial dynamics can be solved using EnKF on the original state vector but using the nonlinear observation operator and iterating by using the updated analysis state as the forecast state until convergence is achieved. Note that Eq. (B.9b) is not needed and solving the full augmented system would include unnecessary computations.

C Discrete KL modes

When the state vector includes physical fields these need to be discretized, using some mesh which is generally unstructured. This appendix describes how to obtain the KL modes in the discrete, unstructured mesh, case. The continuous Fredholm integral equation
\[ \int_{\Omega} C(\xi_1, \xi_2) e_k(\xi_1) d\xi_1 = \lambda_k e_k(\xi_2) \]  
(C.1)
can be discretized as
\[ \sum_{i=1}^{N} C_{ij} e_k \Delta \Omega_i = \lambda_k e_k, \]  
(C.2)
where \( N \) is the number of cells in the discretization, \( \Delta \Omega_i \) is the volume of the \( i \)th cell and \( C \) and \( e_k \) are now discrete covariance matrix and eigenvector. These \( N \) equations (for \( j \in [1,N] \)) can be written as a matrix equation as
\[ (CW) e_k = \lambda_k e_k, \]  
(C.3)
with
\[ W = \begin{bmatrix} \Delta \Omega_1 & \cdot & \cdot & \cdot & \Delta \Omega_N \end{bmatrix}, \]  
(C.4)
which is an eigenvalue problem. The discrete KL modes are then given by
\[ \phi_k = \sqrt{\lambda_k} \hat{e}_k, \]  
(C.5)
where \( \hat{e}_k \) are the normalized eigenvectors (using the norm in Eq. (5.2)).

Optionally, the basis can be changed such that the standard vector dot product in the new basis is equivalent to the weighted L2 norm of Eq. (5.2) in the original basis. The desired change of basis matrix \( L \), which transforms a vector from the original coordinates to the desired coordinates, is such that
\[ x^T W x = (Lx)^T (Lx). \]  
(C.6)
This can be written as
\[ x^T W x = x^T L^T L x, \]  
(C.7)
which leads to

\[ W = L^\top L, \quad \text{(C.8)} \]
\[ L = L^\top = \begin{bmatrix} \sqrt{\Delta \Omega_1} & \cdots & \sqrt{\Delta \Omega_N} \end{bmatrix}, \quad \text{(C.9)} \]
\[ W = LL. \quad \text{(C.10)} \]

Changing the basis of both sides of Eq. (C.3) results in

\[ L(CW)e_k = L\lambda_ke_k, \quad \text{(C.11)} \]
\[ LCLLe_k = \lambda_kLe_k, \quad \text{(C.12)} \]
\[ (LCL)(Le_k) = \lambda_k(Le_k). \quad \text{(C.13)} \]

This is still the same eigenvalue problem but in different coordinates. The eigendecomposition of $LCL$ gives the eigenvalues and the eigenvectors $g_k = Le_k$ which can be normalized using the standard dot product to $\hat{g}_k$. The normalized (weighted norm definition) eigenvectors are then obtained by reverting to the original basis as

\[ \hat{e}_k = (L)^{-1}\hat{g}_k. \quad \text{(C.14)} \]

**D  Code example**

This appendix presents a simple example of a physics model. The physics model is that used for the scalar inversion problem in Section 6.5. In particular, note the implementation of the required method described in Section 4.1. The forecast_to_time method is not implemented since this field inversion problem does not requires it. The model is first initialized via the _init_ method using the following input dictionaries.

```python
inputs_dafi = {
    'model_file': 'model.py',
    'inverse_method': 'EnKF',
    'nsamples': 1000,
    'max_iterations': 100,
    'convergence_option': 'discrepancy',
    'convergence_factor': 1.2,
}

inputs_model = {
    'x_init_mean': [0.5, 0.5],
    'x_init_std': [0.1, 0.1],
    'obs': [0.8, 2.0],
    'obs_std': [0.05, 0.05],
}
```
The DAFI input defines the number of samples, physics model and inverse method to use, and convergence criteria. The model input defines the prior distribution and the observation data. The initial ensemble is generated in the generate_ensemble method using the specified prior mean and standard deviation as in Eq. (6.15). The state_to_observation method maps the ensemble from state space to observation space based on the observation operator in Eq. (6.14). Finally, the get_obs method simply returns the value of the observation and the observation error. The complete physics model is shown below. The physics model and input files for all test cases presented here are included in the tutorials distributed with the code.

```plaintext
# Copyright 2020 Virginia Polytechnic Institute and State University.
""" Dynamic model for solving the scalar inversion problem used for uncertainty quantification. """

# standard library imports
import os

# third party imports
import numpy as np
import yaml

# local imports
from dafi import PhysicsModel

class Model(PhysicsModel):
    def __init__(self, inputs_dafi, inputs_model):
        # save the required inputs
        self.nsamples = inputs_dafi['nsamples']

        # read inputs
        self.init_state = np.array(inputs_model['x_init_mean'])
        self.state_std = np.array(inputs_model['x_init_std'])
        self.obs = np.array(inputs_model['obs'])
        self.obs_err = np.diag(np.array(inputs_model['obs_std'])**2)

        # required attributes.
        self.name = 'Scalar Inversion Case for UQ'

        # other attributes
        self.nstate = len(self.init_state)

    def __str__(self):
        return self.name

    def generate_ensemble(self):
        state_vec = np.empty([self.nstate, self.nsamples])
        for i in range(self.nstate):
            state_vec[i, :] = np.random.normal(
                self.init_state[i], self.state_std[i], self.nsamples)
```

93
def state_to_observation(self, state_vec):
    obs_1 = state_vec[0, :]
    obs_2 = state_vec[0, :] + state_vec[1, :] ** 3
    model_obs = np.array([obs_1, obs_2])
    return model_obs

def get_obs(self, time):
    return self.obs, self.obs_err

References

Chapter 5

Enforcing Boundary Conditions on Physical Fields in Bayesian Inversion
Enforcing boundary conditions on physical fields in Bayesian inversion

Carlos A. Michelén Ströfer\textsuperscript{a}, Xin-Lei Zhang\textsuperscript{a,b}, Heng Xiao\textsuperscript{a,*}, Olivier Coutier-Delgosha\textsuperscript{a,b}

\textsuperscript{a} Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060, USA
\textsuperscript{b} Arts et Metiers ParisTech, 8 Boulevard Louis XIV, 59046 Lille cedex, France

Received 14 November 2019; received in revised form 24 March 2020; accepted 23 April 2020
Available online 18 May 2020

Abstract

Inverse problems in computational mechanics consist of inferring physical fields that are latent in the model describing some observable fields. For instance, an inverse problem of interest is inferring the Reynolds stress field in the Navier–Stokes equations describing mean fluid velocity and pressure. The physical nature of the latent fields means they have their own set of physical constraints, including boundary conditions. The inherent ill-posedness of inverse problems, however, means that there exist many possible latent fields that do not satisfy their physical constraints while still resulting in satisfactory agreement in the observation space. These physical constraints must therefore be enforced through the problem formulation. So far there has been no general approach to enforce boundary conditions on latent fields in inverse problems in computational mechanics, with these constraints often simply ignored. In this work we demonstrate how to enforce boundary conditions in Bayesian inversion problems by choice of the statistical model for the latent fields. Specifically, this is done by modifying the covariance kernel to guarantee that all realizations satisfy known values or derivatives at the boundary. As a test case the problem of inferring the eddy viscosity in the Reynolds-averaged Navier–Stokes equations is considered. The results show that enforcing these constraints results in similar improvements in the output fields but with latent fields that behave as expected at the boundaries.

© 2020 Elsevier B.V. All rights reserved.

Keywords: Bayesian inference; Ensemble Kalman method; Field inversion; Boundary conditions; Inverse problems

1. Introduction

Many problems in computational mechanics consist of known physics described by partial differential equations (PDEs) for some fields of interest (e.g. fluid velocity, solid displacement) but include other physical fields whose truth values are not known exactly. For instance these uncertain fields can be (1) initial conditions of the fields of interest, (2) constant physical properties such as mass (density field) or stress distribution, or (3) physical fields related to the fields of interest through unknown relationships, i.e. the problem is unclosed. Here we refer to such fields as latent physical fields, as they are embedded in the PDEs but not solved for directly. In inverse problems

\* Corresponding author.
E-mail address: hengxiao@vt.edu (H. Xiao).
URL: https://www.aoe.vt.edu/people/faculty/xiaoheng.html (H. Xiao).

https://doi.org/10.1016/j.cma.2020.113097
0045-7825/© 2020 Elsevier B.V. All rights reserved.
these are treated as model parameters that need to be inferred based on observations of the field of interest. For fields of interest \( u \) and latent fields \( \tau \) such problems can be written as

\[
\mathcal{M}(u; \tau) = 0, \tag{1}
\]

where \( \mathcal{M} \) represents the governing partial differential equations (PDEs) including geometry and boundary conditions on \( u \). The fields of interest can be considered a function of the latent fields as

\[
u = \mathcal{F}(\tau). \tag{2}\]

An example of such a problem is the Reynolds-averaged Navier–Stokes (RANS) equations, which describe the mean velocity and pressure fields of a fluid but include the unclosed Reynolds stress field. The unclosed Reynolds stress field is approximated through imprecise turbulence models. Similarly, in many solid mechanics problems the field of interest is displacement and the latent field is the stress in the material, which can be a highly non-linear function of the strain and strain history. In the rest of this work the RANS equations are used as the working example. We are interested in the inverse problem: inferring the true latent fields using an initial estimate of the latent fields and sparse experimental observations of the output fields.

1.1. Inverse problems in computational mechanics

There are many frameworks to improve the predicted fields by using sparse observations [e.g. 1, 2] [3, 4, for comprehensive reviews for RANS simulations], and the choice is between different representations of the uncertainty in the problem. If the governing PDEs are considered exact and therefore the only uncertainty is in the latent field, then the embedded uncertainty formulation by Oliver et al. [2] is most appropriate. In this formulation the problem is written as \( u = \mathcal{F}(\tau) \) as in Eq. (2) and the latent fields are considered uncertain and need to be inferred. This is in contrast to black box approaches where a correction to the model output (e.g. \( \mathcal{F}(\tau) \) in Eq. (2) for some approximation of \( \tau \)) is inferred. The embedded discrepancy formulation is used in this work since the Reynolds stress field is considered the major source of uncertainty in the RANS equations.

In Bayesian inversion approaches the uncertain field is considered a random field, and a statistical model describing this field must be specified. This is referred to as the prior distribution, which is later updated based on the observation data to obtain a posterior distribution. The chosen statistical model, particularly its covariance kernel, can have a large impact in the Bayesian inference. For simplicity in writing the equations, in the rest of the paper it will be assumed that there is a single scalar latent field \( \tau \) which needs to be modeled as a random field. The initial approximation (baseline solution) of the latent field is usually close to the truth, and, in the absence of additional information, the latent field is taken to be a Gaussian process with mean equal to this baseline solution \( \tilde{\tau} \) [e.g. 5–8]. The prior distribution is then

\[
\tau \sim GP(\tilde{\tau}, K), \tag{3}
\]

where \( K \) is the specified covariance. Using the embedded discrepancy formulation described by Eqs. (2) and (3), observations of the output fields can be used to infer a posterior distribution for the latent field. The inferred latent field can then be propagated through the model in Eq. (2) to obtain the inferred output fields of interest. There are many approaches to solve the Bayesian inversion problem directly or approximately, including ensemble based methods [e.g. 5, 9] and optimization/adjoint based methods [e.g. 6, 7].

The Bayesian embedded discrepancy framework has been used in several works to address the RANS equations closure problem. Edeling et al. [10, 11] and Ray et al. [12–14] inferred model coefficients in turbulence models by using Markov Chain Monte Carlo methods. Dow and Wang [6] inferred the eddy viscosity by observing the velocity field, which was among the first efforts to infer a full field rather than model coefficients. Their work used the linear eddy viscosity approximation in the RANS equations, which allows the Reynolds stress tensor field to be represented by a single scalar eddy viscosity field. They obtained the maximum likelihood estimate using an adjoint method to compute the derivatives. Similarly, Singh and Duraisamy [7] also inferred the eddy viscosity by observing velocities but do so by inferring an embedded multiplicative discrepancy into the production term of the Spalart–Allmaras model for eddy viscosity. They obtained the maximum a posteriori (MAP) estimate using an adjoint method to compute the derivatives. Xiao et al. [5] inferred the components of the Reynolds stress tensor field by observing velocities at sparse locations. They used an iterative ensemble Kalman method to infer the Reynolds stress field.
Meldi et al. [15,16] integrated the Kalman method into CFD solvers in an intrusive manner for data assimilation of turbulent flows. Their approach has the advantage of involving only a single simulation rather than an ensemble of simulations as in the ensemble Kalman methods. More recently, Edeling et al. [17] proposed transport equations to describe the deviation from the eddy viscosity model and use Bayesian inference to calibrate the constants in the transport equations. In related efforts, researchers have also used data-driven methods to address closure problems in large eddy simulations [18] and multiphase flows [19,20].

1.2. Ill-posedness and enforcing physical constraints

Inverse problems are inherently ill-posed in that numerous possible latent fields can result in output fields that are close to the observation values. One example is estimating the Reynolds stress from in-plane velocity measurements in a square duct case [5]. In this flow the in-plane velocities are driven by the imbalance $\tau_{22} - \tau_{33}$ in the Reynolds stress, and any Reynolds stress field with the correct imbalance is a possible solution. Ill-posed problems are usually regularized by enforcing additional constraints, such as smoothness of the inferred field. For example, smoothness can be enforced by adding the magnitude of the gradient $\|\nabla \tau\|$ into the cost function.

Because of the ill-posed nature of the problem, most previous research fail at inferring the true latent field while still obtaining global improvements in the fields of interest. Since the latent field is a physical field with known physical constraints, the search space for possible latent fields can be reduced by enforcing these constraints before any additional regularization. One type of physical constraints are global physical constraints that the solution must conform to in the entire domain. These are constraints such as the divergence-free requirement for an incompressible velocity field or the positivity requirement for a pollutant concentration field. Some of these global constraints can be enforced by using a different choice for the statistical model of the random fields in Eq. (3), i.e. using a more complex representation than a simple Gaussian process. The representation of the random latent field is chosen such that the physical constraints are automatically met by any realization of the random field. For example, a positivity constraint can be enforced by modeling the latent field as a lognormal process as

$$\tau = e^\delta$$

(4a)

$$\delta \sim GP(\log(\tilde{\tau}), K),$$

(4b)

where the baseline solution ($\tilde{\tau}$) is now the median of the lognormal distribution, and there is an implicit normalization of $\tau$, $\tilde{\tau}$, and $K$. Xiao et al. [5] use this statistical model to enforce positivity on the turbulent kinetic energy while Dow and Wang [6] use it to enforce positivity on the eddy viscosity.

A second type of physical constraints is boundary conditions (B.C.) on the latent field. Boundary conditions are physical constraints on fields at the boundaries of the domain. Typical examples include fixed values (Dirichlet B.C.), fixed gradient (Neumann B.C.), fixed value and fixed gradient (mixed B.C.), and periodic conditions. For example, in fluid mechanics applications, (i) solid walls enforce a zero-velocity condition referred to as the no-slip condition, (ii) symmetry planes (used as a computational simplification for symmetric problems) enforce a zero-gradient condition on all fields, and (iii) far field conditions enforce zero gradient and fixed value conditions on all fields. Many works have simply ignored some of these boundary conditions and still obtained improvements in the output fields thanks to the ill-posedness of the problem. For instance Xiao et al. [5] do not enforce the periodic boundary condition on the Reynolds stress field. Similarly Dow and Wang [6] do not enforce the symmetry condition on the latent eddy viscosity field. On the other hand, simpler boundary conditions have been often enforced in a problem-specific manner. Particularly, the two works described above [5,6] both enforce the zero-value boundary condition at the wall on the latent fields thanks to the multiplicative nature of lognormal process used as the statistical model. Similarly, Cheung et al. [8] enforce the no-slip (zero-value Dirichlet) condition on the inferred velocity by inferring a multiplicative discrepancy on a baseline velocity that already satisfies the no-slip condition. However, there clearly has been no uniform approach to enforce general boundary conditions on physical latent fields in Bayesian inversion problems. In this work we demonstrate how to enforce arbitrary boundary conditions on the latent fields by choice of statistical model.
1.3. Contribution of current work

In this paper we demonstrate how to enforce general boundary conditions on physical latent fields by choice of the statistical model used to represent them in the Bayesian inversion. We start with the assumption that the latent field is represented by a Gaussian process or a function thereof (e.g. a lognormal process), and the boundary constraints are then enforced by choice of covariance matrix. An appropriate covariance matrix is obtained by first defining an initial Gaussian process and then modifying the covariance matrix in a way that enforces the boundary conditions. We also present how to enforce internal observations of the latent field, which is done in a mathematically similar way by modifying not only the covariance but also the mean of the initial Gaussian process. This can be a realistic scenario since some observations of the latent field can be obtained in practice. For example, Reynolds stress can be obtained from instantaneous measurements of the velocity components. Enforcing both of these types of constraints further reduces the search space in the optimization for the latent field by ensuring it satisfies more of the known physics, driving the inference closer to the goal of obtaining the true latent field. The method is tested for fluid mechanics problems using the RANS equations, but we emphasize that the method is general with applications to many PDE-described physics problems.

1.4. Related work

In what follows we differentiate our contribution from other works using Gaussian processes to represent physical fields. Gaussian processes are commonly used as the Bayesian framework for field inference from observations [21,22] [e.g.23,24, for-contemporary-applications]. In these cases the field being inferred is the same field being observed. The class of problems considered here are inverse problems where the inferred field is related to the observed field through some model (set of PDEs, e.g. RANS equations). We focus on the statistical model used to represent the random field, and not on the particular choice of Bayesian framework used in the inversion. Specifically we seek a choice of statistical model that enforces the boundary conditions for all realizations of the random field. However, the methodology showcased here is directly adopted from these works that use the Gaussian process as the Bayesian framework. The theory of incorporating observed values and derivatives of the random field into the Gaussian process is well developed [25,26], but, to the author’s knowledge, it has not been used in the context of enforcing boundary conditions on the inferred fields in inverse modeling. In many such problems the boundary conditions have simply been ignored as illustrated earlier.

Similarly, there are many works using Gaussian processes to represent solutions to stochastic PDEs [e.g.27–29]. In these works the form of the PDE is embedded into the covariance kernel and realizations of this random fields satisfy the PDE. These are powerful techniques for solving PDEs in a stochastic sense and can be naturally used to solve the inverse problem for model parameters in the PDE [e.g.30,31]). However this is not directly related to the class of problems considered here. Here we consider PDEs with an uncertain latent field which needs to be inferred, but the PDEs are solved deterministically for each proposed latent field during the chosen Bayesian optimization process. As mentioned earlier our focus is seeking a statistical model for the latent field that incorporates known physical constraints. Recently Wu et al. [32] presented a method that incorporates some aspects of Gaussian processes for stochastic PDEs into the problem of specifying a statistical model that incorporates known physical constraints. Specifically, for problems where the latent field is itself governed by PDEs with unclosed fields the structure of these PDEs are incorporated into the covariance kernel of the latent field. Note that this new set of PDEs could be solved directly in conjunction with the main set of PDEs and the latent fields would be the uncertain terms in these additional PDEs. In this sense the method proposed in [32] represents a hybrid approach with the benefit of reducing the number of coupled PDEs that need to be solved for each forward evaluation and reducing the number of latent fields that need to be inferred.

While the current work applies to any Bayesian inversion technique, an ensemble-based algorithm is used in the test cases. There are numerous on-going efforts in improving existing ensemble-based Bayesian inversion algorithms to allow for incorporating physical constraints [33–35]. The current work and that of Wu et al. [32] complement these efforts. The common overarching goal of these works is inferring latent physical fields with limited observation data.

The rest of the paper is organized as follows. Section 2 presents the method of embedding known boundary behavior into the prior statistical model. Section 3 presents the results from a series of test cases, where it is shown that all prior and posterior realizations of the random field satisfy the boundary conditions. Finally, Section 4 concludes the paper and offers some discussions.
2. Methodology

The goal is to ensure that the realizations of the random latent field satisfy any known values and derivatives, be that at internal locations or boundary conditions, by appropriate choice of statistical model for the random field. Internal observations come from experimental measurements and are fundamentally different from boundary conditions which come from physical constraints and modeling choices. However, they can be enforced in a mathematically similar way and thus both are considered here. We consider the cases where the latent field is treated as a Gaussian process (Eq. (3)) or a function of a Gaussian process. Incorporating known values and derivatives into a Gaussian process consists of modifying the mean and covariance of the Gaussian process [25,26]. That is, after observations Eq. (3) becomes

$$\tau \sim GP(\tilde{\tau}^*, K^*)$$,  \hspace{1cm} (5)

where the modified mean $\tilde{\tau}^*$ conforms to all known values and derivatives, and the modified covariance $K^*$ ensures all realizations of the random field do too, within observation errors. This theory is summarized in Section 2.1 and is used to enforce internal observations. In Section 2.2 the application to enforcing boundary conditions is discussed. Additionally, an interactive Jupyter notebook showing an example of enforcing boundary conditions and internal observations in a Gaussian process is available in a GitHub Gist [36].

In the case where the latent field is modeled as a function $\tilde{G}$ of a Gaussian process $\delta$ as

$$\tau = \tilde{G}(\delta)$$ \hspace{1cm} (6a)

$$\delta \sim GP(\mu, K),$$ \hspace{1cm} (6b)

the desired constraints on the latent field $\tau$ first need to be mapped to constraints on the Gaussian process $\delta$. These modified constraints are then enforced in the same manner as for the general Gaussian process. In Section 2.3 we describe this procedure for the general case (Eq. (6)) and show the results for the lognormal process (Eq. (4)) in particular. The lognormal process not only serves as a specific example but is a particularly useful and prevalent formulation in fluid mechanics problems, including the problem we use in the results section to showcase this methodology.

To verify the approach, a Karhunen–Loève (KL) representation of the latent field is used and it is shown that the KL modes satisfy the boundary conditions. A stochastic process can be represented as an infinite linear combination of orthogonal functions through the Karhunen–Loève theorem. The latent field in Eq. (5) can then be written as

$$\tau = \tilde{\tau}^* + \sum_{i=1}^{\infty} \omega_i \sqrt{\lambda_i} \phi_i(x) \approx \tilde{\tau}^* + \sum_{i=1}^{M} \omega_i \sqrt{\lambda_i} \phi_i(x)$$ \hspace{1cm} (7a)

$$\omega_i \sim \mathcal{N}(0, 1),$$ \hspace{1cm} (7b)

where $\lambda_i$ and $\phi_i$ are the eigenvalues and unit eigenfunctions (modes) from the KL decomposition and $\mathcal{N}(0, 1)$ is the standard normal distribution. Realizations of the random process can be obtained by sampling the coefficients $\omega_i$, which have independent standard normal distributions. When the field is discretized, the limit of the summation becomes the number of cells. However, the modes corresponding to the largest eigenvalues account for most of the covariance and it is only necessary to retain a small subset of $M$ modes corresponding to the $M$ largest eigenvalues. The number of retained modes $M$ can be chosen based on the desired coverage (e.g. 99%) of the variance of the field. From Eq. (7) it can be seen that for the realizations to exactly satisfy known values at some locations the KL modes must be zero at those locations. To allow for observation errors, the KL modes must have small magnitudes at these locations. Similarly, the derivatives of the KL modes must be near zero at locations of known derivatives. Therefore, it is sufficient to simply verify that the KL modes satisfy these conditions to guarantee all realizations of the latent field will satisfy the desired conditions. Note that in the case of boundary conditions the values or derivatives of KL modes at these location must be exactly zero, since there are no observation errors.

In addition to showing that the modified mean and covariance guarantee any realization of the latent field satisfies the boundary conditions and internal observations, we also solve the inverse problem. The inverse problem consists of inferring the latent field from sparse observations of the output fields. There are many frameworks for solving the Bayesian inversion problem, from minimization of an objective function using adjoint methods to ensemble-based methods. Here we adopt the framework from Xiao et al. [5], which is based on the iterative ensemble Kalman method developed by Iglésias et al. [9]. The details of the Bayesian inversion are presented in the Appendix. The
mean of the posterior distribution is used as an estimate for the inferred latent field, and correspond to the maximum a posteriori estimate.

For the Bayesian inversion we used a reduced order model consisting of the KL representation in Eq. (7) with a truncated set of $M$ modes to capture 99% of the variance. The state vector to be inferred now consists of the truncated coefficients $\omega = \{\omega_i\}_{t=1}^{M}$, rather than the discretized values of the latent field directly. It is clear that with this representation the inferred latent field will satisfy the boundary conditions since the reduced order model is also a weighted sum of KL modes and the mean (Eq. (7)). This representation is, however, not necessary since the ensemble Kalman filter ensures that the posterior samples are linear combinations of the prior samples [9]. Note that if the observation error is not zero, the inferred state vector $\omega$ could result in a latent field that does not satisfies the known values or derivatives. This could be remedied with regularization to penalize the inferred state vector based on the likelihood that it came from the prior distribution, i.e. where each coefficient $\omega_i$ has a standard normal distribution. For simplicity, in the test cases presented, we solve the inverse problem enforcing boundary conditions alone, without any internal observations of the latent field.

### 2.1. General constraints on Gaussian processes

If the values of a Gaussian process or its derivatives are known at some points, the Gaussian process can be modified to reflect this prior knowledge [25,26]. To incorporate the known values and derivatives, first an augmented vector is defined as $[\tau, t]^T$ where $\tau$ is the sub-vector of the values of the Gaussian process (Eq. (3)) at the inference locations and $t$ is the sub-vector of known values and derivatives of the process at some points. From this point on all fields are discretized unless otherwise noted but the same symbols are used for simplicity. The locations for $\tau$ consist of the cell centers of the discretized domain, and the locations for $t$ are not necessarily co-located with points in $\tau$. Known derivatives in $t$ are defined with respect to one of three orthogonal coordinate directions $x_1$, $x_2$, or $x_3$. To account for derivatives in general directions, the derivative in the three coordinate directions are included in $t$ separately for the same physical location. A Gaussian process is then assumed for the vector $[\tau, t]^T$ as

$$
\begin{bmatrix}
\tau \\
t
\end{bmatrix} \sim GP
\begin{bmatrix}
\tilde{\tau} \\
K_{\tau\tau} & K_{\tau t} \\
K_{t\tau} & K_{tt}
\end{bmatrix},
$$

where $\tilde{\tau}$ are the values of the baseline solution or its derivatives at the observation points, possibly obtained by interpolation, and $K_{\tau\tau}$, $K_{\tau t}$, $K_{t\tau}$, and $K_{tt}$ are sub-matrices of the covariance matrix. Finally, the distribution for inference values conditioned on known values and derivative is then also a Gaussian process given by [25,26]

$$
\tau(t = y) \sim GP(-, K^*)
$$

$$
\tilde{\tau}^* = \tilde{\tau} + K_{\tau\tau}(K_{tt} + R_y)^{-1}(y - \tilde{\tau})
$$

$$
K^* = K_{\tau\tau} - K_{\tau t}(K_{tt} + R_y)^{-1}(K_{tt})^T,
$$

where $y$ is the vector of known values and derivatives and $R_y$ the corresponding error (covariance) matrix. Eq. (9) now corresponds to Eq. (5) but enforces the known values and derivatives, within specified error, on any realization of the latent field.

The covariance matrix in Eq. (8) is created from a chosen continuous covariance kernel $K(x_m, x_n)$ for the covariance between the values of $\tau$ at different physical locations $x_m = [x_{m,1}, x_{m,2}, x_{m,3}]^T$ and $x_n = [x_{n,1}, x_{n,2}, x_{n,3}]^T$. The covariance between two values, between a value and a derivative, and between two derivatives are given by [25,26]

$$
K(\tau_m, \tau_n) = K(x_m, x_n)
$$

$$
K\left(\frac{\partial \tau}{\partial x_i}_m, \tau_n\right) = \frac{\partial}{\partial x_i}K(x_m, x_n)
$$

$$
K\left(\frac{\partial \tau}{\partial x_i}_m, \frac{\partial \tau}{\partial x_j}_n\right) = \frac{\partial^2}{\partial x_i \partial x_j}K(x_m, x_n),
$$

respectively. The most common covariance kernel is the square exponential kernel $K_{se}$, which depends only on the physical distance between two points. The square exponential covariance between two points is given by Eq. (11a),
where \( \sigma^2 \) is the variance and \( l_i \) for \( i \in \{1, 2, 3\} \) are the length scales in each of the three spatial coordinate directions. With this choice of kernel, Eq. (10) becomes

\[
K(\tau_m, \tau_n) = K_{se}(x_m, x_n) = \sigma^2 \exp \left[ -\frac{1}{2} \sum_{i=1}^{3} \frac{\left| x_{m,i} - x_{n,i} \right|^2}{l_i^2} \right] \quad (11a)
\]

\[
K \left( \frac{\partial \tau}{\partial x_i} \right)_m, \tau_n \right) = -\frac{1}{l_i^2} (x_{m,i} - x_{n,i}) K_{se}(x_m, x_n) \quad (11b)
\]

\[
K \left( \frac{\partial \tau}{\partial x_i} \right)_m, \left( \frac{\partial \tau}{\partial x_j} \right)_n \right) = \frac{1}{l_i^2} \left( \delta_{i,j} - \frac{1}{l_j^2} (x_{m,i} - x_{n,i})(x_{m,j} - x_{n,j}) \right) K_{se}(x_m, x_n), \quad (11c)
\]

where \( \delta_{i,j} \) is the Kronecker delta.

2.2. Boundary conditions on Gaussian processes

When enforcing Dirichlet, Neumann, and mixed boundary conditions there are several simplifications to the process described in Section 2.1. In Eq. (8) \( t \) now represent the known boundary values and derivatives at the centers of the boundary faces in the discretization. Since boundary conditions come from the modeled physics and are also enforced exactly on the output fields in the forward model, no observation errors are used, i.e. \( R = 0 \). The baseline solution \( \tau \) is assumed to satisfy the boundary conditions, and the difference between the observed value and the baseline value at the boundaries is then zero, i.e. \( y - \tau = 0 \). With these simplifications Eq. (9) becomes

\[
\tau(t = y) \sim GP \left( \tau, K_{tt} - K_{tt} K_{tt}^{-1} (K_{tt})^t \right). \quad (12)
\]

Eq. (12) shows that only the covariance needs to be updated for enforcing boundary conditions and the mean is unchanged.

Periodic boundary conditions are also very common in computational physics, where they are used in modeling problems with certain types of symmetry. Periodic boundary conditions cannot be implemented through the modifications presented in Section 2.1, instead they can be implemented by choice of covariance kernel \( K \). For instance, if the problem is periodic in the direction \( x_1 \), a mixed kernel can be used with periodic covariance in direction \( x_1 \) and square exponential in the other two directions as

\[
K(x_m, x_n) = \sigma^2 \exp \left[ -\left( 2 \frac{\sin^2(|x_{m,1} - x_{n,1}|\pi/p)}{l_1^2} + \frac{1}{2} \frac{|x_{m,2} - x_{n,2}|^2}{l_2^2} + \frac{1}{2} \frac{|x_{m,3} - x_{n,3}|^2}{l_3^2} \right) \right], \quad (13)
\]

where \( p \) is the periodicity. In order for the length scale in the periodic direction \( l_{per} \) to have a similar effect as a length scale \( l_{se} \) in the square exponential kernel, it is chosen as

\[
l_{per} \approx 6 \frac{l_{se}}{p} \quad \text{for} \quad l_{se}/p \lesssim 0.1.
\]

2.3. Constraints on functions of Gaussian processes

If the latent field is described by a function of a Gaussian process (Eq. (6)), the constraints on the latent field need to be mapped to constraints on the Gaussian process. The function \( \mathcal{G}(\delta) \) is assumed to be invertible and differentiable. If the value of the latent field at some point \( x_m \) is known to be \( \tau_m = y_{val} \), the corresponding value for the Gaussian process at that point is

\[
\delta_m = \mathcal{G}^{-1}(y_{val}). \quad (14)
\]

Similarly, if a derivative in direction \( x_i \) of the latent field at some point \( x_m \) is known to be \( (\partial \tau / \partial x_i)_m = y_{der} \), the corresponding constraints on the Gaussian process is

\[
\mathcal{G}'(\delta_m)_{m}^{(i)} = y_{der}, \quad (15)
\]

where \( \delta_m^{(i)} \) is defined as

\[
\delta_m^{(i)} = \left( \frac{\partial}{\partial x_i} \delta \right)_m.
\]
However, Eq. (15) does not have a unique solution since there are two variables, $\delta_m$ and its derivative $\delta_m^{(i)}$, and only one relation. Enforcing observations of the derivative of the latent field requires an additional constraint. Specifically, if the value at location $x_m$ is also specified, Eq. (14) can be substituted into Eq. (15). It is therefore possible to enforce observations of the value of the latent field at a point (e.g. Dirichlet B.C.) and to enforce combined observations of the value and derivative at a point (e.g. Neumann B.C.). The last type of boundary conditions considered, periodic boundary conditions, is enforced on the latent field with periodicity $p$ by requiring the covariance of $\delta$ to be periodic with the same period.

The lognormal process ($G^{-1}(\cdot) \equiv \log(\cdot))$ is useful in problems where the physical latent field must be non-negative. If the process is lognormal, then it has the form in Eq. (4), and Eqs. (14) and (15) become

$$\delta_m = \log(y_{val})$$  \hspace{1cm} (16)$$

and

$$e^{\delta_m^{(i)}} = y_{der},$$  \hspace{1cm} (17)$$

respectively. In general the derivative constraint still has infinite solutions and requires both the value and slope of the process to be defined at that point. There is however a unique solution for the special case of zero-gradient constraint, which is by far the most common derivative constraint in fluid mechanics. In this case $y_{der} = 0$ and Eq. (17) becomes

$$\delta_m^{(i)} = 0.$$  \hspace{1cm} (18)$$

One issue with the lognormal process is that it is singular when the field has a zero-value condition, i.e. $y_{val} = 0$ in Eq. (16), and the common case of a zero-valued Dirichlet boundary is explored here in more detail. The problem could be addressed numerically by having the baseline solution have a small value at the boundary. However, this would still cause the value of the Gaussian process $\delta$ in Eq. (16) to be very large, albeit finite. This is also a problem for incorporating internal observations near such boundaries, where the baseline solution is close to zero. In such a case, a small difference between known values and the baseline solution corresponds to a large magnitude of the Gaussian process, approaching infinity as the baseline solution approaches zero. This is most obvious when the normalization in Eq. (4) is made explicit and the baseline is used for normalization. This gives

$$\tau/\tilde{\tau} = e^{\delta^*}$$  \hspace{1cm} (19a)$$

and

$$\delta^* \sim GP(0, K_{\delta^*}),$$  \hspace{1cm} (19b)$$

which emphasizes the multiplicative nature of the lognormal formulation. At a location where the baseline $\tilde{\tau}$ is close to zero, even a slightly different value of the latent field $\tau$ would require a large value of the Gaussian process $\delta$. To address this singularity at the boundary we use the formulation in Eq. (19) and take the baseline solution to satisfy the boundary condition. In this case the value of the observation $y^*_{val} = y_{val}/\tilde{\tau} = 1$ and Eq. (16) becomes

$$\delta_m^* = 0$$  \hspace{1cm} (20)$$

for boundaries with zero-value Dirichlet conditions. Internal observations in the vicinity such boundaries should still be avoided. Additionally, utilizing observations of the output field near such boundaries should also be avoided. While the forward model is generally non-local (e.g. due to convection), often there still is a strong dependence on the local neighborhood, which means the inference would result in modification of the latent field near the boundary. Another consequence of this singularity and the representation in Eq. (19) is that since the baseline solution is already close to zero near the boundary, the boundary condition is enforced without need to modify the covariance. Here, however we do enforce the appropriate condition, Eq. (20), at such boundaries. In summary, when using a lognormal distribution as in Eq. (19) a zero-valued Dirichlet condition on the latent field implies the following:

(a) no internal observations of the latent field should be enforced near the boundary,
(b) observations of the output field near the boundary should be similarly avoided,
(c) the equivalent boundary condition for the Gaussian process is as in Eq. (20),
(d) however, the boundary condition is defacto enforced with no need to modify the covariance matrix.
Fig. 1. Problem description for the channel case. The geometry (a) consists of an infinite fully developed channel flow between two walls. The streamwise and wall-normal directions are \( x_1 \) and \( x_2 \), respectively, and the half-channel height is \( H \). The discretized one-dimensional simulation domain (b) consists of only one cell and periodic boundary conditions in the streamwise direction since the interest is in the developed velocity profile. Because of symmetry only the bottom half of the domain is modeled. The baseline (initial guess) and true solutions for the latent eddy viscosity field are shown in (c) and the problem consists of inferring the multiplicative discrepancy (d) between these. The discrepancy in (d) is normalized based on the standard deviation of the prior distribution at each cell.

3. Results

Two fluid flow problems are used as test cases: flow through a channel and flow over periodic hills. The latent field in both cases is the eddy viscosity, which has a positivity requirement and therefore a lognormal process is used to represent it. In Section 3.2 we perform an a priori study to highlight the methodology. The boundary conditions are enforced for both cases and an internal observation is enforced in the periodic hills case. In Section 3.3 Bayesian inversion is performed on both cases using sparse observations.

3.1. Test cases

Two test cases are used to showcase the methodology. In both cases we solve the steady, incompressible, Reynolds-averaged Navier–Stokes (RANS) equations describing the mean properties of fluid flows. The linear eddy viscosity approximation is used to represent the Reynolds stress tensor field with a single scalar field, the eddy viscosity field. The RANS equations are then

\[
\frac{\partial U_j}{\partial x_j} = 0 \tag{21a}
\]

\[
U_j \frac{\partial U_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_i} \left( \nu + \nu_t \right) \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right), \tag{21b}
\]

and correspond to the model in Eqs. (1) and (2), where the output fields \( u \) are the components of velocity \( U_i \) and a pseudo pressure term \( p \) and the latent field is the eddy viscosity \( \nu_t \). The indices \( i, j \) refer to spatial direction, e.g. \( i, j \in \{1, 2\} \) for a two-dimensional problem. The dynamic viscosity \( \nu \) is a known physical property of the fluid. The RANS equations are solved using the SIMPLE (Semi-Implicit Method for Pressure Linked Equations)
Fig. 2. Problem description for the periodic hills case. The geometry (a) consists of infinite periodic hills with bottom and top walls. The streamwise and wall-normal directions are \( x_1 \) and \( x_2 \), respectively, and the hill height is \( H \). The discretized two-dimensional simulation domain (b) consists of a single hill with periodic boundary conditions. The baseline (c) and true (d) latent eddy viscosity fields are shown, and the problem consists of inferring the multiplicative discrepancy (e) between them. The discrepancy in (e) is normalized based on the standard deviation of the prior distribution at each cell. The discrepancy (e) is capped at 5.0. Because of the singularity discussed in Section 2.3 values near the wall boundary blow up and reach up to 536 with the current discretization. However, in most of the channel this discrepancy is within one standard deviation.

The algorithm in OpenFOAM, an open-source CFD platform. The equations are discretized using the finite volume method and second-order spatial discretization scheme on an unstructured mesh. A modified solver was implemented to propagate a given eddy viscosity field to the velocity and pressure fields, corresponding to Eq. (21).

Both cases consist of wall-bounded incompressible flow driven by a pressure gradient source to achieve the desired bulk velocity. The first case is a steady, fully developed, one-dimensional channel flow with top and bottom walls, shown in Fig. 1a. The case has a Reynolds number of \( Re = 5,600 \) based on half channel height \( H \) and bulk velocity \( U_b \). To model this problem a single cell is used in the direction of flow with periodic boundary conditions on the boundaries perpendicular to the flow. Because of symmetry only the bottom half of the channel is modeled, using a symmetry boundary condition on the top boundary. The domain is discretized using 90 cells in the direction normal to flow, as shown in Fig. 1b.

The second case is the two-dimensional infinite hills geometry with top and bottom walls created by Mellen et al. [37] and shown in Fig. 2a. The case has a Reynolds number of \( Re = 5,600 \) based on the hill height \( H \) and bulk velocity through the section at the hill top \( U_b \). This is modeled as a single hill with periodic boundary conditions and the domain was discretized into 3000 cells, as shown in Fig. 2b. To highlight the periodicity, the upstream half of the computational domain is plotted starting after the downstream half in the rest of the paper. Furthermore, a sequential colormap is used for plots of quantities that are non-negative (e.g. eddy viscosity), and a diverging colormap is used for plots of signed quantities (e.g. discrepancy, KL modes).

The periodic boundary conditions require all fields to have the same values and derivatives on both boundaries while the symmetric boundary condition requires the derivative in the direction normal to the boundary to be zero. These requirements are for all fields including the eddy viscosity field. The wall boundary condition translates to different requirements for different fields. The velocities \( U_1 \) and \( U_2 \) are zero at the wall while the pressure has zero gradient normal to the wall. The wall imposes a zero-value Dirichlet condition on the eddy viscosity field since the Reynolds stresses are zero at the wall and the velocity gradient is not necessarily zero. The eddy viscosity has a positivity constraint, and this is enforced by modeling it as a lognormal process as in Eq. (4). The boundary conditions are therefore not implemented directly as described in Section 2.2 but with the modifications described in Section 2.3.

For both the channel and periodic hills cases two solutions are required, the baseline solution and a synthetic truth. The baseline solution is used as the median of the prior lognormal distribution (Eq. (4)) for the latent field. The synthetic truth is used to create synthetic observations based on perturbations of this truth and to evaluate the performance of the Bayesian inference. The RANS equations with linear eddy viscosity approximation are also used in creating the synthetic truth. This is because the RANS equations are considered to contribute little to the
uncertainty of the solution, justifying the embedded discrepancy formulation. Solving the RANS equations requires choosing a turbulence model for the eddy viscosity. Different turbulence models were chosen to create baseline and synthetic truth solutions that were sufficiently different to illustrate the methodology, and these are summarized in Table 1. For the channel case the Spalart–Allmaras [38] and \( k-\varepsilon \) [39] turbulence models are used for creating the baseline and synthetic truth solutions, respectively. For the periodic hills case the \( k-\omega \) turbulence model [40] is used for both the baseline and synthetic truth solutions, but the coefficient \( c_\mu \) is modified to \( c_\mu = 0.45 \) for the latter instead of its standard value of 0.09. The four solutions summarized in Table 1 were all obtained using the simpleFoam solver in OpenFOAM with the specified turbulence models. The synthetic observations are obtained by taking a realization of a multivariate normal distribution with mean equal to the synthetic truth and with assumed uncorrelated errors.

The baseline solution, synthetic truth, and logarithm of the multiplicative discrepancy are shown in Fig. 1 for the channel case and Fig. 2 for the periodic hills. The baseline solutions are used for the formulation in Eq. (19) and the logarithm of the multiplicative discrepancy is the value of the Gaussian process \( \delta^* \) that needs to be inferred. Note that the baseline solutions satisfy all boundary conditions. As discussed in Section 2.3, the lognormal formulation has singularities whenever the baseline solution is zero, as is the case on walls on both test cases. This can be seen in the large values for the discrepancies for both cases. Because of this we will avoid making observations near the singularities, which would drive the entire inferred field to an unrealistic solution. The inferred solutions are presented in Section 3.3, as well as quantitative measures of the discrepancies. To quantify the discrepancy \( \Delta_\theta \) between a field \( \theta \) and the true field \( \theta^* \) we use

\[
\Delta_\theta = \frac{\|\theta^* - \theta\|}{\|\theta^*\|},
\]

(22)

where \( \|\cdot\| \) denotes the \( L^2 \)-norm of the field.

3.2. Enforcing boundary conditions and internal observations

This section presents the a priori results of enforcing boundary conditions and internal observations. Section 3.2.1 presents the results of enforcing boundary conditions on the channel case. Section 3.2.2 presents the results of enforcing boundary conditions on the periodic hills case, a more complex, two-dimensional case. Finally, Section 3.2.3 presents the results of enforcing an internal observation, in addition to the boundary conditions, for the periodic hills case. In all cases it is shown that all realizations of the modified fields satisfy the boundary conditions and the internal observations.

3.2.1. Boundary conditions – channel

For the channel case the eddy viscosity has (1) a zero-value Dirichlet condition at the wall, (2) a zero-gradient Neumann condition at the symmetry plane, and (3) a periodic boundary condition in the flow direction. The boundary conditions are enforced by modifying the covariance of the Gaussian process \( \delta^* \) in Eq. (19). In general, the Dirichlet boundary condition at the wall requires the value of the Gaussian process \( \delta^* \) to be zero at the boundary. As discussed in Section 2.3 it is not necessary to enforce this condition but we still chose to do so. The zero-gradient Neumann condition at the symmetry plane requires that the gradient of \( \delta^* \) normal to the symmetry plane be zero. The periodic boundary condition is implicitly enforced since there is a single cell in the direction of periodicity.

The original covariance matrix consists of a square exponential kernel with variance \( \sigma^2 = 0.01 \) and length scale \( l_2 = 0.1H \). This is then modified to enforce the Dirichlet and Neumann boundary conditions, by using Eq. (12). The original (square exponential) and modified covariance matrices are shown in Figs. 3a and 3b, respectively.
Fig. 3. A priori results for enforcing boundary conditions in the channel case. The left column corresponds to results using the original covariance and the right column to results using the modified covariance. In each row the legend (shown in the far right) is shared between both figures. The first row shows the original (a) and modified (b) covariance matrices. The modified covariance ensures all realizations have zero gradient at the symmetry plane and zero value at the bottom wall. The second row shows the first five KL modes for each case. For the modified covariance (d) the KL modes have zero gradient at the symmetry plane and zero value at the bottom wall. The third row shows five realizations of the prior eddy viscosity field. When using the modified covariance (f) the realizations satisfy the zero gradient boundary condition at the symmetry plane.

to the bottom wall ($x_2/H = 0$), where a Dirichlet boundary condition is enforced, the modified covariance is near zero. This results in the value of the baseline near the wall being used for all realizations. Note that the boundary locations (faces) are not in the covariance matrices in Figs. 3a–3b, since the inferred quantities are the internal cell values. Near the symmetry plane ($x_2/H = 1$), where a Neumann boundary condition is enforced, there is strong covariance with the immediate neighbors and a quick decrease as you move further away. Since the baseline solution has zero gradient at this boundary, this ensures that the gradient of any realization is also zero at the boundary.

As discussed earlier, ensuring that the KL modes satisfy their corresponding conditions is enough to ensure that any realization of the latent field satisfies the boundary conditions. The original and modified covariances require 10 and 9 modes respectively to capture 99% of the variance. These number of modes are used for the truncation. Figs. 3c and 3d show the first five modes using the original and modified covariance matrices, respectively. It can be seen that the modified covariance does enforce zero gradients on the modes whereas the original covariance does not, as expected. While it was shown that this condition is sufficient to ensure the realizations of eddy viscosity
satisfy the zero gradient boundary conditions, it is helpful to actually visualize this. The realizations are created as in Eq. (7) using the truncated modes. Figs. 3e and 3f show created realizations using the original and modified covariance matrices, respectively. It can be seen that the gradient at the symmetry plane is zero for all realizations when using the modified covariance matrix, but is not the case when using the original covariance matrix.

3.2.2. Boundary conditions – periodic hills

For the periodic hills case the eddy viscosity has a periodic boundary condition in the direction of the flow and zero-value Dirichlet conditions at the top and bottom walls. The periodic boundary condition is enforced by making the covariance kernel periodic in the $x_1$ direction, as in Eq. (13), with periodicity of $9H$. Making $\delta$ periodic ensures that the eddy viscosity is also periodic with the same periodicity. The Dirichlet condition is enforced on the bottom and top walls using Eq. (12). Again, it is not necessary to enforce this condition but we still chose to do so.

The original and modified covariance matrices are constructed as before, using $\sigma = 1.0$ and $l_1 = l_2 = 0.25H$. Since this case is two-dimensional the covariance matrix is difficult to interpret and depends on the cell ordering. Instead, Figs. 4a and 4b show the covariance at a selected point for the original and modified covariance matrices, respectively. The modified covariance can be seen to be periodic and the covariance with the wall is zero, as expected. Note that the plotted results are shown with the first half of the simulation domain shifted behind the second half to highlight when the periodicity constraint is met.

Ensuring that the KL modes satisfy their corresponding conditions is enough to ensure that any realization of the latent field satisfies the boundary conditions. Figs. 4c and 4d show a selected KL mode using the original and modified covariance matrices. As can be seen the mode of the modified covariance matrix satisfies the periodicity and Dirichlet conditions, while that for the original covariance matrix do not. All other modes using the modified covariance matrix similarly satisfy the boundary conditions. For the truncation the number of modes where chosen as 317 and 192 for original and modified covariances, respectively, to cover 99% of the covariance. Figs. 4e and 3f show a realization of the eddy viscosity field using the original and modified covariance matrices. It can be seen that the modification to the covariance matrix results in the boundary conditions being enforced on the realizations of the eddy viscosity field. Specifically, the periodic and zero-value boundary conditions are satisfied for all realizations of the eddy viscosity field. Note that the zero-value boundary conditions at the bottom and top walls are also satisfied when using the original covariance as discussed in Section 2.3.

3.2.3. Internal observations - periodic hills

As a last test case the observed value of the eddy viscosity at an internal point is enforced in the periodic hills case. The observation is made at $(x_1/H, x_2/H) = (8.5, 2.5)$ and the true value of the eddy viscosity $\nu_l = 0.000795$ is used. The observation is considered exact with no observation error. The mean and covariance are modified using Eq. (9). The modified baseline eddy viscosity is shown in Fig. 5a for the cross-section $x_1 = 8.5$ passing through the observation point. The modified baseline now satisfies the observation value. The first five KL modes for the modified covariance are shown in Fig. 5b for the same cross-section, and can all be seen to have a value of zero at the observation location. Fig. 5c shows ten realizations of the eddy viscosity field at the same cross section. All realizations satisfy the internal observation, as desired.

3.3. Bayesian inversion

The inverse problem was solved for both the channel case and the periodic hills case using the Bayesian ensemble Kalman scheme presented in the Appendix. The boundary conditions where enforced during the inversion by using the modified covariance matrices in Sections 3.2.1 and 3.2.2 for the channel and periodic hills cases respectively. The inverse problem consists of inferring the eddy viscosity based on the prior distribution and sparse observations of the velocity field. As discussed in Section 2 a reduced order model is used and the state vector to be inferred consists of the coefficients $\omega = \{\omega_l\}_{l=1}^{M}$ of the first $M$ KL modes. All components of velocity are observed at sparse locations. The observation errors are assumed independent and hence the covariance matrix $R$ is diagonal. The error for each observation is constructed using a relative error of 0.001 of the true value of velocity plus an absolute error of 0.00001. The two cases are summarized in Table 2.

For the channel case two observations of velocity are used at locations $x_2/H = 0.1, 0.8$. The inversion was done twice using both the original covariance matrix and the modified covariance matrix that enforces the boundary
Fig. 4. Apriori results for enforcing boundary conditions in the periodic hills case. The left column corresponds to the results using the original covariance and the right column to the results using the modified covariance. In each row the legend (shown in the far right) is shared between both figures. The first row shows the covariance of a selected point (marked by a white circle). The modified covariance (b) ensures zero value at the wall and periodicity. The second row shows a selected mode (the 29th mode) normalized to have unit magnitude. The mode corresponding to the modified covariance (d) has zero value at both walls and is periodic. The third row shows a realization of the eddy viscosity $\nu_t/\nu$. The colorbar is capped at 200 for direct comparison with Fig. 2, but the realizations obtain maximum values of four to five times larger. The modified covariance leads to realizations (f) that satisfy the periodic boundary condition.

Fig. 5. A priori results for enforcing internal observations in the periodic hills case. Results are shown at the cross-section $x_1/H = 8.5$ where the internal observation is made. Panel (a) shows the original prior mean eddy viscosity $\overline{\nu}_t$ and the modified prior mean eddy viscosity $\overline{\nu}_t^*$, which matches the observations $y$. Panel (b) shows the first five KL modes, all of which meet the corresponding constraint that the modes be zero at the observation location. Panel (c) shows ten realizations of the eddy viscosity $\nu_t$, all of which can be seen to match the observation $y$. Note that the small apparent mismatch between the results and the constraint is an artifact of the coarse discretization and the visualization (i.e. straight lines connecting datapoints).
conditions. The inferred streamwise velocity and eddy viscosity fields for both cases are shown in Fig. 6, and the discrepancies are presented in Table 3. In both cases the velocity is improved nearly everywhere. However, the predicted eddy viscosity does not show the same level of improvement over the baseline solution, and is actually significantly worst when considering the qualitative shape. The good match between velocities in spite of the mismatch in eddy viscosity is due to the ill-posedness of the problem, i.e. the same velocity field can come from different eddy viscosity fields. This situation has been common in previous research.

Although at first the two inferred eddy viscosities might seem similarly wrong, the inferred eddy viscosity using the modified covariance matrix satisfies not only the positivity constraint but all boundary conditions. This is not the case for the inferred eddy viscosity using the original covariance matrix. At the symmetry plane location \((x_2/H = 1)\) the inferred eddy viscosity has zero gradient when using the modified covariance but not when using the original covariance. This means that the inferred eddy viscosity using the modified covariance lies within a smaller subspace of possible solutions that includes more of the physics. As can be seen from Table 3, however, enforcing boundary conditions does not necessarily results in inferred internal field closer to the truth. This is due to the problem remaining ill-posed even after enforcing the boundary conditions.

For the periodic hills case a single observation of velocity was used at location \((x_1/H, x_2/H) = (8.5, 2.5)\). The inferred eddy viscosity field is shown in Fig. 7c and profiles of both the inferred eddy viscosity and the propagated velocity are shown in Figs. 7a and 7b, respectively. The discrepancy of the baseline and inferred fields compared to the truth are calculated as in Eq. (22). The pseudo-pressure field is not shown since the true value is zero and therefore the discrepancy only reflects numerical errors.

\[
\text{Table 3} \\
\text{A posteriori results (Bayesian inversion) for enforcing boundary conditions in the channel case. The discrepancy of the baseline and inferred fields compared to the truth are calculated as in Eq. (22). The pseudo-pressure field is not shown since the true value is zero and therefore the discrepancy only reflects numerical errors.}
\]

<table>
<thead>
<tr>
<th></th>
<th>Base</th>
<th>Inferred (Original)</th>
<th>Inferred (Modified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_1)</td>
<td>63.8%</td>
<td>43.9%</td>
<td>48.8%</td>
</tr>
<tr>
<td>(U_1)</td>
<td>5.1%</td>
<td>1.3%</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

The inferred fields could be further improved by using regularization to enforce expected qualities such as smoothness, but it should be noted that the boundary conditions are hard physical constraints that should be enforced regardless of any additional regularization. In both cases the inferred field shows spatial variation at a shorter correlation length and higher magnitude, which could be addressed through regularization by either penalizing the departure from the prior, as discussed earlier, or enforcing smoothness. Both of these can be implemented using the regularized ensemble Kalman filter presented by Zhang et al. [35], which enforces constraints in a Bayesian

\[
\text{Table 2} \\
\text{Summary of the two Bayesian inversion cases. The locations where velocity (output field) is observed and the boundary conditions on the eddy viscosity (latent field) are summarized. For the channel case two observations are used and for the periodic hills a single observation is used.}
\]

<table>
<thead>
<tr>
<th></th>
<th>Observation locations</th>
<th>Boundary conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>(x_2/H = 0.1)</td>
<td>(v_1 = 0) (bottom)</td>
</tr>
<tr>
<td></td>
<td>(x_2/H = 0.8)</td>
<td>(\partial v_1/\partial x_2 = 0) (top)</td>
</tr>
<tr>
<td>Periodic hills</td>
<td>((x_1/H, x_2/H) = (8.5, 2.5))</td>
<td>(v_1 = 0) (bottom, top)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Periodic (inlet, outlet)</td>
</tr>
</tbody>
</table>
Fig. 6. A posteriori results (Bayesian inversion) for enforcing boundary conditions in the channel case. The top row shows the results using the original covariance matrix and the bottom row shows the results using the modified covariance matrix. Both the inferred latent eddy viscosity field (left column) and the propagated streamwise velocity field (right column) are shown for both cases. The inferred viscosity using the modified covariance (c) is similar to the one inferred using the original covariance (a) except it obeys the zero gradient boundary condition at the symmetry plane. The propagated posterior velocities, shown in panels (b) and (d), show the same level of improvement and are nearly identical to each other.

Fig. 7. A posteriori results (Bayesian inversion) for enforcing boundary conditions in the periodic hills case. Panels (a) and (b) show profiles of eddy viscosity and streamwise velocity, respectively, at five cross-sections: $x_1 = 0, 2.5, 4.5, 6.5, \text{ and } 8.5$. The prior mean and posterior mean are compared to the truth and observation values. Panel (c) shows the full inferred (posterior mean) eddy viscosity field $\nu_t/\nu$. The colorbar is capped at 200 for direct comparison with Fig. 2 but the result has a maximum of 267. The velocity can be seen to improve in the entire field, and the eddy viscosity can be seen to satisfy all boundary conditions (zero slip at walls and periodicity along $x_1$).
Table 4
A posteriori results (Bayesian inversion) for enforcing boundary conditions in the periodic hills case. The discrepancy of the baseline and inferred fields compared to the truth are presented. These are calculated as in Eq. (22).

<table>
<thead>
<tr>
<th></th>
<th>Base</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_t$</td>
<td>73.1%</td>
<td>38.1%</td>
</tr>
<tr>
<td>$U_1$</td>
<td>9.6%</td>
<td>3.7%</td>
</tr>
<tr>
<td>$U_2$</td>
<td>50.9%</td>
<td>22.8%</td>
</tr>
<tr>
<td>$p^*$</td>
<td>53.0%</td>
<td>16.7%</td>
</tr>
</tbody>
</table>

manner. Smoothness for instance can be enforced by preferring smaller values of the norm of the first derivative over the field or constraining the maximum value of the second derivative.

It is noticeable that just one or two observations of the mean velocity result in global improvements of the mean velocity field. In both cases the observations locations where determined by choosing one point where the velocity discrepancy was large, and in the case of the channel flow adding a second observation due to lack of global improvement. There are two factors likely driving the global improvement from sparse observations. First, the velocity field has an inherent spatial correlation due most strongly to convection. In particular for the periodic hills case, convection and diffusion cause a correction at one location to have influence on all points on some vicinity of the observations, with larger correlation length along the streamline. Second, for both cases, the flow is driven by a pressure gradient that ensures a specified bulk velocity. The mass conservation dictates that a reduction of the velocity magnitude in one region necessarily requires an increase in another region, even if they are on different streamlines. This mechanism is believed to drive the improvements in areas that are not correlated to the observation point through convection–diffusion, such as the recirculation region in the periodic hills.

4. Conclusions

Problems in computational physics often consist of a forward model that solves a set of partial differential equations and associated boundary conditions for the output fields of interest. The forward model typically has latent physical fields that need to be estimated. For instance, the Reynolds-averaged Navier–Stokes equations describing the mean velocities and pressure of fluid flows contain the unclosed Reynolds stress field. In the inverse problem these latent fields can be treated as model parameters to be inferred. The inverse problem then consists of inferring these latent fields from sparse observations of the output fields. As is the case with the Reynolds stress, these latent fields often have clear physical meaning and constraints. However, the inherent ill-posedness of inverse problems means that numerous possible latent fields can result in improvements in the fields of interest and agreement with observations. Since the focus is usually on the fields of interest, improvements have often been obtained with unphysical latent fields. Particularly, the boundary conditions on these physical latent fields have been often ignored.

In this paper we demonstrate how to enforce known boundary conditions on the latent fields in Bayesian inversion of physical fields. This is done by appropriate choice of the statistical model used to represent the random latent fields. While this does not necessarily result in the true latent field (i.e. the problem is still ill-posed) it does ensure the inferred latent fields satisfy these important physical constraints. In particular, we start with a choice of statistical model consisting of a Gaussian process or a function of a Gaussian process with an initial covariance matrix and modify this matrix in a way that ensures all realizations of the field satisfy the boundary conditions. We demonstrate how to enforce Dirichlet, Neumann, and mixed boundary conditions in this fashion. Particular attention is given to the lognormal process which is a common choice of statistical model for scalar fields with positivity constraint. Some idiosyncrasies of the lognormal model are explored, particularly the implications of the singularity at boundaries with zero-value Dirichlet conditions. One consequence of this singularity is that observations should be avoided at locations near such boundaries. Additionally we demonstrate how to enforce periodic boundary conditions by choice of the initial covariance kernel and how to enforce internal observations of the latent field by modifying not just the covariance but also the mean of the Gaussian process.

As a test case, an iterative ensemble Kalman method was used as the Bayesian framework, and the Reynolds-averaged Navier–Stokes equations describing the mean velocity and pressure of fluid flows were used as the forward model. The inverse problem was solved for the eddy viscosity field from sparse observations of the velocity. The
manner. Smoothness for instance can be enforced by preferring smaller values of the norm of the first derivative over the field or constraining the maximum value of the second derivative.

It is noticeable that just one or two observations of the mean velocity result in global improvements of the mean velocity field. In both cases the observations locations where determined by choosing one point where the velocity discrepancy was large, and in the case of the channel flow adding a second observation due to lack of global improvement. There are two factors likely driving the global improvement from sparse observations. First, the velocity field has an inherent spatial correlation due most strongly to convection. In particular for the periodic hills case, convection and diffusion cause a correction at one location to have influence on all points on some vicinity of the observations, with larger correlation length along the streamline. Second, for both cases, the flow is driven by a pressure gradient that ensures a specified bulk velocity. The mass conservation dictates that a reduction of the velocity magnitude in one region necessarily requires an increase in another region, even if they are on different streamlines. This mechanism is believed to drive the improvements in areas that are not correlated to the observation point through convection–diffusion, such as the recirculation region in the periodic hills.

4. Conclusions

Problems in computational physics often consist of a forward model that solves a set of partial differential equations and associated boundary conditions for the output fields of interest. The forward model typically has latent physical fields that need to be estimated. For instance, the Reynolds-averaged Navier–Stokes equations describing the mean velocities and pressure of fluid flows contain the unclosed Reynolds stress field. In the inverse problem these latent fields can be treated as model parameters to be inferred. The inverse problem then consists of inferring these latent fields from sparse observations of the output fields. As is the case with the Reynolds stress, these latent fields often have clear physical meaning and constraints. However, the inherent ill-posedness of inverse problems means that numerous possible latent fields can result in improvements in the fields of interest and agreement with observations. Since the focus is usually on the fields of interest, improvements have often been obtained with unphysical latent fields. Particularly, the boundary conditions on these physical latent fields have been often ignored.

In this paper we demonstrate how to enforce known boundary conditions on the latent fields in Bayesian inversion of physical fields. This is done by appropriate choice of the statistical model used to represent the random latent fields. While this does not necessarily result in the true latent field (i.e. the problem is still ill-posed) it does ensure the inferred latent fields satisfy these important physical constraints. In particular, we start with a choice of statistical model consisting of a Gaussian process or a function of a Gaussian process with an initial covariance matrix and modify this matrix in a way that ensures all realizations of the field satisfy the boundary conditions. We demonstrate how to enforce Dirichlet, Neumann, and mixed boundary conditions in this fashion. Particular attention is given to the lognormal process which is a common choice of statistical model for scalar fields with positivity constraint. Some idiosyncrasies of the lognormal model are explored, particularly the implications of the singularity at boundaries with zero-value Dirichlet conditions. One consequence of this singularity is that observations should be avoided at locations near such boundaries. Additionally we demonstrate how to enforce periodic boundary conditions by choice of the initial covariance kernel and how to enforce internal observations of the latent field by modifying not just the covariance but also the mean of the Gaussian process.

As a test case, an iterative ensemble Kalman method was used as the Bayesian framework, and the Reynolds-averaged Navier–Stokes equations describing the mean velocity and pressure of fluid flows were used as the forward model. The inverse problem was solved for the eddy viscosity field from sparse observations of the velocity. The
The ensemble Kalman method was tested on two different flows: a one-dimensional channel flow and two-dimensional flow over periodic hills. The results were compared to Bayesian inversion using the initial unmodified covariance, which does not enforce the boundary conditions. This comparison shows that the method is successful in enforcing boundary conditions on the inferred latent field while still resulting in similar improvements on the fields of interest and similar agreement with observations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This material is based on research sponsored by the U.S. Air Force under agreement number FA865019-2-2204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Appendix. Framework for Bayesian field inversion

This appendix presents a summary of the Bayesian field inversion framework used in this paper. The framework is adopted from Xiao et al. [5] and is based on the iterative ensemble Kalman method developed by Iglesias et al. [9]. Ensemble methods are used in practice to avoid calculating the derivatives (e.g. via adjoint methods) of complex computational physical models and use a finite number of samples to represent the distribution of stochastic fields. The ensemble Kalman method is presented first followed by the iterative ensemble Kalman method used in this work.

In the ensemble Kalman method the distribution for the state vector $\omega$ describing the field is represented by a finite number of $N$ samples. The samples of the prior distribution are denoted $\hat{\omega}^{(j)}$ for $j \in \{1, 2, \ldots, N\}$. The observations consist of observation values $d$ and observation error $R$ (covariance matrix). The distribution representing the observations is also represented using $N$ samples $d^{(j)}$ for $j \in \{1, 2, \ldots, N\}$ obtained from the multivariate Gaussian distribution $N(d, R)$. The operator $\mathcal{H}$ is the general mapping from state space to observation space and is linearized as $H$, i.e.

$$\mathcal{H}(\omega) \approx H\omega. \quad (A.1)$$

Given these observations each of the prior samples is updated as

$$\omega^{(j)} = \hat{\omega}^{(j)} + K_g (d^{(j)} - H\hat{\omega}^{(j)}) , \quad (A.2)$$

to obtain the posterior distribution described by the samples $\omega^{(j)}$. The Kalman gain matrix $K_g$ is given by

$$K_g = PH^\top(HPH^\top + R)^{-1} \quad (A.3a)$$

or

$$K_g = \frac{1}{N} (\omega - \bar{\omega})(H\omega - \bar{H}\omega) \left[ \frac{1}{N} (H\omega - \bar{H}\omega)(H\omega - \bar{H}\omega)^\top + R \right]^{-1} , \quad (A.3b)$$

where $P$ is the sample covariance, $\omega = [\omega^{(1)}, \ldots, \omega^{(N)}]$ is the matrix of all samples, and $\bar{\omega}$ is the mean over all samples. The formulation in Eq. (A.3b) is obtained by using the definition of the sample covariance $P$ and avoids creating the matrix $H$ by directly using the values of the samples in observation space $H\omega$. The values $H\omega$ can often be obtained in a computationally efficient way avoiding performing this large matrix multiplication.

The ensemble Kalman method assumes linearity in the mapping from state space to observation space as shown in Eq. (A.1). In the problems considered here the mapping from state space to observation space consists of two steps: (1) a set of PDEs for the output fields $u$ denoted by the operator $\mathcal{F}$, and (2) an observation operator $\mathcal{H}_0$ that maps from the output fields to the observation space. That is,

$$\mathcal{H}(\omega) = \mathcal{H}_0(\mathcal{F}(\omega)) . \quad (A.4)$$

We can avoid linearization of $\mathcal{H}$ if the model $\mathcal{F}$ is nonlinear and use $\mathcal{H}(\omega)$ directly in Eqs. (A.2) and (A.3b) in place of $H\omega$. These values are obtained as in Eq. (A.4) or with the linearization of the operator $\mathcal{H}_0 \approx H_0$ as $\mathcal{H}_0$

$$\mathcal{H}(\omega) \approx H_0\mathcal{F}(\omega) . \quad (A.5)$$
In this case an iterative approach is used where the posterior $\omega^{(j)}$ samples are used as the new prior samples and the observation data ensemble is resampled each iteration. This is done until the mean value of the posterior samples converges. This process is summarized as

$$\omega^{(j)}_{(n+1)} = \omega^{(j)}_{(n)} + K_{g(n)} \left( d^{(j)}_{(n)} - \mathcal{H}(\omega^{(j)}_{(n)}) \right)$$  \hspace{1cm} (A.6)

with $\omega^{(0)} = \tilde{\omega}^{(0)}$, where the subscript $(n)$ indicates the iteration step. The Kalman gain at each iteration is given as

$$K_{g(n)} = \frac{1}{N} \left( \tilde{\omega}_{(n)} - \bar{\omega}_{(n)} \right) \left( \frac{1}{N} \left( \mathcal{H}(\omega) \right) \left( \mathcal{H}(\omega) \right)^\top + R \right)^{-1},$$  \hspace{1cm} (A.7)

where a tilde indicates mean subtracted values, i.e.

$$\tilde{\omega}_{(n)} = \omega_{(n)} - \bar{\omega}_{(n)}$$

$$\mathcal{H}(\omega)_{(n)} = \mathcal{H}(\omega)_{(n)} - \mathcal{H}(\bar{\omega})_{(n)},$$

and the operator $\mathcal{H}$ operating on a matrix is the matrix of $\mathcal{H}$ operating on each column. Note that each iteration requires $N$ evaluations of the model.

One disadvantage of the iterative EnKF is the collapse of the sample variance, which results in an accurate posterior mean but loses any information on the posterior variance. This can be addressed by using different or modified methods other than the ensemble Kalman filter. The ensemble Kalman filter with multiple data assimilation (EnKF-MDA) [41] performs the inversion over multiple steps with inflated observation error. The ensemble randomized maximum likelihood (EnRML) [42,43] is an ensemble implementation of the randomized maximum likelihood method which is based on randomizing the likelihood function. A recent approach to address the sample collapse is the projected Stein variational Newton (pSVN) method [44], where the state is first projected to a data-informed low-dimensional subspace.

References


Chapter 6

Conclusions and Future Work
This dissertation explored two approaches to data-driven turbulence modeling for the RANS equations. In both, the data-driven techniques are used to predict or correct the Reynolds stress, which is the source of uncertainty in RANS simulations. The first approach consists of training a predictive turbulence model using machine learning. Such a model could be trained on a number of flows representing a diverse set of turbulence physics, and later used as a predictive model for unseen flows. This dissertation presents a fully differentiable framework for training deep neural networks turbulence models from indirect observations, i.e. from observations of velocity and pressure rather than Reynolds stress. Two different methods for obtaining the gradient information are explored: the continuous adjoint equations and ensemble gradient approximations.

The second approach explored in this dissertation is based on field inversion for data assimilation. For a specific flow of interest, the goal is to infer the input Reynolds stress field for the RANS equations from sparse observations of the output velocity and pressure fields. This is an ill-posed inverse problem. The inversion is done by Kalman filtering, a Monte Carlo, Bayesian approach with the Reynolds stress field from a traditional turbulence model as the prior. This methodology combines two sources of information — the prior model predictions and the sparse measurements — and enforces the constraints imposed by the RANS equations, to infer an improved Reynolds stress field. Unlike the machine learning approach this does not result in a predictive model, rather it combines different sources of information for the same flow to seek improved predictions. This approach could play an important role in data driven turbulence modeling for two reasons. First, while data-driven predictive models are promising there is no such deployment-ready general model and RANS modeling remain inaccurate. Second, it is already common in engineering practice to perform both RANS simulations and experimental tests. The data assimilation approach would provide a framework for combining these two sources of information which are typically already available. This dissertation presents a code developed for this purpose as well as efforts to embed prior physics knowledge during the inversion. While some more development is needed, in my opinion this approach is closer to being useful in practice than a general predictive model.

The rest of this section discusses possible future work and challenges related to developing a general predictive data-driven turbulence model using machine learning. Particularly, we advocate the use of deep learning as the machine learning paradigm, and the use of indirect observations for training. The challenges these pose are discussed below. Finally, this section ends with a discussion of possible alternative data-driven approaches to turbulence modeling other than that presented here.

6.1 Deep learning with indirect observations

Deep learning [1] is a class of machine learning techniques characterized by sequential non-linear function composition and is by far the dominant machine learning approach. This is mostly due to its demonstrated superior predictive capabilities that have revolutionized
numerous fields including machine vision and natural language comprehension [2]. This breakthrough is accredited to both its exponential expressive power [3] and to the algorithmic improvements to the gradient-based optimization and regularization methods. It is therefore not surprising that deep learning found its way into data-driven turbulence modelling [4]. We advocate the use of deep neural networks and have used it in the work presented in this dissertation. However there are several challenges associated with deep learning for turbulence modeling, which are discussed below.

In general, deep neural networks are difficult to train, due to several reasons, including the large parameter space and the vanishing gradient problem whereby the sensitivity of the outputs to earlier layers becomes vanishingly small. Effectively training deep networks has only become possible due to algorithmic improvements and innovation both in the training procedure and regularization techniques. Examples include gradient descent algorithms, such as the ADAM algorithm, that include adaptive gradients to deal with the vanishing gradient problem, novel random initialization techniques such as the Kaiming initialization, input normalization, and regularization techniques such as early stopping and dropout.

Training with indirect observations poses additional challenges to training deep neural networks since predicting these quantities requires solving an additional model, i.e. the RANS equations, prior to comparing with the training data. The first challenge is that the gradient information for the RANS equations is traditionally not trivial to obtain. In this work we used both the continuous adjoint equations and an ensemble approximation to obtain this gradient information. Fortunately, there is an emerging push for differentiable programming in scientific computing which promises to make such gradient information more readily available. This includes modern programming languages that come with built-in automatic differentiation [5], or neural-network-based solutions of partial differential equations [6].

The second challenge of training deep networks with derived quantities is that several of the accepted best practice training methods cannot be employed. As discussed in Chapter 2, random initialization, which has been shown to significantly improve training, cannot be employed since it results in a random Reynolds stress field and a diverging RANS simulation. Similarly, the input scaling was modified to an iterative scaling since the inputs are iteratively updated based on the RANS solutions. More research is needed to develop improved training methods specifically for turbulence model training with indirect observations. This is expected to have a large effect, given the sensitivity of training deep neural networks to such algorithmic implementations.

Another challenge of training with derived quantities is the need to solve the RANS equations for each training flow at each training step. Neural networks are typically trained for hundreds of thousands of steps, which becomes unfeasible when having to solve the RANS equations at each step. The problem is compounded by the number of training flows. Addressing this likely requires both developing novel, more efficient training algorithms as discussed above, and a judicious or systematic choice of a representative set of training flows.

Finally, we would like to discuss the issue of interpretability which is considered the main
disadvantage of neural networks. Several data-driven turbulence modelling works have used machine learning algorithms that learn symbolic expressions in terms of elementary functions. Weatheritt and Sandberg [7] and Zhao et al. [8] use gene expression programming, while Schmelzer et al. [9] and Beetham and Capecelatro [10] use linear sparse symbolic regression. We recognize the advantages of having interpretable expressions and point out the method in Martius and Lampert [11] that learns symbolic expressions within a deep learning framework. We also point out that interpretability is potentially at odds with generalizability, since it is uncertain whether a single-point closure model exists. In order to account for diverse flow regimes, a general symbolic model could potentially be so complex as to be uninterpretable. One option for interpretable data-driven models is a zonal approach (e.g. Zhao et al. [8]) where different models are learned for different flow regimes. The zonal approach to RANS modeling, however, requires modeling the transition between zones which can be a source of uncertainty. On the other hand, deep neural networks with rectified linear (ReLU) activation functions can be interpreted as a combination of a large number of simpler models [12] and a turbulence model represented by a deep network can be seen as a unified turbulence model that intrinsically learns closures in different flow regimes and their transitions.

6.2 Approaches to data-driven RANS turbulence modeling

In this work we learned eddy viscosity models, which are single-point closures that assume the Reynolds stress tensor field to be a local function of the velocity gradient. The approach taken here is that of learning the invariant mapping from the normalized velocity gradient to the Reynolds stress anisotropy, while retaining traditional transport equations for the turbulence scales—i.e. the turbulent kinetic energy and the turbulent time scale, which is typically obtained by modeling the turbulent dissipation rate. While this is a theoretically sound approach, it suffers from still using the traditional transport equations for the turbulence scales. These transport equations are both highly empirical and more importantly they are likely inconsistent with the data-driven closures as Taghizadeh et al. [13] point out. Specifically, the closure coefficients in traditional models were typically chosen such that the LEVM-predicted turbulence satisfies known asymptotic or equilibrium behaviours, which are no longer satisfied when the linear term is modified and additional non-linear terms are included. The best choice of turbulence scales to model is an open question in turbulence modeling and one unlikely to be solved soon. It is generally accepted that a true closure might not exist and that particularly a true two turbulence quantities model likely does not exist. However, there is room to improve the existing turbulence transport equations to be consistent with the modified closures.

Another approach to data-driven eddy viscosity models is keeping a traditional constant linear closure (e.g. $C_\mu = 0.09$) but learning corrections to the production terms in the turbulence scales transport equations [e.g. 14, 15]. Neither of these approaches are completely
correct since both the closure form and the transport equations are uncertain. A possible area of future research is using data-driven models to tackle both of these uncertainties. However, for simple flows, addressing either source of uncertainty can provide improved results (e.g. overfitting) and regularizing the training in such a way as to separate the effects of each component requires some thought.

Finally, it is noted that eddy viscosity models are not the only option for RANS turbulence modelling. Reynolds stress transport equations directly model the Reynolds stress components and capture non-local dynamics. These models are more complex than eddy viscosity model but capture more of the physics. It would therefore be worth exploring the use of data-driven models for the uncertain terms in Reynolds stress transport equations, such as the rapid pressure-strain correlation term. Such an effort could benefit from the work already done in data-driven eddy viscosity models, including embedding known physical knowledge such as invariance into the model rather than learning it, training with indirect observations, and developing fully differentiable frameworks for gradient-based training, e.g. of deep neural networks.

Bibliography


