UNCERTAINTY QUANTIFICATION FOR DATA-DRIVEN TURBULENCE MODELLING WITH MONDRIAN FORESTS

A PREPRINT

Ashley D. Scillitoe\textsuperscript{1}, Pranay Seshadri\textsuperscript{1, 2}, and Mark Girolami\textsuperscript{1, 3}

\textsuperscript{1}Data-Centric Engineering, The Alan Turing Institute, London, UK
\textsuperscript{2}Department of Mathematics, Imperial College London, London, UK
\textsuperscript{3}Department of Engineering, University of Cambridge, Cambridge, UK

March 5, 2020

ABSTRACT

Data-driven turbulence modelling approaches are gaining increasing interest from the CFD community. Such approaches generally aim to improve the modelled Reynolds stresses by leveraging data from high fidelity turbulence resolving simulations. However, the introduction of a machine learning (ML) model introduces a new source of uncertainty, the ML model itself. Quantification of this uncertainty is essential since the predictive capability of data-driven models diminishes when predicting physics not seen during training. In this work, we explore the suitability of Mondrian forests (MF’s) for data-driven turbulence modelling. MF’s are claimed to possess many of the advantages of the commonly used random forest (RF) machine learning algorithm, whilst offering principled uncertainty estimates. On a manufactured test case these claims are substantiated, providing feature selection is first performed to remove irrelevant features from the training data. A data-driven turbulence modelling test case is then constructed, with a turbulence anisotropy constant derived from high fidelity data the quantity to predict. A number of flows at several Reynolds numbers are used for training and testing. Irrelevant features are not found to be a problem here. MF predictions are found to be superior to those obtained from a commonly used linear eddy viscosity model. Shapley values, borrowed from game theory, are used to interpret the MF predictions. Predictive uncertainty is found to be large in regions where the training data is not representative. Additionally, the MF predictive uncertainty is compared to the uncertainty estimated from applying jackknifing to random forest predictions, and to an a priori statistical distance measure. In both cases the MF uncertainty is found to exhibit stronger correlation with predictive errors, which indicates it is a better measure of prediction confidence. This work suggests that uncertainty quantification can be incorporated into existing data-driven turbulence modelling frameworks by replacing random forests with Mondrian forests. This would also open up the possibility of online learning, whereby new training data could be added without having to retrain the Mondrian forest.

Keywords Uncertainty quantification · Supervised machine learning · Turbulence modelling · Dataset shift · Random Forests · Mondrian Forests · Machine learning interpretability

1 Introduction

Turbulence is a key characteristic of fluid flows across many different industries. For aircraft, delaying the transition to turbulence over the wing surfaces can reduce drag \cite{1}, while large turbulent structures can increase the effectiveness of cooling systems in aircraft engines \cite{2}, both of which lead to reduced fuel consumption. For wind turbines, turbulence can increase the power output, at the expense of increased fatigue loading \cite{3}. In oil refineries, turbulence plays a key role in critical processes such as fluid catalytic cracking \cite{4}. These examples, as well as many others, indicate the importance of being able to accurately predict the effects of turbulence for a range of industrial flows.
The characterisation of turbulence is challenging due to its chaotic nature, and the broad range of spatio-temporal scales involved. The continued growth of computing power has enabled the unsteady Navier-Stokes equations to be computed directly, so that all the scales of turbulence are simulated. Unfortunately, the extreme computational cost of these direct numerical simulations (DNS) limits their application to relatively simple flows. To mitigate the high computational cost, large eddy simulation (LES) techniques only simulate the larger energy containing turbulent eddies, whilst the smaller unresolved turbulent scales are modelled. LES type approaches are gaining popularity in many industrial applications, for example for turbo-machinery [5], wind turbines [6], and urban flows [7]. But, due to their high cost, such simulations are often simplified representations of real industrial flows. As discussed by Tucker and DeBonis [2], even with the ever-increasing computing power available, LES methods are unlikely to be routinely used in the design process of complex engineering systems over the next decade and beyond.

Instead, computations based on Reynolds-averaged Navier–Stokes (RANS) models are still the workhorse for predicting turbulent flows in many industries. RANS computations are orders of magnitude cheaper than LES techniques. However, RANS models are known to perform poorly in many flows of engineering relevance, including those with swirl, pressure gradients, and streamline curvature [8][9]. Many researchers have attempted to use LES or DNS to better understand the physics of turbulence in various flows, with the aim of developing better turbulence models. As summarised in the recent review paper by Duraisamy, Iaccarino, and Xiao [10], data-driven turbulence modelling is emerging as a promising way to inform turbulence models with data in a more systematic way.

Many of the data-driven turbulence modelling strategies fall under what is known as supervised machine learning (ML). This involves learning a function based on training data consisting of a set of input-output pairs (see Sec. 2). The learned function can then be used to make predictions on the test data. Ling and Templeton [11] were one of the first to apply ML to turbulence modelling, using a random forest classifier to predict when RANS modelling assumptions would fail. Ling, Kurzawski, and Templeton [12] further used neural networks to predict Reynolds stress anisotropy. Matai et al. [13] used field inversion to determine functional discrepancies in existing RANS models, which are then reconstructed as functions of local flow features using the Adaboost ML algorithm. Wu, Xiao, and Paterson [14] investigated the use of random forests to predict the discrepancies of RANS modelled Reynolds stresses in separated flows. Kaandorp and Dwight [15] pursue similar lines, but they modify the random forest algorithm to accept a tensor basis in order to guarantee Galilean invariance.

These studies demonstrate the growing interest in applying ML techniques to turbulence modelling. However, for such methods to be employed in industry more trust is needed in the ML predictions. All ML algorithms have an inductive bias, which is the assumptions the learner uses to predict outputs given inputs that it has not encountered. The no free lunch theorem [16] states that if a bias is correct on some cases, it must be incorrect on equally many cases, thus in the context of data-driven turbulence modelling it is unlikely for one learner to be able to generalise to all unseen flows. This is evidenced in [11] and [14], where the authors report that their data-driven closures performed poorly on flows that were significantly different from the ones on which they were trained. It is desirable for the learning algorithm to be able to generalise well in situations with a significant dataset shift, where the test and training data have significantly different distributions. Yet, a more realistic and equally important requirement is that the algorithm can at least quantify the uncertainty in its own predictions. Then the user can at least know when the training data is suitable, and whether the ML predictions can be trusted. This is a pertinent issue for the field of data-driven turbulence modelling, since many of the available LES and DNS datasets are on simplified geometries at low Reynolds number [1]. But, we wish to know whether we trust ML models trained on these flows to make predictions on the more complex, higher Reynolds number flows seen in industry.

The random forest algorithm is commonly used for turbulence modelling applications [11] [14] [15] for its robustness and high predictive accuracy. Jackknifing re-sampling can be used to provide uncertainty estimates for random forest predictions [17]. However, these estimates do not correctly account for the uncertainty arising from differences between the training and test data. Ling and Templeton [11] and later Wu et al. [15] proposed the use of statistical distance metrics to measure co-variate shift between the test and training datasets. Although informative, these metrics don’t provide uncertainty estimates for the ML predictions. A number of authors have explored the use of ML techniques with inbuilt uncertainty estimates. Parish and Duraisamy [19] used Gaussian processes (GPs) to predict the turbulence intermittency and correction terms for the turbulence transport equations. To predict the turbulence anisotropy tensor Geneva and Zabaras [20] used Bayesian neural networks (BNN), while Blauw and Dwight [21] used Bayesian additive regression trees (BART). All three approaches show promise, but also have disadvantages. GPs are known to deliver high quality uncertainty estimates, however they can be challenging to scale to large datasets since their cost scales cubically with dataset size [22]. Meanwhile, neural networks can be challenging and costly to train, and Bayesian neural networks are particularly computationally expensive [23]. For BART, Metropolis-Hastings type samplers commonly used to perform inference can struggle with large high dimensional datasets [24].

1 As noted in [9], the grid resolution required for DNS and LES increases dramatically with Reynolds number.
An alternative supervised ML technique is the Mondrian forest (MF), first conceived for classification [25], and extended to regression by Lakshminarayanan, Roy, and Teh [26]. This relatively new technique was demonstrated to be competitive in computational cost and accuracy to popular tree ensemble approaches such as random forests. Unlike random forests, the trees in a Mondrian forest are probabilistic; they use a hierarchical Gaussian prior, and the posterior parameters are efficiently computed using Gaussian belief propagation. Lakshminarayanan et al. claim that this endows MF’s with principled uncertainty estimates, more akin to those of a GP, with predictions smoothly returning to the prior and exhibiting higher uncertainty far from training data. To the authors’ knowledge the use of Mondrian forests for data-driven turbulence modelling is yet to be explored. The present paper aims to explore this, to understand whether Mondrian forests offer useful uncertainty estimates in addition to accurate predictions in data-driven turbulence frameworks. The paper is structured as follows: In Section 2 commonly used frameworks of supervised ML for turbulence modelling are introduced; the Mondrian forest algorithm is then outlined in Section 3, and compared to the popular random forest algorithm in Section 4. In Section 6 random and Mondrian forests are applied to a data-driven turbulence modelling task (see Sec. 5) to assess their performance. Finally, conclusions are made in Section 7.

2 Supervised machine learning for turbulence modelling

2.1 RANS modelling uncertainties

The difficulty of RANS modelling arises from the fundamental closure problem that is introduced when the Navier–Stokes equations are averaged with respect to time. The incompressible RANS momentum equation in the $i$th direction is

$$\langle u_i \rangle \frac{\partial \langle u_i \rangle}{\partial z_j} = \frac{\partial}{\partial z_j} \left[ -\frac{\langle p \rangle}{\rho} \delta_{ij} + \nu \left( \frac{\partial \langle u_i \rangle}{\partial z_j} + \frac{\partial \langle u_j \rangle}{\partial z_i} \right) + \tau_{ij} \right] + \langle g_i \rangle,$$

(1)

where $\langle \cdot \rangle$ indicates a time-averaged quantity and $z = (z_1, z_2, z_3)$ is the spatial coordinate vector. The RANS equations are unclosed, with the turbulent (or Reynolds) stress term, $\tau_{ij}$, unknown. It is common for the Boussinesq linear eddy-viscosity assumption to be used

$$\tau_{ij} = -\rho \langle u'_i u'_j \rangle = \nu_t \left( \frac{\partial \langle u_i \rangle}{\partial z_j} + \frac{\partial \langle u_j \rangle}{\partial z_i} \right) - \frac{2}{3} k \delta_{ij},$$

(2)

where $k = \langle u'_i u'_i \rangle / 2$ is the turbulent kinetic energy. However, the turbulent eddy viscosity $\nu_t$ is still unknown, and RANS models must be used to estimate this term. Xiao and Cinnella [27] divide the uncertainties arising from RANS modelling into two categories: (i) structural (or model-form) uncertainties, which arise from the structure of the RANS equations solved to provide $\nu_t$, or from the eddy-viscosity assumption in Equation 2, and (ii) parametric uncertainties, which are caused by a lack of generalisation of closure coefficients in the RANS model. As discussed in the following section, machine learning can be used to inform both categories of uncertainties.

2.2 Machine learning to inform RANS

This paper deals only with the general framework of supervised machine learning. In this context, define $x \in \mathbb{R}^d$ to be a vector of $d$ non-dimensional flow-features obtained at a given location in a flow-field. These flow features, described further in Section 5, are intended to uniquely describe the RANS flow-field. Let $y \in \mathbb{R}$ be a scalar-valued output quantity of interest also obtained at each location within the flow-field. Assuming a flow-field is specified by $N$ spatial locations, the following matrix-vector notation is adopted to characterise the data

$$X_N = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix}, \quad Y_N = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix},$$

(3)

where the subscript denotes the $N$ training samples. Analogously, predictions of the spatially-varying scalar quantity of interest for $M$ different feature vectors is given by

$$\hat{X}_M = \begin{bmatrix} \hat{x}_1^T \\ \vdots \\ \hat{x}_M^T \end{bmatrix}, \quad \hat{Y}_M = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_M \end{bmatrix},$$

(4)
Various authors have used the aforementioned framework to predict different turbulent quantities. Ling and Templeton [11] used a machine learning classifier, where the response \( y \) is a binary label which indicates whether the RANS model can be trusted in different flow regions. For example, since the Boussinesq linear eddy-viscosity assumption (Eq. 2) presumes a positive eddy-viscosity \( \nu_t \), the response

\[
y = \begin{cases} 
0, & \text{if } \nu_t < 0 \\
1, & \text{otherwise} 
\end{cases}
\]  

\hspace{1cm} (5)

can be used to suggest regions where Equation 2 cannot be trusted. More recently, many authors [13, 19, 12, 15, 20, 14] have used machine learning regressors to predict the Reynolds stress discrepancy \( \Delta \tau = \tau^{RANS} - \tau^{NDS} \), which can be used to correct \( \tau \) before it is injected back into a CFD solver. The raw discrepancy \( \Delta \tau \) is not Galilean invariant, so it is common to instead perform the eigenspace decomposition:

\[
\tau = 2k \left( \frac{1}{3} I + A \right) = \left( \frac{1}{3} I + VAV^T \right)
\]  

\hspace{1cm} (6)

where \( I \) is the second order identity tensor, \( A \) is the anisotropy tensor, and

\[
V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}
\]  

\hspace{1cm} (7)

are the eigenvectors and eigenvalues of \( A \), respectively, indicating its shape and orientation. The eigenvalues are sorted such that \( \lambda_1 \geq \lambda_2 \geq \lambda_3 \) and \( \lambda_1 + \lambda_2 + \lambda_3 = 0 \) as the trace of the anisotropy tensor is zero. These eigenvalues can be transformed to the Barycentric coordinates \( C_1, C_2, \) and \( C_3 \), and then to Cartesian coordinates \( \zeta \) and \( \eta \). Wu, Xiao, and Paterson [14] then use machine learning to predict the discrepancy \( \Delta \tau = (\Delta \log k, \Delta \zeta, \Delta \eta, h_1, h_2, h_3) \), where \( h_1, h_2 \) and \( h_3 \) are the first three unit quaternions used to represent the transformation from the RANS eigenvectors \( V^{RANS} \) to the target eigenvectors \( V^{NDS} \).

The prediction of \( \Delta \tau \) accounts for the overall uncertainty due to the RANS model. However, Wu et al. [28] show that the injection of Reynolds stresses directly into a CFD solver can lead to the RANS equations becoming ill-conditioned. Instead, field inversion (inference) can be used [19, 13] to find a field parameter \( \beta(z) \) which improves the RANS model’s predictions. For example, Matai et al. [13] use \( \beta(z) \) to scale the production term in the \( \omega \) equation of the \( k-\omega \) RANS model:

\[
\frac{D \omega}{Dt} = \beta(z) P(k, \omega, \langle u \rangle) - D(k, \omega, \langle u \rangle) + T(k, \omega, \langle u \rangle)
\]  

\hspace{1cm} (8)

This field parameter accounts only for the parametric uncertainties in the RANS model. The parameter is problem-specific since it is a function of the problem’s coordinates \( z \), but machine learning can then be used to infer the relationship between \( \beta(z) \) and \( x \).
3 The machine learning algorithms

This paper explores the use of Mondrian forests for data-driven turbulence modelling. Comparisons will be made to the commonly used random forest algorithm, thus both algorithms are introduced in this section.

3.1 Random forests

Random forests [29] are a popular supervised machine learning algorithm. They can often achieve good accuracy even with minimal tuning of their hyper-parameters. Random forests consist of an ensemble of decision trees, like that in Figure 2a.

3.1.1 Decision trees

Following the notation of [26], we describe a decision tree by the tuple \((T, \delta, \xi)\), where \(T\) is the tree, \(\delta_j \in \{1, \ldots, d\}\) is the split dimension and \(\xi_j \in \mathbb{R}\) is the split location for the \(j^{th}\) node in the tree. A decision tree trained on the training data \((X_N, y_N)\) is a hierarchical partitioning of the input data. At each node in the tree, the data is split in a binary fashion

\[
\begin{align*}
B_{\text{left}(j)} & := \{ x \in B_j : x_{\delta_j} \leq \xi_j \} \\
B_{\text{right}(j)} & := \{ x \in B_j : x_{\delta_j} > \xi_j \}
\end{align*}
\] (9)

where the \(j^{th}\) block of data is

\[
B_j = (l_{j1}, u_{j1}] \times \cdots \times (l_{jd}, u_{jd}],
\] (10)

with \(l_{jd}\) and \(u_{jd}\) the lower and upper bounds of the rectangular block \(B_j\) along dimension \(d\). As an example, for the decision tree fitted to the input data \(x \in [0, 1]^2\) in Figure 2a, the root node \(\varrho\) splits with \(\delta_1 = 2\) and \(\xi_1 = 0.6\), leading to its right child node \(j = \text{right}(\varrho)\) having the data block \(B_j = (0, 1] \times (0.6, 1]\).

![Decision tree](image)

Figure 2: Example of classification trees trained on data with four classes over eight data points in \([0, 1]^2\). Trees are shown on the left, and the partitions in feature space are shown on the right. Regression trees are similar, except the leaf nodes contain continuous values instead of classes.

There are many induction algorithms available to learn a decision tree structure from training data, such as the popular CART algorithm [30] for classification and regression. Generally, these algorithms learn the tree structure \(T\) and leaf node parameters \(\theta_j\) by optimising an appropriate criterion, such as mean squared error (MSE) for regression. The parameter \(\theta_j\) parametrises the conditional distribution \(p(y | x \in B_j)\), and for regression is simply the mean of the \(K_j\) number of responses \(y_{Kj}\) residing in the leaf node’s block \(B_j\):

\[
\theta_j = \frac{1}{|K_j|} \sum_{q \in K_j} y_q
\] (11)

For a given test data point \(\tilde{x}\), a prediction is made by walking through the decision tree to identify its corresponding leaf node \(j\) and then returning the parameter \(\theta_j\).
3.1.2 Forests of decision trees

Decision tree have low bias, but high variance (they over-fit to training data). Random forests [29] are ensembles of decision trees, intended to reduce variance. Let \( \varphi \) be a random forest consisting of \( L \) decision trees \( T_1, \ldots, T_L \). For the data point \( \tilde{x} \), the random forest’s prediction is an average of each tree’s prediction

\[
\varphi(\tilde{x}) = \frac{1}{L} \sum_{i=1}^{L} w_i T_i(\tilde{x})
\]

where \( w_i \) is the \( i \)-th weighting term. The individual trees in a random forest are randomised with bootstrap aggregation (bagging), where each tree is trained on a slightly different subset of the training data, and additionally the set of candidate splits within each node are randomly sub-sampled. This randomisation may slightly increase bias, but it significantly decreases over-fitting (prediction variance).

The RandomForestRegressor algorithm from [github.com/scikit-learn/scikit-learn] is used in this paper. The number of decision trees in the forest is controlled with the n_estimators hyperparameter, while the depth (complexity) of the trees is controlled by max_depth. Unless otherwise stated, all hyper-parameters are kept at their defaults (as of version 0.22.1).

3.1.3 Jackknife variance estimates

Wager, Hastie, and Efron [17] propose the use of jackknife re-sampling to provide confidence intervals for random forest predictions. The infinitesimal jackknife, provides a variance measure, i.e., \( V_{ij} = \text{var}(\varphi(\tilde{x})) \) for a random forest’s predicted response to the input \( \tilde{x} \). For a random forest \( \varphi \) trained on the data \((X_N, y_N)\), the infinitesimal jackknife variance is given by

\[
V_{ij} = \sum_{i=1}^{N} \left[ \frac{1}{L} \sum_{j=1}^{L} (|\kappa_i|_j - 1) (T_j(\tilde{x}) - \mathbb{T}(\tilde{x})) \right]
\]

where \( |\kappa_i|_j \) denotes the number of times \( \kappa_i \) appears in the \( j \)-th bootstrap sample, \( T_j(\tilde{x}) \) is the predicted response of the \( j \)-th tree, and \( \mathbb{T}(\tilde{x}) \) is the mean of \( T_j(\tilde{x}) \) over \( j = 1, \ldots, L \). The variance in (13) can be biased upwards when \( L \) is small, therefore a bias corrected version is suggested

\[
V_{ij-U} = V_{ij} - \frac{N}{L^2} \sum_{j=1}^{L} (T_j(\tilde{x}) - \mathbb{T}(\tilde{x}))^2.
\]

A Jackknife-after-bootstrap variance estimate is also defined, and Wager, Hastie, and Efron [17] suggest that the arithmetic mean of this and \( V_{ij-U} \) provide more unbiased variance estimates. However, since the infinitesimal jackknife \( V_{ij-U} \) tends to overestimate the variance, this is used alone here as a conservative estimate. The standard deviation \( \sigma = \sqrt{V_{ij-U}} \) provides confidence intervals. They measure how far the random forest prediction \( \varphi(\tilde{x}) \) – obtained by building \( M \) trees on a sample – is from its expected value, which is the the average random forest prediction obtained by building \( M \) trees across different samples. They do not provide prediction intervals, meaning they do not tell us how far the predicted value \( \varphi(\tilde{x}) \) is from the true value \( \tilde{y} \).

3.2 Mondrian forests

Lakshminarayanan et al. [25,26] introduce Mondrian forests, which are constructed much like random forests, in that they are ensembles of Mondrian trees. However, unlike random forests, Mondrian forests are probabilistic, and provide principled uncertainty estimates.\(^3\)

3.2.1 Mondrian trees

Mondrian trees are restrictions of Mondrian processes to a finite set of points. Mondrian processes are families \( \mathcal{M}_t : t \in [0, \infty) \) of random hierarchical binary partitions of \( \mathbb{R}^d \), where \( \mathcal{M}_t \) is a refinement of \( \mathcal{M}_s \) whenever \( t > s \).\(^4\) As shown in Figure [25] Mondrian trees are similar to decision trees. However, there a number of important differences:

\(^3\)Mondrian trees/forests are also capable of online learning, and the distribution of trees sampled from the online algorithm is identical to the corresponding batch counterpart. In this paper only the batch algorithm is used.

\(^4\)It is referred to as a time, but this should not be confused with a physical time related to the data, or to discrete time in an online learning setting.
(i) A Mondrian tree $T$ is a tuple $(T, \delta, \xi, \zeta)$, where $(T, \delta, \xi)$ is a decision tree. The additional parameter $\zeta = \{\zeta_j\}_{j \in T}$ specifies the split time $\zeta$ for each node $j$. Split times increase with depth of the tree, and are involved in the setting of a hierarchical prior (amongst other things).

(ii) Splits are committed only within the range of training data, hence the split represented by an internal node $j$ holds only within $B_j^\delta$ and not the whole of $B_j$. The data block $B_j^\delta$, shown for $j = \text{right}(g)$ in Figure 2b, represents the smallest rectangle enclosing the input data $X$ at node $j$:

$$B_j^\delta = (l_{jd}^\delta, u_{jd}^\delta) \times \cdots \times (l_{jd}^\delta, u_{jd}^\delta),$$

where $l_{jd}^\delta$ and $u_{jd}^\delta$ denote the lower and upper bounds of the training data points in node $j$ along dimension $d$.

(iii) Unlike the CART algorithm, which samples splits based on the MSE of the predicted response data $y$, here splits are sampled independently of $y$.

### 3.2.2 Hierarchical prior and predictive posterior

Mondrian trees are probabilistic models, which determine their responses in each leaf to be Gaussian distributed, so that every node $j \in T$ has a mean parameter $\mu_j$. A hierarchical Gaussian prior is then used for $\mu = \{\mu_j : j \in T\}$, such that

$$\mu_j | \mu_H \sim \mathcal{N}(\mu_H, \phi_{\varnothing}), \quad \mu_j | \mu_{\text{parent}(j)} \sim \mathcal{N}(\mu_{\text{parent}(j)}, \phi_j),$$

where $\varnothing$ denotes the root node, and $\phi_j = \gamma_1 \sigma(\gamma_2 \zeta) - \gamma_1 \sigma(\gamma_2 \zeta_{\text{parent}(j)})$. The sigmoid function $\sigma(t) = (1 + e^{-t})^{-1}$ encodes the prior assumption that children are expected to be more similar to their parent nodes as tree depth increases. The hyperparameters $\mu_H, \gamma_1, \gamma_2$ are set according to Appendix B in Ref. [26].

As discussed in Section 3.1.1 for a typical decision tree the predicted response $\hat{y}$ for a test point $\hat{x}$ is simply the average of the responses in $B_{\text{leaf}(\hat{x})}^\delta$. With Mondrian trees, the test point can branch off the existing tree at any point along the path from the root node to leaf($\hat{x}$). Hence, the predictive posterior over $y$ is a weighted mixture of Gaussians along the path from the root node to leaf($\hat{x}$)

$$p_T(\hat{y}|\hat{x}, (X, y)) = \sum_{j \in \text{path(leaf($\hat{x}$))}} w_j \mathcal{N}(m_j, v_j),$$

where the weight $w_j$ describes the probability of branching off just before reaching the $j^{th}$ node, and $m_j$ and $v_j$ are the predictive mean and variance at the $j^{th}$ node. As the test point $\hat{x}$ moves further away from the training data at a given node $j$, i.e. as $\hat{x}$ moves away from the block $B_j^\delta$ in Figure 2b the probability $w_j$ increases. This causes Mondrian forests to exhibit higher uncertainty as $\hat{x}$ moves further away from the training data $(X, y)$. Additionally, the Mondrian forest’s predicted response $\hat{y}$ approaches the prior as $\hat{x}$ moves further away from the training data. To obtain the predictive mean and variance at each node, we require $p_T(\mu | (X, y))$, the posterior over $\mu$. This is computed using Gaussian belief propagation. Lakshminarayanan, Roy, and Teh [25] note that, since a hierarchical tree structure is used, posterior inference can be performed with a computational cost of $O(N)$ for $N$ training samples. This is compared to a Gaussian process whose computational cost is typically $O(N^3)$.

### 3.2.3 Forests of Mondrian trees

Lakshminarayanan, Roy, and Teh [25] propose combining Mondrian trees to form a Mondrian forest as a way to reduce over-fitting behaviour. The prediction of a Mondrian forest is then the average prediction from the $L$ number of Mondrian trees

$$p(\hat{y}|\hat{x}, (X, y)) = \frac{1}{L} \sum_{j} p_j(\hat{y}|\hat{x}, (X, y)).$$

Again, this predictive posterior over $\hat{y}$ is a mixture of Gaussians, and it is straightforward to calculate the predictive mean and variance from this. For further details see Section 3 in [26].
Following the approach of other practitioners, such as Mourtada, Gaïffas, and Scornet [31], bootstrap aggregation is not used when training Mondrian forests in this paper. Due to the randomisation involved in the construction of Mondrian trees, each tree is still likely to be different. The MondrianForestRegressor algorithm from github.com/scikit-garden/scikit-garden is used in this paper. Lakshminarayanan, Roy, and Teh [26] control the depth of their Mondrian trees by stopping the splitting of nodes which have less than min_samples_split number of data points. To provide a fairer comparison with the random forest algorithm, in this paper we set min_samples_split=2 and additionally limit the depth of Mondrian trees with the max_depth hyperparameter.

4 Manufactured test case

Before applying the machine learning algorithms introduced in Section 3 to a real turbulence modelling application, a manufactured test case will be explored to better understand the performance of the two algorithms. The trigonometric function

\[ y = x_2 \sin(10x_1) - x_1 \cos(10x_2), \quad \text{where} \quad x_1 \in [0, 2], \ x_2 \in [0, 1], \]  

shown by the grey surface in Figure 3 is sampled \( N = 700 \) times (the black dots) to generate a training data set \((\mathbf{X}, \mathbf{y})\), where \( \mathbf{X} \in \mathbb{R}^{N \times 2} \) and \( \mathbf{y} \in \mathbb{R}^N \). Note that all the input training data falls within \( x_1 \subset [0, 1] \) and \( x_2 \subset [0, 1] \). A random noise term given by \( \epsilon \sim \mathcal{N}(0, 0.05) \) is added to \( \mathbf{y} \) to represent measurement noise. Equation 19 is two dimensional, but the input space is extended to \( d = 2 + d_{irrel} \) dimensions, where the additional dimensions \( x_3, \ldots, x_d \) are irrelevant features sampled from a uniform distribution \( U[0, 1) \). These irrelevant features are non-zero features, or elements, in each \( \mathbf{x} \) that do not affect the response \( y \).

Figure 3: The test function given in Eq. 19 with the black circles representing the training data.

To quantify predictive accuracy, we generate a testing data set with \( M = 300 \) data points \((\tilde{\mathbf{X}}, \tilde{\mathbf{y}})\). Both random forests (RF) and Mondrian forests (MF) are trained on \((\mathbf{X}, \mathbf{y})\) and the \( \overline{R^2} \) scores (Eqn. 25) between the predicted responses \( f_{ML}(\tilde{\mathbf{X}}) \) and true responses \( \tilde{\mathbf{y}} \) are calculated. For both algorithms, fully grown trees (max_depth=None) are used. Training and testing is repeated five times for a range of \( d_{irrel} \) values, and the averaged \( \overline{R^2} \) scores are plotted in Figure 4a. At each node in a random forest a random \( k \)-dimensional subset of the original \( d \) features is sampled, and splits are made based on the most important variables. Figure 4a indicates that this implicit feature selection helps make the random forest algorithm relatively insensitive to irrelevant features. On the other hand, as noted in point (iii) in Section 3.2.1, Mondrian forests choose splits independent of the response data \( y \). This means Mondrian forests can be sensitive to irrelevant features, as is the case in Figure 4a. To address this issue, Lakshminarayanan, Roy, and Teh [26] suggest applying manual feature selection to down weight or filter irrelevant features prior to training.

As discussed in Section 3.1.3, the infinitesimal jackknife variance estimates can be biased when the number of trees \( L \) in the random forest is small. The standard deviation arising from the bias corrected variance, \( \sigma = \sqrt{V_{IJ} - L} \), is plotted against \( L \) in Figure 4b. Wager, Hastie, and Efron [17] suggest that \( L = O(N) \) is necessary for converged variance estimates. Figure 4b shows that \( \sigma \) has started to level off at \( L = 1000 \) (\( N = 1000 \) in this case), however convergence still has not been reached at \( L = 5000 \) trees. In comparison, the Mondrian forest predicted \( \sigma \) converges around \( L = 50 \) in Figure 4b. Since forests with larger numbers of trees are more costly to train and save, requiring less trees for converged variance estimates is a potential advantage of Mondrian forests.
As an important aspect of this paper is the study of uncertainty from unrepresentative training data; we desire methods which yield a high uncertainty as the test data moves away from the training data. To explore this, the trained random forest (with $L = 800$ trees) and Mondrian forest (with $L = 50$ trees) are used to make predictions over the entire domain in (19). Three new training points are also added near $x = (1.5, 0.25)$. The resulting predictive mean and standard deviations are plotted in Figure 5. The random forest’s predicted response (Fig. 5a) exhibits a strong wavy pattern in the $x_2$ direction between $1 < x_1 < 2$, and the small confidence intervals suggest the algorithm is confident in this prediction. This is despite there being almost no training data available in this region for the random forest to base its prediction upon. In comparison, the Mondrian forest’s predictions (Fig. 5a) are more akin to what one would expect from a GP model, with the predictive mean approaching the prior ($\mu_H = 0.13$ in this case) and large prediction intervals in the region $1 < x_1 < 2$, where there is a lack of training data.

A one dimensional slice of the function $f(x_1, x_2)$ and the algorithms’ predictions are plotted in Figure 6. The previously mentioned behaviour can be seen more clearly in Figure 6a, with the Mondrian forest correctly having less confidence in the extrapolation region $x_1 > 1$, while the random forest is strongly overconfident. If more data points were added in the region $1 < x_1 < 2$ the predicted mean from a GP would be expected to move towards these points, with the prediction intervals shrinking accordingly. Figure 6b shows that this also occurs with the Mondrian forest prediction when three training points are added near to $x = (1.5, 0.25)$. 
5 Data-driven turbulence modelling

The manufactured test case in Section 4 suggests that Mondrian forests offer principled uncertainty estimates, similar to those from GP models, while their accuracy is competitive with random forests as long as irrelevant features are not present. To explore whether these findings hold for a more challenging turbulence modelling applications, a testing framework is introduced in this section. A python package implementing the framework outlined in this section is available from github.com/ascillitoe/mondrian_turbulence.

5.1 Database of turbulent flows

Training and testing data are generated from the ten turbulent flow cases outlined in Table 1 and visualised in Figure 7. Each case is a well-validated near direct simulation (NDS), with the majority obtained from publicly accessible databases. For each NDS case a companion RANS dataset is generated using the SU2 open source CFD solver (version 6.2.0) available from github.com/su2code/SU2. For all cases the incompressible solver and SST turbulence model are used, and grid refinement is carried out to ensure mesh independence.

For cases 1 and 3 a 1D inlet velocity profile is set at the inlet to match the NDS data, and a uniform static pressure outlet boundary condition is enforced to match the area-averaged static pressure from the NDS at the same location. For Cases 2, 4 and 5 streamwise periodicity is enforced, with the streamwise forcing calibrated to match the NDS mass flow rate. All cases are assumed to be fully turbulent, with no transition model used. All RANS cases are 2D, and the NDS cases are span-wise averaged to match the RANS cases.

The RANS and NDS data is pre-processed according to the framework in Figure 1. At each mesh point in the RANS cases the mean flow data is converted to a $d$-dimensional feature vector $\mathbf{x}$ as discussed in Section 5.3. To reduce the amount of data the input data is filtered to remove non-unique data points (to the $4^\text{th}$ significant figure), and 20% of the points are randomly sub-sampled. After filtering and sub-sampling all ten cases yields a total of 25000 observations. The responses $y$ are then calculated from the NDS data, as discussed in Section 5.2.

Table 1: Summary of flow cases used for training and testing.

| Case | Description               | $Re_\tau$   | LES/DNS | Ref. |
|------|---------------------------|-------------|---------|------|
| 1    | Curved backwards step     | $Re_\tau = 618$ | LES     | [32] |
| 2    | Periodic hills            | $Re_\tau = 160$ | LES     | [33] |
| 3a/b/c | Convergent-divergent channel | $Re_\tau = 395/617/950$ | DNS     | [34], [35] |
| 4a/b | Duct with aspect ratio $AR = 1$ | $Re_\tau = 180, 360$ | DNS     | [36] |
| 4c/d | Duct with aspect ratio $AR = 3$ | $Re_\tau = 180, 360$ | DNS     | [36] |
| 5    | Ribbed channel flow       | $Re_\tau = 360$ | LES     | [37] |

*5turbmodels.larc.nasa.gov*, *turbase.cineca.it* and *flow.kth.se/flow-database*.
5.2 Predictor response

Section 2.2 reviewed some of the turbulence parameters that a machine learning algorithm can use, to predict or augment a RANS turbulence model. Predicting the Reynolds stress discrepancy $\Delta \tau$ would allow for direct augmentation of the RANS model. However, as the purpose of this paper is to explore the performance of the machine learning algorithm, we instead choose to simplify matters, and set the training response $y$ to be the turbulence anisotropy constant proposed by Banerjee et al. [38],

$$C_{aniso} = -3\lambda_3,$$

where $\lambda_3$ is calculated from the NDS Reynolds stress fields. The constant $C_{aniso}$ describes the turbulent anisotropy, with $C_{aniso} = 0$ indicating isotropic turbulence (the $3 \langle u'_i u'_i \rangle$ components are equal) and $C_{aniso} = 1$ indicating fully one component turbulence (only one non-zero $\langle u'_i u'_i \rangle$ component). If the augmentation of a RANS model was desired it would be a simple exercise to extend the present framework to predict other turbulence parameters such as the Reynolds stress discrepancy term $\Delta \tau = (\Delta \log k, \Delta \zeta, \Delta \eta, h_1, h_2, h_3)$ used in [14].

5.3 Flow features

The $d$ flow features utilized in this paper are the same as in [14]. Following [11], the raw features $\alpha$ are non-dimensionalised by a local normalisation factor $\beta$

$$\hat{\alpha} = \frac{\alpha}{|\alpha| + |\beta|},$$

where $\alpha$ and $\beta$ are given in Table [2]. The strain-rate tensor $S$ and rotation-rate tensor $\Omega$ are included as it has been recognised that all algebraic Reynolds stress and eddy viscosity models can be written in the general form $\tau = f(S, \Omega)$ [39]. The pressure gradient is included since it is known that turbulence is also influenced by this, while the turbulent kinetic energy (TKE) gradient is added in an attempt to account for non-local turbulent transport effects. Finally, near wall viscous effects have an important influence on turbulence so a non-dimensional wall distance feature is included, along with features to describe the length and time-scales of turbulence.

To achieve Galilean and rotational invariance of the machine learning function, the approach of [14] is adopted. The normalised gradients $\nabla \hat{p}$ and $\nabla \hat{k}$ are transformed into the anti-symmetric tensors $\hat{A}_p$ and $\hat{A}_k$, and the minimal integrity bases for the tensorial set $\{ \hat{S}, \hat{\Omega}, \hat{A}_p, \hat{A}_k \}$ are calculated. Each of the 47 invariant bases is the trace of a tensor such
Table 2: Non-dimensional features used to represent the flow fields. Notations are as follows: $u$ is the mean velocity vector, $k = \frac{u_i u_i}{2}$ is the turbulence kinetic energy (TKE), $\omega$ is the specific turbulent dissipation rate, $| \cdot |$ and $|| \cdot ||$ indicate vector and matrix norm.

| Description                  | Raw input, $\alpha$ | Normalisation, $\beta$ |
|------------------------------|---------------------|------------------------|
| Strain-rate tensor           | $S^2$               | $\omega$              |
| Rotation-rate tensor         | $\Omega$            | $||\Omega||$           |
| Pressure gradient            | $\nabla p$           | $\rho |Du/Dt|$          |
| TKE gradient                 | $\nabla k$           | $\omega \sqrt{k}$     |
| Wall-distance based Reynolds number, $Re_d$ | $\min \left( \frac{\tau_d}{\nu_2}, 2 \right)$ | - |
| Turbulence intensity, $Ti$   | $k$                 | $\frac{3}{2} |u|^2$   |
| Ratio of turbulent to mean time-scales, $t_1/t_s$ | $\omega$ | $\frac{1}{||S||}$ |

as $S^2$ and $\Omega \hat{A}_p \hat{A}_k \hat{S}$. Combining the remaining 3 scalar features in Table 2 with the invariant bases leads to a 50 dimensional feature space $x \in \mathbb{R}^{50}$. For further details see Appendix B in Ref. [14].

Remark 1 By learning a function mapping between $y$ and $x$, we have assumed the Reynolds stress $\tau$ can be described by purely local (point-wise) quantities $(k, \omega, \nu)$, and the local gradients $(\nabla u, \nabla k, \nabla p)$ at a given point. This assumption is likely to be questionable in regions where the turbulence scales are large or transport of turbulence is strong. Additionally, since $y$ is derived from NDS data and $x$ from RANS data, we have implicitly assumed the RANS and NDS mean velocity and pressure fields match. This assumption is likely to be invalid if more complex flow cases were used for training.

To check for the effect of irrelevant features, backward elimination was performed, with prediction accuracy monitored as the least important features were iteratively removed. Surprisingly, irrelevant features were not found to be an issue in this case, perhaps because all non-zero features are important and many of the 47 invariant terms are zero for two-dimensional flows. Nevertheless, to reduce training/prediction times and data storage requirements, a 12 dimensional subset (shown in Eqn. 28) of the original 50 features is used for both RF’s and MF’s. These 12 features were selected from a strictly computational perspective and based on iterating with different features and comparing their $R^2$ values; for further details see Sec. A.2. This pruning down of features also constitutes a departure from the methodology pursued in [14].

6 Results

The Mondrian forest ML framework introduced in Section 5 is now tested on a number of the flow cases. Random forest predictions are not explored here due to the lack of convergence of their uncertainty estimates (see Sec. A.3). Note that generally, the MF predictions were found to have competitive accuracy compared to their RF counterparts (see Sec. A.2).

6.1 Convergent-divergent channel

The turbulence anisotropy constant $C_{aniso}$ obtained from the DNS data [34] for the convergent-divergent channel at $Re_x = 617$ is shown in Figure 8. As expected, the turbulence anisotropy is generally high near the upper and lower viscous walls. Near the lower wall, downstream of the bump ($5.5 < z_1 < 7$), a strong adverse pressure gradient leads to flow separation and a small recirculation region. The production of $\langle u'_i u'_j \rangle$ is an order of magnitude larger than $\langle u'_i u'_2 \rangle$ in this region [34], leading to highly anisotropic turbulence. Near the upper surface at the same horizontal location, a weaker adverse pressure gradient is observed, implying that the flow is on the verge of separation, and turbulence anisotropy is also high. The turbulence returns towards isotropy in the center of the channel around $z_1 = 5.5$ due to the favourable pressure gradient (leading to flow acceleration) in this region.

6.1.1 Initial Mondrian forest predictions

In Figure 9, the Mondrian forest predictions for $C_{aniso}$ and the truth (DNS) are plotted across the $z_1$-planes shown in Figure 8. When judging the Mondrian forest predictions here it is important to note that the Mondrian forest has not seen this flow case during training, so it is making predictions based on the physics it has learnt from the other flow cases. In Figure 9a, the Mondrian forest is trained only on the curved backward step (case 1, $Re_x = 618$) and periodic hills (case 2, $Re_x = 160$). Hence, a degree of extrapolation in feature space is likely here. Generally, the MF
Figure 8: Contours of the true turbulent anisotropy constant $C_{aniso}$ for the convergent-divergent channel at $Re_\tau = 617$ (case 3b). The inflow is located at $z_1 = 0$, and $z$ is non-dimensionlised by the channel half-height.

is predicting the physics seen in the DNS well, with high anisotropy predicted in the near-wall regions, and a further increase in anisotropy in the adverse pressure gradient, downstream of the bump (slices 3 and 4). Where predictions are less accurate, for example the return to isotropy region in the centre of the channel at slices 2-4, the Mondrian forest returns large predictive uncertainty values. Generally, the MF’s predictions are more accurate than the $C_{aniso}$ predicted by the linear eddy viscosity model (LEVM) in Equation 2, suggesting MFs offer the potential to augment LEVM based RANS models.

Remark 2 The uncertainty bounds given by $\mu \pm \sigma$ do not entirely encompass the truth data here. However, this is expected, since we are plotting $\pm 1\sigma$, so would only expect $\approx 68\%$ of truth values to fall within these uncertainty bounds. Additionally, Lakshminarayanan, Roy, and Teh [26] examine the probability calibration measures for MF vs RF, and although MFs perform significantly better than RFs with jacknifing, they still tend to be slightly over-confident with their uncertainty estimates—an artifact that could perhaps be fixed with calibration of their uncertainty estimates. That said, it should be noted these uncertainty estimates do not account for uncertainty due to the assumptions discussed in Remark 1.

6.1.2 More representative training data

In Figure 9b all cases except the present test case (case 3b) are used for training. The effect of this additional training data is to improve the accuracy of the MF’s mean predictions, while reducing its predictive uncertainty. This is especially noticeable in the return to isotropy region near the channel centre at slices 2-4. To understand why, the MF’s predictions are interpreted using SHapley Additive exPlanation (SHAP) values, recently proposed by Lundberg and Lee [40]. This is an additive feature attribution method based on Shapley values from coalitional game theory. They are defined as Shapley values of a conditional expectation function of the original model [40]; effectively the $j$-th SHAP value $\phi_j$ represents the contribution to the prediction $\tilde{y}$, compared to the average prediction for the dataset, when conditioning on the $j$-th feature. Thus, the sum of the SHAP values satisfies

$$\sum_{j=1}^{D} \phi_j(\tilde{y}) = \tilde{y} - \mathbb{E}(y).$$

(22)

To efficiently compute exact SHAP values for tree-based machine learning models Lundberg et al. [41] propose TreeSHAP, available from [github.com/slundberg/shap]. This is used to compute SHAP values for predictions at the two locations marked by the circles in Figure 9. The seven SHAP values with greatest magnitude are plotted in Figure 10. Near the wall (Fig. 10a), the relatively small turbulent to strain timescale ratio ($t_t/t_s = 0.16$) decreases the predicted $C_{aniso}$ value compared to the average, whereas the axi-symmetric pressure gradient tensor value, $A_p^2 = -0.092$, increases the predicted turbulent anisotropy. However, the sum of the SHAP values is relatively small here ($\sum \phi_i = 0.03$), meaning the predicted anisotropy is close to the average value ($\tilde{y} = 0.71$). In the center of the channel (Fig. 10b), the predicted anisotropy is considerably lower than the average mainly due to the high $Re_d$, but the other six features shown also contribute to the $C_{aniso}$ constant tending towards isotropy here.

In Figure 11 Kernel Density Estimates (KDE) show the distribution of training data in feature space, for the two training data sets used for the predictions in Figure 9. The six features identified as important for the two locations in question

6The code is slightly modified to be compatible with the scikit-garden MondrianForestRegressor, and made available at github.com/ascillitoe/shap
Figure 9: Mondrian forest predicted turbulent anisotropy constant $C_{\text{aniso}}$ for the convergent-divergent channel at $Re_\tau = 617$ (case 3b), at the slices marked in Figure 8.

(a) Near lower wall, $z_2 = 0.72$

(b) Mid-channel, $z_2 = 1.35$

Figure 10: The seven largest SHAP values at two different $z_2$ locations across $z_1 = 5.5$, for the convergent-divergent channel at $Re_\tau = 617$ (case 3b). Computed using the Mondrian forest trained on all cases (except for case 3b).

are shown, with $Re_d$ ignored since $Re_d = 2$ for all mid-channel locations. The near-wall test point ($z_2 = 0.72$, green circle) lies within both of the training data distributions. However, the mid-channel test point ($z_2 = 1.35$, grey circle) lies outside the first training data distribution (red contours), indicating that this training data isn’t representative of the test point. The second training data set includes a training case (case 3c) with the same geometry as the test case, and the blue contours show that this training data better represents the mid-channel test point. The reduced extrapolation in feature space (for the six locally important features) explains why the predictive error and uncertainty are reduced in the return to isotropy region (fig 9b) when more training data is added.
Figure 11: Bivariate kernel density estimation plots showing distributions of the two training data sets used for the convergent-divergent channel test case at \( Re_\tau = 617 \) (case 3b). The six most important features (after \( Re_d \)) are shown. The circular markers indicate the feature values at the two points plotted in Figure 10. Density is normalised by the maximum density of each KDE.

### 6.2 Periodic hill

The second case considered is the periodic hill (case 2), for which contours of \( C_{aniso} \) from the DNS [33] are shown in Figure 12. The flow physics here is similar to the previous case, with high turbulence anisotropy observed in the shear layer where the flow separates at the hill crest (\( z_1 < 1.5 \)). After the flow reattaches to the lower surface around \( z_1 = 2 \) the turbulence anisotropy decreases near the wall. When the windward side of the next hill is reached around \( z_1 = 7.5 \), the spanwise turbulent stress \( \langle u'_3 u'_3 \rangle \) increases significantly compared to the \( \langle u'_1 u'_1 \rangle \) and \( \langle u'_2 u'_2 \rangle \) stresses near the wall, leading to high \( C_{aniso} \) values here. This effect is caused by pressure–strain interactions in this region [33], and is interesting as many pressure-strain models assume that pressure–strain interactions isotropise the normal Reynolds stresses.

Figure 12: Contours of the true turbulent anisotropy constant \( C_{aniso} \) for the periodic hill (case 2). The stream-wise periodic boundaries are located at \( z_1 = 0 \) and \( z_1 = 9 \), with \( z \) non-dimensionlised by the hill height.

In Figure 13, the Mondrian forest predictions for \( C_{aniso} \) and the truth are plotted across the \( z_1 \)-planes shown in Figure 12. In Figure 13a, the Mondrian forest is trained only on the four duct flow cases (cases 4a/b/c/d). No pressure induced separation is present in the duct flows, hence these predictions represent a considerable extrapolation. The anisotropy close to the lower wall is well captured, with small uncertainty bounds. However, away from the wall predictive errors are larger and the MF is less confident in its predictions. When the other flow cases (except for case 2) are added to the training data (Fig. 13b) the predictions are significantly more accurate, and the uncertainty bounds are much smaller. Again, the MF achieves significantly better accuracy compared to the LEVM’s predictions.
6.3 Square duct

As a final test case, predictions are made for the square duct ($AR = 1$) test case with $Re_\tau = 360$ (case 4b). From comparing contours of the true (Fig. 14a) and predicted (Fig. 14b) anisotropy constant it is apparent that the MF’s predictions are generally reasonable. There are errors of up to $C_{aniso} = 0.14$ (Fig. 14c). However, these errors mainly occur in regions where the predictive uncertainty (Fig. 14d) is high, so at least a user would know that predictions in these regions are less trustworthy.

Figure 14: Contours of turbulent anisotropy constant $C_{aniso}$, Mondrian forest prediction error and $\sigma$, for the square duct at $Re_\tau = 360$ (case 4b). Coordinates are scaled by the duct half-height. All flow cases (except 4b) are used for training.
For a more quantitative examination of accuracy the predicted and true $C_{\text{aniso}}$ are plotted along two slices in Figure 15. Across $z_2 = 0$ (Fig. 15a), a MF trained only on the duct cases at $Re_\tau = 180$ (cases 4a/c, blue line) provides generally good accuracy, albeit with a noticeable error at $z_2 \approx 0.2$. This prediction represents an extrapolation in Reynolds number, from $Re_\tau = 180$ to $Re_\tau = 360$, and adding $Re_\tau = 360$ data to the training set (case 4d) does slightly reduce the error at $z_2 \approx 0.2$ (see green dashed line in Fig. 15a). Adding more training data (orange dotted line in Fig. 15b) reduces the predictive error further, however the predictive uncertainty is actually increased here. Furthermore, across the diagonal slice (Fig. 15c), adding more training data leads to slightly increased predictive errors and uncertainty. This highlights the fact that adding more training data will not always improve predictions, and the new training data must be representative of the test flow.

6.4 Calibration of predictive uncertainty

In the preceding sections, the MF predictive mean $\mu$ and variance $\sigma^2$ are used to produce prediction intervals where, for example, the 68.27% interval says we expect 68.27% of the true test values to lie within the interval $\mu \pm 1.0\sigma$. For these prediction intervals to be trusted, it is important to determine whether they are well calibrated. Following Lakshminarayanan et al. [26], probability calibration curves are plotted in Figure 16. For each $p\%$ (e.g. 90%), the prediction interval $\mu \pm q_p\sigma$ is calculated assuming Gaussian quantiles

$$q_p = \sqrt{2 \text{ erfc}^{-1}(p)}.$$  

The percentage $M\%$ of true test points $\tilde{y}$ that lie within each prediction interval $\mu \pm q_p\sigma$ is then measured. If the model is perfectly calibrated, $M\%$ would equal $p\%$ for all $p$. The MF calibration curves for cases 2 and 3b are plotted in Figure 16 and the RF jackknife uncertainty estimates are included for comparison. For both cases, the MF uncertainty estimates are well calibrated, with the MF calibration curves lying close to the ideal dashed line. This means that for a given $p\%$ interval, approximately $p\%$ of the test predictions are expected to lie within the interval. On the other hand, the RF jackknife uncertainty estimates are poorly calibrated, with the confidence intervals displaying significant under-confidence. This suggests the random forest confidence intervals are not suitable to be used as prediction intervals.

6.5 Comparison to distance based prediction confidence

A number of other approaches have been suggested to supplement data-driven turbulence modelling frameworks with measures of prediction confidence. Ling and Templeton [11] suggest using the Mahalanobis distance, while Wu et al. [18] suggest a measure based on KDE. Both are statistical distance measures, which are used to measure the distance between a test point and a distribution of training data. The Mahalanobis distance is defined as the distance between a point $x$ in feature space and the mean of the training points $\mu$,

$$D_m = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)},$$  

7The jackknife estimate returns confidence intervals not prediction intervals for the random forest (see Sec. 3.1.3).
Uncertainty Quantification for Data-driven Turbulence Modelling with Mondrian Forests

![Probability Calibration Curves](image)

Figure 16: Probability calibration curves for random forest and Mondrian forest predictions. The percentage $M\%$ of test data points $\tilde{y}$ that lie within the interval $\mu \pm q_p\sigma$ is measured for each value of $p\%$.

where $\Sigma$ is the covariance matrix of the training points. The normalised Mahalanobis distance for the point $x$ is then defined as $\hat{D}_m = 1 - \gamma D_m$, where $\gamma D_m$ is the fraction of training points with a larger raw Mahalanobis distance than the point $x$. It follows that a prediction at a point where $\hat{D}_m = 0$ involves no extrapolation, whereas $\hat{D}_m = 1$ involves very high extrapolation.

To assess how well $\hat{D}_m$ performs as a measure of prediction confidence, violin plots are used to inspect the correlation between $\mathit{C_{aniso}}$ prediction error and $\hat{D}_m$. The normalised Mahalanobis distance $\hat{D}_m$ is discretised into ten bins, and a kernel density plot is generated for each bin. Figure 17 consists of violin plots of a random forest $\mathit{C_{aniso}}$ predictions for the convergent-divergent channel at $Re_\tau = 617$ (case 3b). Comparing Figures 17a and 17b it is apparent that when there is more extrapolation (Fig. 17b) a greater proportion of the test points have high predictive errors and large $\hat{D}_m$ values.

Within the test data there is some correlation between error and $\hat{D}_m$. However, there are many data points where $\hat{D}_m$ is large while the error is small.

In Figure 18 violin plots for RF and MF predictive uncertainties are plotted for the same test case (case 3b). Generally, the correlation between prediction error and $\sigma$ is far stronger than for error and $\hat{D}_m$. For the RF (Figs. 18a and 18b) the correlation is generally strong, but there are some trends that suggest the the RF $\sigma$ may not provide a good prediction confidence measure. Firstly, for the low extrapolation case (Fig. 18a), $\sigma > 0.175$ even for data points with no error. This is not a problem in itself, but suggests the RF jackknife estimates could be underconfident. Although a degree of conservatism may be desired, too much (i.e. if $\sigma$ is always high) might result in the uncertainty estimates being ignored.

---

8Scott’s rule is used to estimate kernel bandwidth.

9For the RF in this section, $L = 1280$ is chosen with no maximum depth to offer slightly more converged $\sigma$ estimates.
Secondly, for the high extrapolation case (Fig. 18b), the $\sigma = 0.1$ bin has very high error, despite the low uncertainty here. The MF predictive uncertainties (Figs. 18c and 18d) do not exhibit these issues, and generally the correlation between error and $\sigma$ is strong. However the MF is not infallible, and there are a small number of outliers where the MF is overconfident, with small $\sigma$ yet high error.

As a final comparison, the same violin plots are repeated for the square duct at $Re_\tau = 360$ (case 4b). Similar trends are observed: in Figure 19a, the correlation between error and $\hat{D}_m$ is limited; the RF uncertainties (Fig. 19b) show more correlation, but $\sigma > 0.09$ even where errors are very small which suggests underconfidence; finally, the MF uncertainties (Fig. 19b) show good correlation with error, and the spread of errors across each $\sigma$ bin is small.

7 Conclusions

For the emerging field of data-driven turbulence modelling to enter mainstream use, methods to quantify the uncertainties arising from such approaches will be necessary. To this end, the present work explores the use of a recently proposed machine learning algorithm, Mondrian regression forests, for data-driven turbulence modelling. On a manufactured test case (Sec. 4), Mondrian forests were shown to offer principled uncertainty estimates, similar to those from GP models, while their accuracy was competitive with the popular random forest algorithm as long as irrelevant features were not present.

The use of Mondrian forests for data-driven turbulence modelling was then explored with the framework outlined in Section 5. Mondrian forests were found to offer comparable accuracy to random forests when predicting the turbulence anisotropy constant derived from near direct simulations (NDS). Interestingly, feature selection indicated that irrelevant
features did not degrade the Mondrian forests’ accuracy in this case. However, this may be specific to the flow cases and feature set used in this paper, and it would be prudent to perform a similar feature selection procedure on any new training data. Furthermore, hyperparameter tuning demonstrated that the Mondrian forests achieve converged uncertainty estimates with considerably less trees compared to the random forests with jackknife estimates, offering significant computational cost savings.

Exploring Mondrian forest predictions for a number of flow cases in Section 6 provided further insights into their performance. Their uncertainty estimates appeared to provide a good measure of prediction confidence, with high uncertainty in regions of the flow where predictions were far from the truth data. As in the manufactured test case, adding more representative training data generally reduced both predictive errors and uncertainty. Comparing to random forest uncertainty estimates and an a priori distance measure suggested by Ling and Templeton [11], the Mondrian forest uncertainty estimates showed better correlation with predictive errors with fewer regions of significant under or over-confidence. Additionally, probability calibration curves suggest the Mondrian forest uncertainty estimates are well calibrated, which is important in order for the prediction intervals to be trusted. The use of SHAP values in Section 6 also demonstrates how Mondrian forest driven RANS corrections could be interpreted, possibly allowing for important physical insights into flows where high fidelity data is not available.

Based on the aforementioned findings, Mondrian forests appear to offer a promising alternative to other probabilistic methods such as BART trees and Bayesian neural networks. The current framework could easily be extended to a full data-driven turbulence modelling framework by replacing the random forests used by Wu et al. [18] with Mondrian forests. However, for such approaches to enter mainstream use, future work must address other sources of uncertainty in the ML predictions. For example, the point-wise locality assumption made in the present framework, and any differences between the RANS and NDS mean velocity and pressure fields. The first error source could be mitigated by introducing additional flow features to account for non-locality. The second error source could perhaps be reduced by modifying the present framework so that input-output training data is derived purely from NDS data. Improved accuracy could also be achieved by improving the Mondrian forest implementation itself. Mourtada, Gaïffas, and Scornet [31] have recently proposed extensions which use hyperplane splits instead of axis-aligned splits, while the response data could be used to guide splits via a recently proposed sequential Monte Carlo algorithm for decision trees [42].

A final potential area of future work is online learning. Mondrian forests and trees can be updated in an online fashion as new data becomes available, and the resulting models should be identical to their batch trained counterparts. This capability can save time, as the model must not be completely retrained when new data is obtained. Additionally, the Mondrian trees are only altered in the region of feature space populated by the new online data. As an example, this would allow for new higher Reynolds number training data to be added, without affecting the model’s predictions at a lower Reynolds number. Such capability might be important in an industrial setting where repeatability is crucial.

Acknowledgements

This work was supported by the Lloyd’s Register Foundation-Alan Turing Institute Strategic Priorities Fund. The fund is delivered by UK Research and Innovation, with this award managed by EPSRC (EP/T001569/1). The authors would like to thank Bryn Noel Ubald for his assistance in creating computational meshes for a number of the test cases, and James Tyacke for providing the ribbed channel flow data.

Appendix A Feature selection and hyperparameter tuning

A.1 Measuring predictive accuracy

To quantify predictive accuracy two metrics are used. The coefficient of determination, or $R^2$ score, given by

$$R^2 = 1 - \frac{\sum_{n=1}^{N} (y_n - \varphi(x_n))^2}{\sum_{n=1}^{N} (y_n - \bar{y})^2}$$

measures how well the true data $y$ is replicated by the model’s predictions, $\varphi(x_n)$. The $R^2$ score spuriously increases with the dimension $D$ (recall $x \in \mathbb{R}^D$), therefore the adjusted $R^2$ score is used:

$$\bar{R}^2 = 1 - (1 - R^2) \frac{N-1}{N-D-1}.$$ 

20
The mean absolute error (MAE) provides a measure of error in the model’s predictions

\[
MAE = \frac{1}{N} \sum_{n=1}^{N} |y_n - \varphi(x_n)|,
\]  

and is given as a percentage of the true mean response \( \bar{y} \) in this paper. MAE is chosen over root-mean-square error (RMSE) due to RMSE’s sensitivity to sample size and exaggeration of small numbers of large errors.

When MAE is calculated based on the training response \( y \) and training data predictions \( \varphi(x) \) it is referred to as the training error, meanwhile when it is based on the test data \( y^* \) and \( \varphi(x^*) \) it is referred to as the test error. The out-of-sample error, or test error, is important as it assesses how well the machine learning model generalises to data it has not seen during training. As is standard practice amongst machine learning practitioners, cross validation is used to provide fair error measures. The commonly used \( k \)-fold cross validation involves randomly partitioning \( D_{1:N} \) into \( k \) folds. Training is then performed on \( k-1 \) folds, with the left-out fold used as test data. This is performed \( k \) times so that all folds are used as test data, and the averaged errors over the \( k \) folds are taken. Roberts et al. [43] notes that performing such a strategy on spatial data results in serious underestimation of predictive errors due to the dependence between neighbouring observations. To avoid this problem leave-one-group-out (LOGO) cross validation is used in this paper, where each flow case in Table 1 is used as a fold.

### A.2 Feature selection

Generally, increasing the dimension \( D \) of the input feature space increases the computational cost of training and predictions, and Section 4 showed that irrelevant features can be detrimental to Mondrian forest predictions. Therefore, backward elimination is used to remove less important features. At each iteration the least important feature \( x_d \) in \( x = (x_1, \ldots, x_D)^T \) is removed, and the forest is retrained on the reduced data. Feature importance is measured using permutation importance, proposed by Breiman [29]. The permutation importance \( PI(x_d) \) quantifies the change in predictive error due to permuting the feature \( x_d \) in the input data.

Backward elimination is performed on Random forests and Mondrian forests, with features iteratively removed one at a time until \( D = 2 \). For both forests max_depth=40 and n_estimators=160. The mean \( \bar{R}^2 \) scores from LOGO cross-validation across all ten flow cases are plotted against \( D \) in Figure 20a. The key points from this plot are as follows:

- The random forest (RF) generally has a 1 – 2% higher \( \bar{R}^2 \) score than the Mondrian forest (MF), but the MF still returns competitive accuracy.
- Many features can be removed without affecting the predictive accuracy of the MF or RF. This is perhaps unsurprising, since many of the 47 invariants are zero for two-dimensional flows. For \( D < 12 \) the \( \bar{R}^2 \) scores begin to decrease as features containing important information are removed.
- The problem of irrelevant features degrading MF accuracy, observed in Section 4, is not observed here. This may be because most of the non-zero features are relevant in the flow cases considered.

![Figure 20a](image)

**Figure 20:** Effect of number of features. Features are removed using backward elimination based on permutation importance. \( \bar{R}^2 \) scores are the mean scores from LOGO cross-validation. Results are averaged over three runs.
Since the feature space can be reduced to \( D = 12 \) with no significant loss of accuracy, this is done for the remainder of this paper. When backward elimination is halted at \( D = 12 \) with the Mondrian forest, the following feature set is obtained:

\[
\mathbf{x} = \left\{ \hat{S}^2, \hat{\Omega}_p^2, \hat{\Omega}_k^2, \hat{\Omega}_\lambda^2, \hat{\Omega}_S^2, \hat{\Omega}_k^2 \hat{S}, \hat{\Omega}_k^2 \hat{S}_p, \hat{\Omega}_k^2 \hat{S}_p \hat{S}, \hat{\Omega}_k^2 \hat{S}_p \hat{S}_k \right\}^T, \tag{28}
\]

and it is used for all subsequent RF and MF computations in this paper. As shown in Figure 20b, the reduced feature space allows for reduced training/prediction times, with no appreciable loss in predictive accuracy.

A.3 Hyper-parameter tuning

To determine suitable settings for the number of trees \( M \) (\( n \_ \text{estimators} \)) and maximum depth of trees \( D_{\text{max}} \) (\( \text{max} \_ \text{depth} \)) an exhaustive grid search is performed. LOGO cross validation is performed for every combination of the hyperparameters in the sets \( M = \{2, 4, 6, 8, 10, 20, 40, 80, 100 \} \) and \( D_{\text{max}} = \{1, 5, 10, 20, 40, 60, 80, 100 \} \), and the resulting \( R^2 \) scores are shown in Figures 21a and 21b. Both RF’s and MF’s display similar trends here. The high training and test errors for low values of \( D_{\text{max}} \) in Figure 21a indicate that the shallow trees suffer from under-fitting to the \( C_{\text{aniso}} \) relationships in the data. Whereas the moderate difference between training and test errors at higher \( D_{\text{max}} \) values is possibly due to over-fitting and a lack of generalisation. Increasing \( M \) (Fig. 21b) yields only a slight reduction in MAE, suggesting that if interpretability of the ML model is important single trees \((M = 1)\) could be used with only a slight increase in predictive errors.

![Image](url)

(a) MAE, \( M = 100 \)  
(b) MAE, \( D_{\text{max}} = 40 \)  
(c) \( \sigma(C_{\text{aniso}}) \), \( D_{\text{max}} = 40 \)  
(d) \( \sigma(C_{\text{aniso}}) \), \( D_{\text{max}} = \infty \)

Figure 21: Effect of hyper-parameters on MAE and predicted \( \sigma(C_{\text{aniso}}) \). Due to the computational cost of jackknifing, the RF results in c) and d) are for case 3b only. All other results are averaged across all ten folds of the LOGO cross-validation. Error bars represent standard deviation of MAE and predicted \( \sigma(C_{\text{aniso}}) \) across individual folds in cross-validation.

A second grid search is performed over a larger range of \( M \) and \( D_{\text{max}} \) to determine the sensitivity of the prediction uncertainty \( \sigma(C_{\text{aniso}}) \) to the hyperparameters. For the RF’s \( \sigma(C_{\text{aniso}}) \) is estimated using the jackknife (Sec. 3.1.3), while for the MF’s \( \sigma(C_{\text{aniso}}) \) is naturally returned when predicting \( C_{\text{aniso}} \) (see Sec. 3.2). Similarly to the findings of Section 4, Figures 21c and 21d show that the MF’s achieve converged \( \sigma \) estimates with a relatively low number of trees \((M > 160)\). On the other hand, convergence isn’t achieved with the RF’s even with \( M > 1000 \) trees. Increasing \( M \) further (i.e. towards \( N \)) is impractical due to the significant computational cost of computing the infinitesimal jackknife for large values of \( M \).

Based on the aforementioned hyperparameter tuning, \( M = 160 \) and \( D_{\text{max}} = 40 \) is chosen for all subsequent MF’s to give low MAE’s and converged \( \sigma \) estimates.

References

[1] J. Wild. “High-Performance High-Lift Design for Laminar Wings”. In: Proc. Seventh Eur. Aeronaut. Days. Ed. by J.-P. S. D. Knörzer, C. Warsop, C. Diaconescu. London, UK: Aviation in Europe - Innovation for Growth, 2015, pp. 305–310. DOI: [10.2777/62810]

[2] P. G. Tucker and J. R. DeBonis. “Aerodynamics, computers and the environment.” In: Philos. Trans. A. Math. Phys. Eng. Sci. 372.2022 (2014). DOI: [10.1098/rsta.2013.0331]
[29] L. Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: [10.3390/rs10060911](https://doi.org/10.3390/rs10060911)

[30] L. Breiman et al. *Classification and Regression Trees.* 1st. Chapman & Hall, 1984.

[31] J. Mourtada, S. Gaïffas, and E. Scornet. “AMF: Aggregated Mondrian Forests for Online Learning”. In: *arXiv:1906.10529v1 [stat.ML]* (2019), pp. 1–32. arXiv: [1906.10529](https://arxiv.org/abs/1906.10529)

[32] Y. Bentaleb, S. Lardeau, and M. A. Leschziner. “Large-eddy simulation of turbulent boundary layer separation from a rounded step”. In: *J. Turbul.* (2012).

[33] J. Fröhlich et al. “Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constrictions”. In: *J. Fluid Mech.* 526 (2005), pp. 19–66. DOI: [10.1017/S0022112004002812](https://doi.org/10.1017/S0022112004002812)

[34] J. P. Laval and M. Marquillie. “Direct Numerical Simulations of Converging–Diverging Channel Flow”. In: *Prog. Wall Turbul. Underst. Model. ERCOFTAC Ser.* 14 (2011), pp. 203–209.

[35] L. A. Schiavo et al. “Large Eddy Simulations of convergent–divergent channel flows at moderate Reynolds numbers”. In: *Int. J. Heat Fluid Flow* 56 (2015), pp. 137–151. DOI: [10.1016/J.IJHEATFLUIDFLOW.2015.07.008](https://doi.org/10.1016/J.IJHEATFLUIDFLOW.2015.07.008)

[36] R. Vinuesa et al. *Turbulent boundary layers around wing sections up to Rec=1,000,000.* 2018. DOI: [10.1016/j.ijheatfluidflow.2018.04.017](https://doi.org/10.1016/j.ijheatfluidflow.2018.04.017)

[37] J. C. Tyacke and A. Scillitoe. “Separated flow prediction and assessment using LES and machine learning”. In: *Proc. ICNAAM.* Rhodes, Greece, 2019, pp. 1–4.

[38] S Banerjee et al. “Presentation of anisotropy properties of turbulence , invariants versus eigenvalue approaches Presentation of anisotropy properties of turbulence”. In: 5248 (2009). DOI: [10.1080/14685240701506896](https://doi.org/10.1080/14685240701506896)

[39] S. B. Pope. *Turbulent Flows.* Vol. 1. 2000, p. 771. DOI: [10.1088/1468-5248/1/1/702](https://doi.org/10.1088/1468-5248/1/1/702)

[40] S. M. Lundberg and S. I. Lee. “A unified approach to interpreting model predictions”. In: *Adv. Neural Inf. Process. Syst.* 2017. arXiv: [1705.07874](https://arxiv.org/abs/1705.07874)

[41] S. M. Lundberg et al. “Explainable AI for Trees: From Local Explanations to Global Understanding”. In: *arXiv:1905.04610v1 [cs.LG]* (2019), pp. 1–72. arXiv: [1905.04610](https://arxiv.org/abs/1905.04610)

[42] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh. “Top-down particle filtering for Bayesian decision trees”. In: *30th Int. Conf. Mach. Learn. ICML 2013.* 2013. arXiv: [1303.0561](https://arxiv.org/abs/1303.0561)

[43] D. R. Roberts et al. *Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure.* 2017. DOI: [10.1111/ecog.02881](https://doi.org/10.1111/ecog.02881)