

# Ensemble gradient for learning turbulence models from indirect observations

Carlos A. Michelén Ströfer<sup>\*\*1</sup>, Xin-Lei Zhang<sup>\*\*1,2</sup>, and Heng Xiao<sup>1,\*</sup>

<sup>1</sup> Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, Virginia, USA

<sup>2</sup> Arts et Métiers ParisTech, 8 Boulevard Louis XIV, 59046 Lille cedex, France

<sup>\*\*</sup> These authors contributed equally to this work.

---

**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 sensitivities 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 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 equations. 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 limitations of the ensemble gradient approximation are discussed, in particular as compared to the adjoint equations.

**AMS subject classifications:** 76F99, 76M99, 65Z05

**Key words:** 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  $\boldsymbol{\tau}$  which captures the effects of turbulence on the mean flow and requires modeling. The incompressible steady RANS equations are

$$\begin{aligned} \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla \cdot \boldsymbol{\tau} + \nabla p - \mathbf{s} &= \mathbf{0}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{1.1}$$

---

\*Corresponding author. *Email addresses:* [hengxiao@vt.edu](mailto:hengxiao@vt.edu) (H. Xiao)

where  $\mathbf{s}$  are the external body forces. Compactly, this can be written as  $\mathcal{M}(\mathbf{u}, p; \boldsymbol{\tau}) = 0$  where the Reynolds stress  $\boldsymbol{\tau}$  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 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 (NLEVM) 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 can be impractical, which has led to the use of measurements derived from the velocity and pressure fields 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 derivative 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 in this work are the same as in [4] except for the use of the ensemble gradient in place of the adjoint. This framework is summarized here.

The Reynolds stress can be separated into isotropic and anisotropic components as

$$\boldsymbol{\tau} = 2k\mathbf{b} - 2k\frac{1}{3}I, \quad (1.2)$$

where  $\mathbf{b}$  is the normalized anisotropic (deviatoric) component of the Reynolds stress,  $k$  is the turbulent kinetic energy, and  $I$  is the second rank identity tensor. An eddy viscosity turbulence model is an invariant mapping from the mean flow velocity gradient to the Reynolds stress tensor,  $\nabla\mathbf{u} \mapsto \boldsymbol{\tau}$ . Any such mapping can be expressed using the integrity basis representation [6] as

$$\mathbf{b} = \sum_{i=1}^{10} g^{(i)} T^{(i)}, \quad (1.3)$$

$$g^{(i)} = g^{(i)}(\theta_1, \dots, \theta_5),$$

$T^{(i)}$  are the tensor functions in the integrity basis,  $g^{(i)}$  are the scalar coefficient functions, and  $\theta_j$  are the scalar invariants of the input velocity gradient tensor. The full list of basis tensors and input invariants are presented in [6] and the linear and quadratic terms used here are

$$\begin{aligned} T^{(1)} &= S, & T^{(2)} &= SR - RS, \\ T^{(3)} &= S^2 - \frac{1}{3}\{S^2\}I, & T^{(4)} &= R^2 - \frac{1}{3}\{R^2\}I, \end{aligned} \quad (1.4)$$

and

$$\theta_1 = \{S^2\}, \quad \theta_2 = \{R^2\}, \quad (1.5)$$

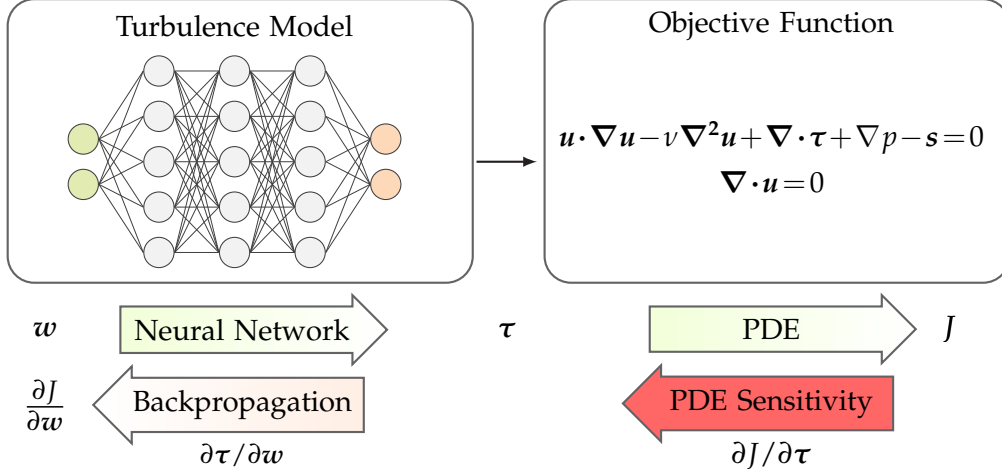


Figure 1: Schematic of the training framework adapted from [4]. For any value of the network’s parameters,  $w$ , the gradient of the objective function,  $J$ , can be obtained by combining the gradients from backpropagation of the neural network and the gradient of the PDEs (e.g. RANS). The PDE gradients (highlighted in red/dark grey) are obtained here by using ensemble approximations, which is in contrast to that in [4] where it is obtained by solving the RANS adjoint equations.

where  $S$  and  $R$  are the non-dimensionalized symmetric and antisymmetric components of the velocity gradient tensor. Equations (1.2)–(1.5) together with transport equations for the turbulent kinetic energy  $k$  and at least one more turbulent scale—to obtain the turbulence time scale for non-dimensionalizing the velocity gradient—constitute a complete eddy viscosity model. Data-driven eddy viscosity models retain the turbulent scales transport equations from a traditional model but learn the closure form, i.e. the functions  $g^{(i)}(\theta_1, \dots, \theta_5)$ , from data. Here, a deep neural network is used to represent this mapping,  $\theta \mapsto g$ .

Training the neural network with indirect observations, i.e. with velocity and pressure quantities rather than with Reynolds stress, requires derivative information for 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 trainable parameters 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

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \tau} \frac{\partial \tau}{\partial w}. \quad (1.6)$$

Here the sensitivity of cost function to the Reynolds stress  $\partial J/\partial \tau$  is obtained via an ensemble approximation rather than by solving the adjoint equations as in [4].

## 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 sensitivities of the RANS equations. Three options are (i) using gradient-free optimization to avoid the gradient calculation, (ii) solving the adjoint equations to obtain the exact gradient, or (iii) approximating the gradient from multiple evaluations of the RANS model.

**Gradient-free optimization** Gradient-free optimization updates an ensemble of states based on a heuristic other than gradient-descent, such as natural selection in genetic algorithms. Zhao et al. [5] used gene expression programming to train a model in a gradient-free manner. Although this can be an effective approach, using the gradient information is more efficient and should be used if available [7]. In particular, the recent success of deep learning methods [8] is in large part do to the use of novel gradient-based optimization methods.

**Adjoint equations** Solving the adjoint equations is an efficient method for obtaining the exact derivative information [9–11]. Both the continuous adjoint [4] and the discrete adjoint [12] have been used to obtain the sensitivities of the RANS equations and train a deep neural networks using gradient-based optimization. The adjoint equations, however, can suffer from known instabilities making it difficult to converge [13]. More importantly, the adjoint method is intrusive requiring significant effort, e.g. to derive continuous adjoint equations for a new problem or to implement the discrete adjoint on existing large codes via automatic differentiation tools.

**Numerical approximations** In this work we explore the use of approximate derivatives using ensemble methods in place of the exact derivatives 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 numerical approximations of model derivatives have been devised, including the finite difference [14], simplex gradient [15], simultaneous perturbation stochastic approximation (SPSA) [16], and ensemble-based optimization (EnOpt) [17]. 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, a modified SPSA [18] or stochastic Gaussian search direction (SGSD) method draws multiple ( $N < D$ ) samples from a 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 process and has the benefit of significantly reducing the number of 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 [18]. The connections between the ensemble-based optimization and other methods is discussed in [19], where it is shown that the 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 [20] and ensemble maximum randomized likelihood method [21]. Ensemble-based data assimilation performs an implicit optimization, but it essentially uses an explicit approximate stochastic gradient based on the ensemble realizations [19, 22].

### 1.3 Contribution of present work

In this work we demonstrate the use of the ensemble gradient approximation to train a data-driven turbulence model represented by a deep neural network. Specifically, we combine the

ensemble approximations of the RANS sensitivities with the gradients of the neural network to obtain a fully differentiable training framework. While we use it for turbulence modeling we point out that this differentiable *deep neural network plus PDEs* framework could be used for differentiable learning of any physical model (represented by the neural network) from indirect observations, i.e. observations requiring the solutions of additional models (PDEs). The ensemble gradient represents a non-intrusive method for obtaining gradients that can treat the PDEs as a black box model. 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 [23]. Here, we use both the EnOpt method [17] 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 a 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, including a comparison of the computed gradients between different methods and the adjoint as well as two test cases of learning turbulence models from velocity observations. 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 reason why it fails is discussed. Next, the common approach of preconditioning the gradient descent with the state covariance is presented. Lastly, the method used in this work, which is based on direct projection onto the subspace spanned by the samples, is presented. This section concludes with a summary of the application of these different methods to the problem of training data-driven NLEVM with indirect observations. Numerical comparison of these methods will be shown in Section 3.1.

### 2.1 Direct ensemble gradient

The cost function to be minimized is the least squares discrepancy between the observations  $y$  and predictions  $z$ , given as

$$J(x) = \frac{1}{2} \|z - y\|_{C_y}^2, \quad (2.1)$$

$$z = \mathcal{H}(x), \quad (2.2)$$

where  $x$  is the state vector to be optimized,  $\|\cdot\|_W^2$  indicates the L-2 norm weighted by the matrix  $W$ ,  $y$  is the vector of measurement data, and  $C_y$  is the measurement covariance matrix representing measurement uncertainties. For a given state  $x$  the model predictions  $z$  are the state mapped to observation space by means of the non-linear observation operator  $\mathcal{H}$  that maps from state space to observation space. For gradient descent optimization the state is updated as

$$x_{n+1} = x_n - \alpha_n \nabla J(x_n), \quad (2.3)$$

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 2.1 is

$$\nabla J(\mathbf{x}) = (\mathcal{H}'(\mathbf{x}))^\top \mathbf{C}_y^{-1} (\mathcal{H}(\mathbf{x}) - \mathbf{y}), \quad (2.4)$$

where  $\mathcal{H}'(\mathbf{x})$  is the sensitivity matrix of the vector  $\mathcal{H}(\mathbf{x})$  with respect to the state  $\mathbf{x}$ . The matrix  $\mathbf{C}_y$  and vector  $\mathbf{y}$  are fixed while the vector  $\mathcal{H}(\mathbf{x})$  is obtained by solving the forward (e.g. RANS) model. The sensitivity matrix  $\mathcal{H}'(\mathbf{x})$  is estimated using an ensemble gradient approximation as

$$\mathcal{H}'(\mathbf{x}) \approx [\mathcal{H}(\mathbf{x}^{(i)}) - \overline{\mathcal{H}(\mathbf{x})}] [\mathbf{x}^{(i)} - \bar{\mathbf{x}}]^+ = \Delta_z \Delta_x^+, \quad (2.5)$$

where for a quantity  $\phi$  the matrix  $\Delta_\phi = [\phi^{(1)} - \bar{\phi}, \dots, \phi^{(N)} - \bar{\phi}]$  consists of the mean-subtracted samples,  $\phi^{(i)}$  indicates the  $i^{\text{th}}$  sample,  $\bar{\phi}$  is the ensemble mean, and the superscript  $+$  indicates the pseudoinverse. However, the inverse problem described by Equation (2.1) is ill-posed in that multiple states  $\mathbf{x}$  can result in  $\mathcal{H}(\mathbf{x}) = \mathbf{y}$  or at least  $\mathcal{H}(\mathbf{x}) \approx \mathbf{y}$  up to high accuracy. This inherent ill-posedness can result in a cost function and sensitivity matrix with frequent, abrupt changes for small changes in the state  $\mathbf{x}$ . 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  $\nabla J(\mathbf{x})$  by the state covariance  $\mathbf{C}_x$  is a common method for preconditioning of the gradient descent optimization, as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \mathbf{C}_x \nabla J(\mathbf{x}_n), \quad (2.6)$$

referred to as EnOpt [17]. This correspond to steepest descent rather than gradient descent on a discrete vector space with inner-product defined by the covariance  $\mathbf{C}_x$  [24]. 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

$$\mathbf{C}_x \approx \frac{1}{N} \Delta_x \Delta_x^\top. \quad (2.7)$$

The product of the state covariance and the forward model sensitivity, which arises when substituting Equation (2.4) into Equation (2.6), can be approximated using the ensemble as

$$\begin{aligned} \mathbf{C}_x \mathcal{H}'(\mathbf{x}) &\approx \frac{1}{N} \Delta_x \Delta_x^\top (\Delta_y \Delta_x^+)^\top \\ &= \frac{1}{N} \Delta_x \Delta_x^\top (\Delta_x^\top)^+ \Delta_y^\top \\ &= \frac{1}{N} \Delta_x \Delta_y^\top \\ &\approx \mathbf{C}_{xz}, \end{aligned} \quad (2.8)$$

where  $\mathbf{C}_{xz}$  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 (2.5) is noisy, the ensemble cross-covariance  $\mathbf{C}_{xy}$  captures the correct qualitative correlations [25]. Finally, the update scheme in Equation (2.6) becomes

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \mathbf{C}_{xz} \mathbf{C}_y^{-1} (\mathcal{H}(\mathbf{x}_n) - \mathbf{y}), \quad (2.9)$$

where  $C_{xz}$  is approximated with the ensemble, based on Equation (2.8) as

$$C_{xz} \approx \frac{1}{N} \Delta_x \Delta_z^\top. \quad (2.10)$$

### 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 [26]. The ill-posedness of the inverse problem in Equation (2.1) 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 (2.1) now has gradient, with respect to the new state  $\beta$ , given as

$$\begin{aligned} \nabla_\beta J(x(\beta)) &= \Delta_x^\top (\mathcal{H}'(x))^\top C_y^{-1} (\mathcal{H}(x(\beta)) - y) \\ &\approx \Delta_x^\top (\Delta_z \Delta_x^\top)^\top C_y^{-1} (\mathcal{H}(x(\beta)) - y) \\ &= \Delta_z^\top C_y^{-1} (\mathcal{H}(x(\beta)) - y), \end{aligned} \quad (2.11)$$

where the model discrepancies  $\Delta_z$  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

$$\beta_i = \frac{\langle x - \bar{x}, \Delta_x^{(i)} \rangle_W}{\langle \Delta_x^{(i)}, \Delta_x^{(i)} \rangle_W}, \quad (2.12)$$

or

$$\beta = (\Delta_x^\top W \Delta_x)^{-1} \Delta_x^\top W (x - \bar{x}), \quad (2.13)$$

where  $W$  is the weight matrix in the definition of the vector inner product  $\langle \cdot \rangle$ . E.g. if  $x$  is a discretized field,  $W$  is the diagonal matrix containing cell volumes. The gradient is then

$$\nabla \beta(x) = (\Delta_x^\top W \Delta_x)^{-1} \Delta_x^\top W. \quad (2.14)$$

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

$$\beta = (\Delta_x^\top W \Delta_x + \lambda I)^{-1} \Delta_x^\top W (x - \bar{x}) \quad (2.15)$$

and consequently

$$\nabla \beta(x) = (\Delta_x^\top W \Delta_x + \lambda I)^{-1} \Delta_x^\top W, \quad (2.16)$$

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.

The desired gradient can be obtained by using the chain rule to combine Equation (2.11) and Equation (2.16) as

$$\nabla J(x) = \left( (\Delta_x^\top W \Delta_x + \lambda I)^{-1} \Delta_x^\top W \right)^\top \left( \Delta_z^\top C_y^{-1} (\mathcal{H}(x) - y) \right), \quad (2.17)$$

where  $\Delta_z$  and  $\Delta_x$  are both obtained from the ensemble. The ensemble of mean-subtracted states  $\Delta_x$  could be fixed throughout the optimization, and only the ensemble of model outputs  $\Delta_z$  be updated using the fixed  $\Delta_x$  but updated mean  $\bar{x} = x_n$ . Here, however, we follow the common practice in EnVar [27] and resample the ensemble state at each iteration. Similarly, we use the true mean  $\bar{x} = x_n$  and  $\overline{\mathcal{H}(x)} = \mathcal{H}(\bar{x})$  rather than the ensemble mean for the ensemble discrepancies and denote these true means by  $\tilde{\Delta}_x$  and  $\tilde{\Delta}_{\mathcal{H}}$ , respectively. Finally the gradient is given by

$$\nabla J(x) = \left( (\tilde{\Delta}_x^\top W \tilde{\Delta}_x + \lambda I)^{-1} \tilde{\Delta}_x^\top W \right)^\top \left( \tilde{\Delta}_z^\top C_y^{-1} (\mathcal{H}(x) - y) \right). \quad (2.18)$$

## 2.4 Ensemble gradient for the RANS equations

For the present case of training a turbulence model from indirect observations, the cost function is given by Equation (2.1) with the state consisting of the discretized values of the Reynolds stress  $x = \tau$  at each cell, and the forward model  $\mathcal{H}$  consisting of 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, i.e. obtaining  $\mathcal{H}(\tau)$ , as well as creating an ensemble of Reynolds stress fields  $\Delta_\tau$  and solving the RANS equations for each sample in the ensemble to obtain the ensemble of model predictions  $\Delta_z$ . 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  $D_V$  with cell volumes along the diagonal.

method	$\partial J / \partial \tau$
direct	$(\Delta_z \Delta_\tau^\top)^\top C_y^{-1} (\mathcal{H}(\tau) - y)$
preconditioned	$\frac{1}{N} \Delta_\tau \Delta_z^\top C_y^{-1} (\mathcal{H}(\tau) - y)$
projection	$\left( (\tilde{\Delta}_\tau^\top D_V \tilde{\Delta}_\tau + \lambda I)^{-1} \tilde{\Delta}_\tau^\top D_V \right)^\top \left( \tilde{\Delta}_z^\top C_y^{-1} (\mathcal{H}(\tau) - y) \right)$

Table 1: Ensemble gradient approximations of the sensitivities of the cost function to the Reynolds stress using 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 [28]. The formulation of the Reynolds stress in Equation (1.2), however, provides a convenient way of creating the ensemble by perturbing the predicted coefficients  $g^{(i)}$ . This is done by sampling the coefficients  $g^{(i)}$  from a Gaussian process and using Equations (1.2) and (1.3) to reconstruct the Reynolds stress. One challenge, however, is that a covariance matrix must be specified for each coefficient function. Simply using the same covariance for all functions is not adequate because of the large difference in magnitude between them. The order of magnitude of the coefficients are not known a priori and can change during the optimization process. To address this challenge, the samples are created from Gaussian processes  $g^{(i),(j)} \sim GP(g^{(i)}, C_g^{(i)})$ , where each covariance kernel  $C_g^{(i)}$  shares the same correlation kernel but has a standard deviation field proportional to the norm of  $g^{(i)}$ .



The correlation kernel  $K$  used is a square exponential, and the covariance kernel is given by

$$C_g^{(i)} = \alpha^2 \|g^{(i)}\|_{D_V}^2 K, \quad (2.19)$$

$$K(x_i, x_j) = \exp\left(-\frac{1}{2} \frac{\|x_i - x_j\|^2}{l^2}\right), \quad (2.20)$$

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 for learning turbulence models from synthetic full field velocities in two cases, similar to those in [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 both cases, a neural network with 10 hidden layers and 10 neurons per hidden layer is used. A ReLU activation is used for the hidden layers and a linear activation for the output layer. The training is done with the ADAM algorithm, a common gradient-based training algorithm for deep neural networks.

#### 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  $u_b$  and channel half height  $h$ , is 10,000, and the linear  $k$ - $\omega$  model is used for the truth. The cost function is the full field discrepancy of the velocity  $u_x$ , i.e.

$$J(\boldsymbol{\tau}) = \|u_x(\boldsymbol{\tau}) - u_x^*\|_{D_V}^2, \quad (3.1)$$

where  $u_x^*$  is the true velocity field and  $D_V$  contains the cell volumes in the diagonal. Two validation studies are presented where the different ensemble gradients are compared to the gradient from solving the adjoint equations.

First, the sensitivity of the cost function to the Reynolds stress  $\partial J / \partial \boldsymbol{\tau}$  is compared between the different methods at  $\tau = 0$ , corresponding to laminar velocity. The ensemble sensitivities are approximated with 20 samples. The results are shown in Figure 2, where 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 [29]. This is in addition to the other approximations in both the adjoint and ensemble methods.

Before using the ensemble gradient to train the neural network, which has 1021 parameters, it is validated using a simplified one parameter turbulence model. In this second validation

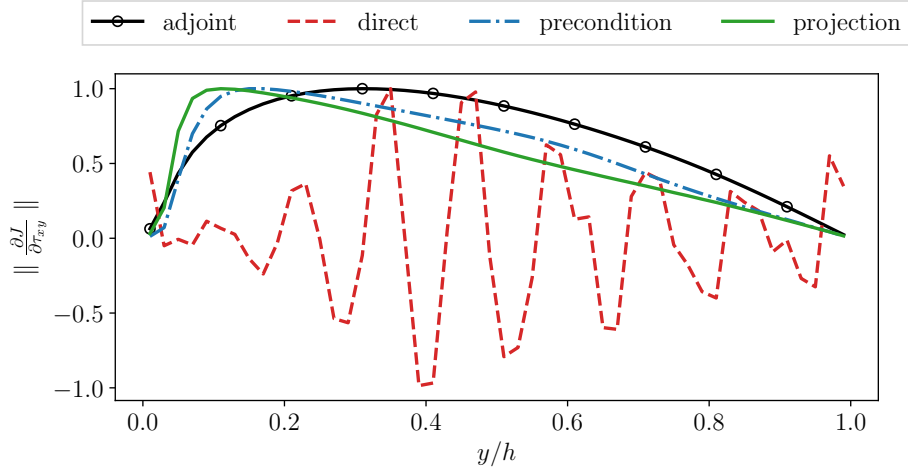


Figure 2: Comparison of sensitivity  $\frac{\partial J}{\partial \tau_{xy}}$  among adjoint method, direct ensemble method, precondition method and projection method. The sensitivity is normalized by  $\max\left(\frac{\partial J}{\partial \tau_{xy}}\right)$  to keep the maximum value as 1

test, the gradient of the cost function for the one parameter model is calculated for a range of values of the parameter and compared between the different methods. The model consists of  $g^{(1)} = -C_\mu$  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 ensemble sensitivities are approximated with 50 samples. The results are shown in Figure 3, and it can be seen that although they are noisy and have different magnitude the ensemble gradients result in the same sign and same zero as from the adjoint in the search region near the true value. The ensemble gradients 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.

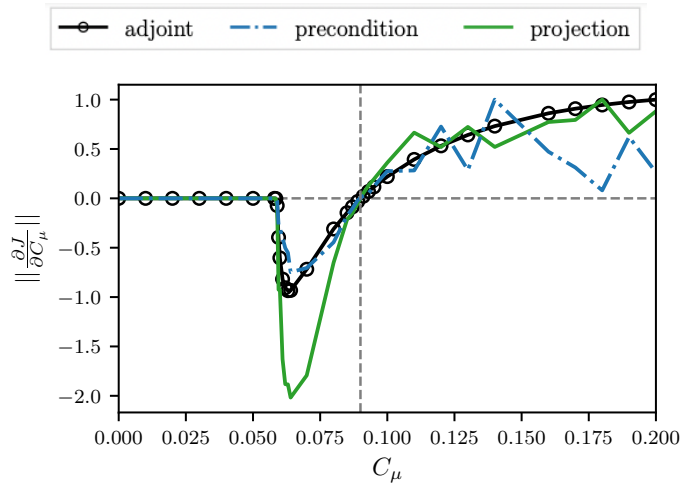


Figure 3: Comparison of the gradient of the cost function for the simplified one-parameter turbulence model using different methods. The gradients are normalized based on their maximum value. The vertical dashed line indicate the synthetic truth  $C_\mu = 0.09$ .

### 3.2 Learning a linear model

The full-field velocity in the channel case is used to learn the linear turbulence model. The neural network has one input  $\theta_1$  and one output  $g^{(1)}$  with 1,021 parameters (trainable weights). 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  $g^{(1)} = 0.0$  used in [4] since the later would result in samples with  $g^{(1)} > 0$  which are nonphysical and tend to result in diverging RANS simulations. The network is then trained using the ADAM algorithm with default parameters. For the gradient, the projection ensemble method presented in Section 2.3 is used with 20 samples. 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  $g^{(1)}$ .

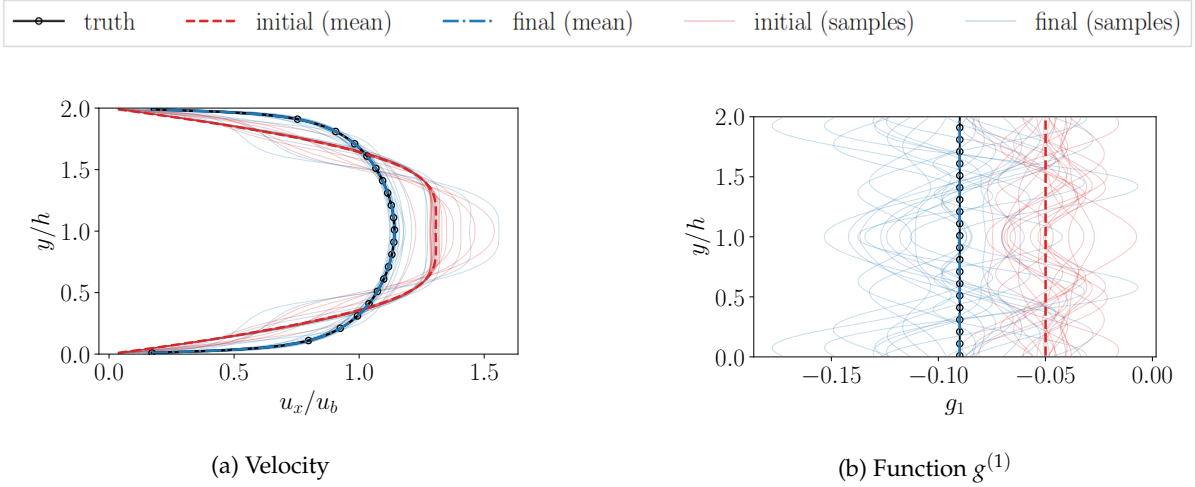


Figure 4: Results of learning a linear model from velocity data of the channel flow. The samples are used to estimate the gradient. The wall normal coordinate is indicated by  $y$  with  $y = h$  the center of the channel and  $y = 0$  and  $2h$  the bottom and upper walls, respectively.

### 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  $k-\varepsilon$  [30] given by

$$\begin{aligned}
 g^{(1)}(\theta_1, \theta_2) &= \frac{-2/3}{1.25 + \sqrt{2\theta_1} + 0.9\sqrt{-2\theta_2}}, \\
 g^{(2)}(\theta_1, \theta_2) &= \frac{7.5}{1000 + (\sqrt{2\theta_1})^3}, \\
 g^{(3)}(\theta_1, \theta_2) &= \frac{1.5}{1000 + (\sqrt{2\theta_1})^3}, \\
 g^{(4)}(\theta_1, \theta_2) &= \frac{-9.5}{1000 + (\sqrt{2\theta_1})^3}.
 \end{aligned} \tag{3.2}$$

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

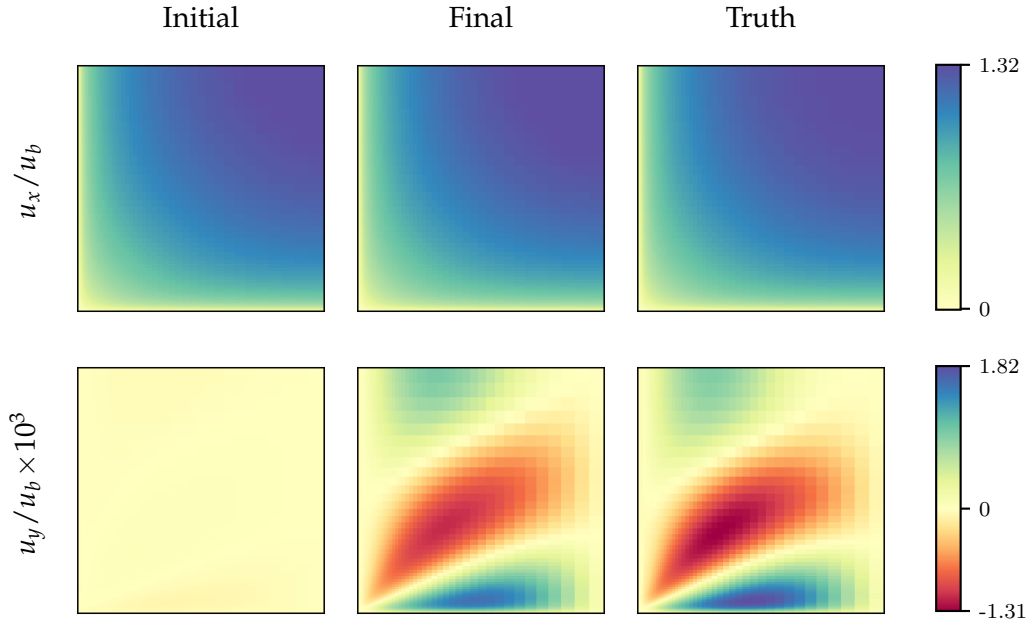


Figure 5: Velocity results of learning a NLEVM from full field velocity measurements in flow through a square duct. The axial ( $u_x$ ) and in-plane ( $u_y$ ) velocities are shown. The  $u_z$  component is the reflection of  $u_y$  along the diagonal through the center channel and is therefore not shown.

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. The gradient is obtained with the projection ensemble gradient using 20 samples. 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. It was noted in [4] 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 one to two orders 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 a 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 a 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. It is also 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.

## A Regularized projection

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

$$J_b = \frac{1}{2} \|\Delta_x \beta - (\mathbf{x} - \bar{\mathbf{x}})\|_W^2. \quad (\text{A.1})$$

We regularize the problem with Tikhonov regularization as

$$\operatorname{argmin}_{\beta} J_b(\beta) = \frac{1}{2} \|\Delta_x \beta - (\mathbf{x} - \bar{\mathbf{x}})\|_W^2 + \lambda \frac{1}{2} \|\beta\|^2, \quad (\text{A.2})$$

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

$$\nabla_{\beta} J_b = \Delta_x^{\top} W (\Delta_x \beta - (\mathbf{x} - \bar{\mathbf{x}})) + \lambda \beta. \quad (\text{A.3})$$

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

$$\beta = (\Delta_x^{\top} W \Delta_x + \lambda I)^{-1} \Delta_x^{\top} W (\mathbf{x} - \bar{\mathbf{x}}). \quad (\text{A.4})$$

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. 6 for the channel case in Section 3.2. The condition number is 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.

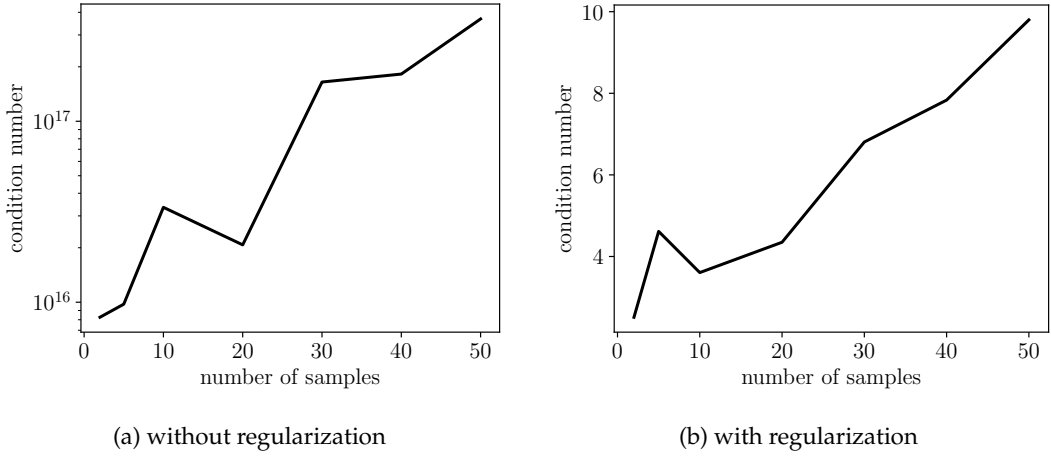


Figure 6: Condition number of the matrix  $\Delta_x^\top W \Delta_x$  for the channel case in Section 3.2 with and without regularization. The condition number is shown for ensembles with different number of samples.

## B State covariance as a preconditioner

The preconditioned gradient descent in Equation (2.6) 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 [24]. 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  $\langle a, b \rangle_p = a^\top P b$  the steepest direction is simply  $P$  times the gradient direction, as in Equation (2.6). 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

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

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 (2.6). These two cases are summarized next.

### 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 [24]. The state covariance as the preconditioner can be obtained in this manner with some approximations [31]. Using the regularized cost function in Equation (B.1), the gradient and Hessian are

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

$$\nabla^2 \tilde{J}(x_n) = C_x^{-1} + (\mathcal{H}'(x_n))^\top C_y^{-1} \mathcal{H}'(x_n) + (\mathcal{H}''(x_n))^\top C_y^{-1} (\mathcal{H}(x_n) - y), \quad (\text{B.3})$$

where here  $\nabla^2$  is the outer product ( $\nabla \otimes \nabla$ ) indicating the Hessian. For the quasi-Newton method the second derivative term  $\mathcal{H}''$  is dropped, resulting in

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

and the update scheme is

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

A crude approximation is obtained by treating the term  $G(x) = \|\mathcal{H}(x) - y\|_{C_y^{-1}}^2$  without consideration to its form. The Hessian then becomes

$$\begin{aligned} \nabla^2 \tilde{J}(x_n) &= C_x^{-1} + G'', \\ &\approx C_x^{-1}. \end{aligned} \quad (\text{B.6})$$

The update scheme becomes

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

which becomes the original update scheme preconditioned by the state covariance (Equation (2.6)), when ignoring the previous step term ( $x_n - x_{n-1}$ ).

## B.2 From ensemble Kalman inversion

The ensemble Kalman inversion (EKI) [32] is a method for general problem inversion based on iterative application of the ensemble Kalman filter (EnKF). It has recently been used for model learning [22, 33, 34] and adapted to be able to enforce arbitrary constraints [35–37]. In the EKI the state is augmented to include the observations  $\mathcal{H}(x)$  and the problem is reformulated as an artificial dynamics problem where all non-linearities 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 [38]. The EKI solves the regularized inverse problem in Equation (B.1) implicitly. The gradient-free update for each sample is given as

$$\begin{aligned} x_{n+1}^{(i)} &= x_n^{(i)} + C_{x_n z_n} (C_{z_n} + C_y)^{-1} (\mathcal{H}(x_n) - y), \\ &= x_n^{(i)} + \frac{1}{N} \Delta_{x_n} \Delta_{z_n}^\top \left( \frac{1}{N} \Delta_{z_n} \Delta_{z_n}^\top + C_y \right)^{-1} (\mathcal{H}(x_n) - y), \end{aligned} \quad (\text{B.8})$$

where  $j$  indicates the sample index and the superscript  $n$  indicates the iteration step. In the continuous time limit ( $\frac{1}{N} \rightarrow 0$ ) the evolution of a sample in the EKI is [39]

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

That is, the continuous time limit of the EKI has an update direction consistent with preconditioning the original gradient with the state covariance (Equation (2.6)).

## References

- [1] C. G. Speziale, On turbulent secondary flows in pipes of noncircular cross-section, *International Journal of Engineering Science* 20 (7) (1982) 863–872.
- [2] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, *Annual Review of Fluid Mechanics* 51 (2019) 357–377.

- [3] J. Ling, A. Kurzwaski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, *Journal of Fluid Mechanics* 807 (2016) 155–166.
- [4] C. Michelén Ströfer, H. Xiao, End-to-end differentiable learning of turbulence models from indirect observations, arXiv preprint arXiv:2104.04821 (2021).
- [5] Y. Zhao, H. D. Akolekar, J. Weatheritt, V. Michelassi, R. D. Sandberg, Rans turbulence model development using cfd-driven machine learning, *Journal of Computational Physics* 411 (2020) 109413.
- [6] S. Pope, A more general effective-viscosity hypothesis, *Journal of Fluid Mechanics* 72 (2) (1975) 331–340.
- [7] C. Audet, M. Kokkolaras, Blackbox and derivative-free optimization: theory, algorithms and applications, *Optimization and Engineering* 17 (2016) 1–2.
- [8] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature* 521 (7553) (2015) 436–444.
- [9] C. Othmer, A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows, *International journal for numerical methods in fluids* 58 (8) (2008) 861–877.
- [10] C. He, Y. Liu, L. Gan, A data assimilation model for turbulent flows using continuous adjoint formulation, *Physics of Fluids* 30 (10) (2018) 105108.
- [11] C. Sabater, S. Görtz, Gradient-based aerodynamic robust optimization using the adjoint method and gaussian processes, in: *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, Springer, 2021, pp. 211–226.
- [12] J. R. Holland, J. D. Baeder, K. Duraisamy, Towards integrated field inversion and machine learning with embedded neural networks for RANS modeling, in: *AIAA Scitech 2019 Forum*, 2019, 1884.
- [13] M. Oriani, G. Pierrot, Alternative solution algorithms for primal and adjoint incompressible navier-stokes, in: *ECCOMAS Congress*, 2016, pp. 3858–3882.
- [14] A. R. Conn, K. Scheinberg, L. N. Vicente, *Introduction to derivative-free optimization*, SIAM, 2009.
- [15] J. Nocedal, S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.
- [16] J. C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE transactions on automatic control* 37 (3) (1992) 332–341.
- [17] Y. Chen, D. S. Oliver, D. Zhang, et al., Efficient ensemble-based closed-loop production optimization, *SPE Journal* 14 (04) (2009) 634–645.
- [18] G. Li, A. C. Reynolds, Uncertainty quantification of reservoir performance predictions using a stochastic optimization algorithm, *Computational Geosciences* 15 (3) (2011) 451–462.
- [19] S. T. Do, A. C. Reynolds, Theoretical connections between optimization algorithms based on an approximate gradient, *Computational Geosciences* 17 (6) (2013) 959–973.
- [20] G. Evensen, The ensemble Kalman filter: Theoretical formulation and practical implementation, *Ocean dynamics* 53 (4) (2003) 343–367.
- [21] Y. Gu, D. S. Oliver, et al., An iterative ensemble Kalman filter for multiphase fluid flow data assimilation, *SPE Journal* 12 (04) (2007) 438–446.
- [22] N. B. Kovachki, A. M. Stuart, Ensemble Kalman inversion: a derivative-free technique for machine learning tasks, *Inverse Problems* 35 (9) (2019) 095005.
- [23] P. J. Blonigan, Q. Wang, Multiple shooting shadowing for sensitivity analysis of chaotic dynamical systems, *Journal of Computational Physics* 354 (2018) 447–475.
- [24] A. Tarantola, *Inverse problem theory and methods for model parameter estimation*, SIAM, 2005, Ch. 6.22 Gradient-based optimization algorithms, pp. 203–223.
- [25] Y. Chen, D. S. Oliver, Ensemble randomized maximum likelihood method as an iterative ensemble smoother, *Mathematical Geosciences* 44 (1) (2012) 1–26.
- [26] C. Liu, Q. Xiao, B. Wang, An ensemble-based four-dimensional variational data assimilation scheme. Part I: Technical formulation and preliminary test, *Monthly Weather Review* 136 (9) (2008) 3363–3373.
- [27] V. Mons, J.-C. Chassaing, T. Gomez, P. Sagaut, Reconstruction of unsteady viscous flows using data assimilation schemes, *Journal of Computational Physics* 316 (2016) 255–280.
- [28] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach, *Journal of Computational Physics* 324 (2016) 115–136.
- [29] C. A. Michelén Ströfer, *Machine learning and field inversion approaches to data-driven turbulence modeling*, Ph.D. thesis, Virginia Tech. <https://vtechworks.lib.vt.edu> (2021).
- [30] T.-H. Shih, J. Zhu, J. L. Lumley, A realizable Reynolds stress algebraic equation model, *Technical Memorandum 105993*, NASA (1993).



- [31] J. E. Nwaozo, Dynamic optimization of a water flood reservoir, Ph.D. thesis, University of Oklahoma (2006).
- [32] M. A. Iglesias, K. J. Law, A. M. Stuart, Ensemble Kalman methods for inverse problems, *Inverse Problems* 29 (4) (2013) 045001.
- [33] T. Schneider, A. M. Stuart, J.-L. Wu, Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data, arXiv preprint arXiv:2007.06175 (2020).
- [34] T. Schneider, A. M. Stuart, J.-L. Wu, Learning stochastic closures using ensemble Kalman inversion, arXiv preprint arXiv:2004.08376 (2020).
- [35] J. Wu, J.-X. Wang, S. C. Shadden, Adding constraints to Bayesian inverse problems, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 1666–1673.
- [36] T. Schneider, A. M. Stuart, J.-L. Wu, Imposing sparsity within ensemble Kalman inversion, arXiv preprint arXiv:2007.06175 (2020).
- [37] X.-L. Zhang, C. Michelén-Ströfer, H. Xiao, Regularized ensemble Kalman methods for inverse problems, *Journal of Computational Physics* 416 (2020) 109517.
- [38] C. Michelén Ströfer, X.-L. Zhang, H. Xiao, DAFI: An open-source framework for ensemble-based data assimilation and field inversion, *Communications in Computational Physics* 29 (5) (2021) 1583–1622.
- [39] C. Schillings, A. M. Stuart, Analysis of the ensemble Kalman filter for inverse problems, *SIAM Journal on Numerical Analysis* 55 (3) (2017) 1264–1290.