AIAA SciTech Forum
11–15 & 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-1138

Instantaneous Flowfield Estimation with Gaussian Ridges

Ashley Scillitoe*[a], Pranay Seshadri†[a,b], and Chun Yui Wong‡[c]
[a]*The Alan Turing Institute, London, NW1 2DB, United Kingdom.*
[b]*Imperial College London, London, SW7 2BU, United Kingdom.*
[c]*The University of Cambridge, Cambridge, CB2 1TN, United Kingdom.*

Computational fluid dynamics plays a key role in the design process across many industries. Recently, there has been increasing interest in data-driven methods, in order to exploit the large volume of data generated by such computations. This paper introduces embedded Gaussian ridge functions, for rapid flowfield predictions. Gaussian ridge functions, which involve fitting a Gaussian process over a dimension reducing subspace, are obtained for numerous points within training flowfields. The functions can then be used to predict flow variables for new, previously unseen, flowfields. Their dimension reducing nature alleviates the problems associated with visualising high dimensional datasets, enabling improved understanding of design spaces and potentially providing valuable physical insights.

A training and prediction framework is proposed, and demonstrated on the incompressible flow around a set of aerofoils. The framework is computationally efficient; consisting of either heavily parallelizable tasks, or linear algebra operations. To further reduce the computational cost, the computational grid is randomly subsampled, and ridge functions are obtained only at the sampled points. The flow physics encoded within covariance matrices obtained from the training flowfields is explored, and it is found that only a number of the leading modes are required to capture most of the relevant physics. This physics can be used to predict flow quantities, conditional upon those predicted by the ridge functions at the sampled points. This enables full flowfield predictions to be obtained, despite only having ridge functions at a small number of sample points. The resulting flowfield predictions are found to be competitive with those given by a state-of-the-art convolutional neural network trained on the same data.

The underlying Gaussian processes allow for principled uncertainty quantification. Their posterior variance is incorporated into the covariance matrices, resulting in the upsampled flowfield predictions falling back on prior knowledge when predictive uncertainty is high. The end user can also view this uncertainty, giving them increased confidence in predictions. Additionally, this the possibility of including the CFD uncertainties within the framework exists, allowing for uncertainties in the CFD training data to be accounted for in the frameworks final predictions.

I. Introduction

The continued advances in computing power and computational algorithms have allowed simulation-based design and optimisation to play a key role in the industrial design process [1]. Computational fluid dynamics (CFD) simulations are an important tool across many industries; from the optimisation of aero-engines [2] and aircraft wings [3], to the preliminary design of tall buildings [4]. The workhorse of industrial CFD is Reynolds-Averaged Navier-Stokes (RANS) modelling, where a RANS model is used to represent the effects of turbulence. By avoiding the need to resolve the length and time-scales of turbulence, the cost of the computations is significantly reduced. However, in many situations, the simulations are still computationally intensive and time-consuming, leading to a costly bottleneck in the design process.

Data-driven methods offer the possibility of replacing expensive computational simulations with cheaper approximations. In a range of fields, including uncertainty quantification, design optimisation, and sensitivity analysis, lower fidelity surrogate models (or emulators) are constructed from existing simulation data. Recently, supervised deep learning methods have seen increasing attention for this purpose [5]. Deep learning architectures, routinely used in

*Research Fellow, Data-Centric Engineering, The Alan Turing Institute. Email: ascillitoe@turing.ac.uk; Web: www.ascillitoe.com.
†Research Fellow, Department of Mathematics (Statistics Section), Imperial College London; Group Leader, Data-Centric Engineering, The Alan Turing Institute. Email: p.seshadri@imperial.ac.uk; Web: www.psesh.com.
‡PhD Student, Computational Design Group, Department of Engineering, University of Cambridge. AIAA Student Member.

data mining, have been used with considerable success as a function approximation technique for high-dimensional physics derived datasets [6, 7]. Guo et al. [8] introduced the idea of using a form of deep learning, a convolutional neural network (CNN), to learn a mapping between an object's geometric representation and the flowfield around it. Bhatnagar et al. [9] and Thuerey et al. [10] recently built upon this work by introducing boundary conditions as an additional input. Jin et al. [11] take a different approach, using a CNN to capture spatio-temporal information, mapping the pressure fluctuations on a cylinder to the velocity field around it. Such approaches offer accurate and fast data-driven flowfield predictions, allowing for near-immediate feedback for real-time design iterations.

Despite their potential, a number of barriers are restricting the industrial uptake of CNN's for flowfield predictions. Firstly, their predictive errors are not easy for *testing data* to quantify or bound without running additional CFD simulations. Neural networks with principled uncertainty quantification, such as Bayesian convolution neural networks [12] , are particularly computationally expensive [13]. Secondly, deep neural networks such as CNN's are often criticised for being difficult to interpret. Although techniques for interpretation are available [14], they may not be particularly accessible for those without specialised deep learning knowledge.

An alternative approach for flowfield predictions, *embedded ridge functions*, is proposed by Wong et al. [15]. Dimension reducing ridge functions [16] are found at numerous points within a flowfield, with the target output being flowfield variables such as pressure or velocity. Reducing the high dimensional input geometry representation to a reduced number of important dimensions can provide the user with valuable physical insights alongside the flowfield predictions. In this paper, we build upon this concept by introducing *embedded Gaussian ridge functions*. Embedded Gaussian ridge functions harness the recently proposed Gaussian ridge functions (GRF's) [17], in order to provide a principled quantification of uncertainty in addition to delivering instant flowfield predictions. Principled uncertainty estimates provide a measure of prediction confidence, as well as allowing the user to determine if the training data set used to train the model is a suitable representation of unseen test data. Such capability is crucial if the framework is to be used to explore new designs.

Computationally, obtaining a Gaussian ridge function is a moderately expensive procedure. The embedded ridge function approach is embarrassingly parallel, meaning it is trivial to scale it with problem size by utilizing more processor cores. However, to obtain a model in a reasonable time frame, it is still desirable to reduce the number of points where ridge functions have to be obtained. To this end, in this paper we show how covariances in a flowfield can be used to efficiently upsample a previously downsampled flowfield. To demonstrate the utility of the resulting framework, we apply it to a dataset of incompressible flows around aerofoils, and compare predictions to a state-of-the-art CNN based flowfield prediction method.

II. Mathematical formulation

Let $(\mathbf{x}, \boldsymbol{\zeta}; s)$ represent the inputs of a scalar-field quantity of interest such as static pressure p, density ρ, a velocity component i.e. v_x, or even the turbulent viscosity ν_t, at a particular location within the flow domain $\mathcal{D}$. Here $\mathbf{x} \in \mathbb{R}^d$ parameterizes the geometry, $\boldsymbol{\zeta} \in \mathbb{R}^z$ characterizes the uncertainties—assumed to be aleatory in nature, and $s \in \mathbb{R}$ is a scalar that denotes the spatial location. For simplicity we consider s to be synonymous with the nodes within the CFD domain, where $s = 1$ corresponds to the first node and $s = N$ to the last—obviating the need for more variables to characterize the precise spatial location in Cartesian coordinates. We can think of the entire flow-field as being a collection of vectors given by

$$\mathcal{F} = \left\{ \begin{bmatrix} p(\mathbf{x}, \boldsymbol{\zeta}; s_1) \\ \vdots \\ p(\mathbf{x}, \boldsymbol{\zeta}; s_N) \end{bmatrix}, \begin{bmatrix} t(\mathbf{x}, \zeta, s_1) \\ \vdots \\ t(\mathbf{x}, \boldsymbol{\zeta}; s_N) \end{bmatrix}, \begin{bmatrix} v_x(\mathbf{x}, \boldsymbol{\zeta}; s_1) \\ \vdots \\ v_x(\mathbf{x}, \boldsymbol{\zeta}; s_N) \end{bmatrix}, \ldots, \begin{bmatrix} \rho(\mathbf{x}, \boldsymbol{\zeta}; s_1) \\ \vdots \\ \rho(\mathbf{x}, \boldsymbol{\zeta}; s_N) \end{bmatrix} \right\}. \tag{1}$$

For purposes of exposition we refer to any scalar field quantity by f and thus with a slight abuse in notation, each of the vectors in $\mathcal{F}$ is generalized via

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}, \boldsymbol{\zeta}; s_1) \\ \vdots \\ f(\mathbf{x}, \boldsymbol{\zeta}; s_N) \end{bmatrix}. \tag{2}$$

In this section we introduce the mathematical machinery that will be used to rapidly estimate flowfields from computational fluid dynamics.

A. Ridge approximations

Identifying a suitable surrogate model for $\mathbf{f}$ is challenging because it effectively has $\mathbb{R}^d \times \mathbb{R}^z \times \mathbb{R}^N$ degrees of freedom, leading to an insuperable number of model evaluations for designing a vanilla, physically oblivious surrogate model. However, a surrogate model of some form is necessary to arrive at a flowfield estimate. To thwart the high-cost of constructing such a surrogate, a key observation is leveraged: many seemingly high-dimensional CFD problems have intrinsically low-dimensional structure. Known under the handles of active subspaces [18], sufficient dimension reduction [19], and ridge functions [16], these methods posit that high-dimensional functions lie in some low dimensional sub-manifold, delineated by a few, particular linear combinations of all input parameters.

To clarify this, consider two scalar-valued functions $h : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$, where $m << d$ for some $\mathbf{x} \in \mathcal{X}$. We define g to be a ridge approximation for h if

$$h(\mathbf{x}) \approx g\left(\boldsymbol{W}^T\mathbf{x}\right), \tag{3}$$

holds for some orthogonal matrix $\boldsymbol{W} \in \mathbb{R}^{d\times m}$. By construction the columns of $\boldsymbol{W}$ identify important linear combinations within $\mathcal{X}$ which are necessary for estimating h. The function g is a low-dimensional emulator of h; its functional form can inform us what variation h exhibits when perturbed along the subspace $\boldsymbol{W}^T\mathbf{x}$. The expression in (3) can be phrased as an optimisation problem in the standard Euclidean norm

$$\underset{g,\mathbf{W}}{\text{minimise}} \quad \left\| h(\mathbf{x}) - g\left(\mathbf{W}^T\mathbf{x}\right)\right\|_2^2, \tag{4}$$

where the subscripts indicate that there are two different spaces over which this minimisation needs to be undertaken. First, let us consider g. If g is a polynomial, then its coefficients need to be determined; if g is a kernel-based surrogate, such as a Gaussian process, then both the covariance function and its associated hyperparameters need to be determined. Second, the subspace $\boldsymbol{W}$, which requires an input matrix with $d \times m$ entries needs to be obtained. Although the columns of $\boldsymbol{W}$ do not necessarily have to be orthogonal, they must not be linear combinations of each other. In other words the rank of $\boldsymbol{W}$ must be m. Approaches for idenfiying such strcuture, i.e., solving both optimisation problems simultaneously, include the works of Hokanson and Constantine [20] and Seshadri et al. [17].

Whilst typically used in the context of global scalar quantities, such as the lift and drag coefficients [21], and other aerothermal performance metrics [22–24], Wong et al. [15] identifies such structure in scalar-fields too, setting the stage for our work. Here we assume that our scalar-field quantities can be approximated via

$$\mathbf{f} \approx \begin{bmatrix} g_1\left(\mathbf{W}_1^T\mathbf{x},\ \mathbf{Z}_1^T\boldsymbol{\zeta};\ s_1\right) \\ \vdots \\ g_N\left(\mathbf{W}_N^T\mathbf{x},\ \mathbf{Z}_N^T\boldsymbol{\zeta};\ s_N\right) \end{bmatrix} \tag{5}$$

where $\left\{\boldsymbol{W}_i \in \mathbb{R}^{d\times n} \mid i = 1,\ldots,N\right\}$ and $\left\{\mathbf{Z}_i \in \mathbb{R}^{d\times k} \mid i = 1,\ldots,N\right\}$ for $n << d$ and $k << d$. Functions $g_1,\ldots,g_N$ are the low-dimensional ridge approximations that need to be estimated at the N nodes. For reasons that will become clear later, in what follows, we will assume that each ridge approximation is independent of the other, and thus provided we have a recipe for finding a ridge approximation, obtaining (5) is an embarrassingly parallel operation.

B. Probabilistic perspective of CFD

Without loss in generality, we focus our efforts on estimating the flowfield for a specific set of boundary conditions, but where the geometry is permitted to vary. In other words, each primal flow quantity can be given by

$$\mathbf{f} \approx \begin{bmatrix} g_1\left(\mathbf{W}_1^T\mathbf{x},\ \boldsymbol{\zeta};\ s_1\right) \\ \vdots \\ g_N\left(\mathbf{W}_N^T\mathbf{x},\ \boldsymbol{\zeta};\ s_N\right) \end{bmatrix}. \tag{6}$$

Let us assume that each g above can be well represented by a Gaussian process model. In other words, g is completely defined by its mean μ and a two-point covariance function Σ [25], given by

$$g \sim \mathcal{N}\left(\mu\left(\boldsymbol{W}^T\mathbf{x}\right), \Sigma\left(\boldsymbol{W}^T\mathbf{x}, \boldsymbol{W}^T\mathbf{x}'\right)\right), \tag{7}$$

which is constructed in m dimensions and not d. The distribution of g evaluated at different design parameters is jointly Gaussian. Let $\{\hat{\mathbf{x}}_M, \ldots, \hat{\mathbf{x}}_M\}$ be a set of discretised input parameters, for which flowfield scalar fields are avaliable. Restricting our attention to the data from a single node s_1 across all M flowfields, we set

$$\begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_M \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1, \boldsymbol{\zeta}; s_1) \\ \vdots \\ f(\mathbf{x}_M, \boldsymbol{\zeta}; s_1) \end{bmatrix}. \tag{8}$$

The mean of the Gaussian process model in (7), conditioned upon the training data provided, is given by

$$\mu(\mathbf{z}) = \boldsymbol{R}^T \left(\boldsymbol{C} + \sigma^2 \boldsymbol{I}\right)^{-1} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_M \end{bmatrix}, \tag{9}$$

where the (i, j) entries of the matrices above are given by

$$\boldsymbol{R}_{ij} = k\left(\boldsymbol{W}^T \hat{\mathbf{x}}, \boldsymbol{W}^T \mathbf{x}'\right), \quad \boldsymbol{C}_{ij} = k\left(\boldsymbol{W}^T \hat{\mathbf{x}}, \boldsymbol{W}^T \hat{\mathbf{x}}'\right), \tag{10}$$

and where k is a user-chosen kernel function, and $\sigma_m^2 \boldsymbol{I}$ is a diagonal covariance matrix, where the true scalar-field value at node s_1 is assumed to be corrupted by some noise σ_m^2. The covariance of g, also conditioned upon the training data, is given by

$$\Sigma = \boldsymbol{K} - \boldsymbol{R}^T \left(\boldsymbol{C} + \sigma_m^2 \boldsymbol{I}\right)^{-1} \boldsymbol{R}, \tag{11}$$

where $\boldsymbol{K}_{ij} = k\left(\boldsymbol{W}^T \mathbf{x}, \boldsymbol{W}^T \mathbf{x}'\right)$. The kernel function used in this paper is the squared exponential kernel (see Chapter 2 in [25]) which has the form

$$k\left(\boldsymbol{W}^T \mathbf{x}, \boldsymbol{W}^T \mathbf{x}'\right) = \sigma_f^2 \ \exp\left(-\frac{1}{2}\left(\boldsymbol{W}^T \mathbf{x} - \boldsymbol{W}^T \mathbf{x}'\right)^T \boldsymbol{\Gamma}^{-1} \left(\boldsymbol{W}^T \mathbf{x} - \boldsymbol{W}^T \mathbf{x}'\right)\right), \tag{12}$$

where σ_f is the signal variance hyperparameter and $\boldsymbol{\Gamma} = \text{diag}\left[l_1^2, \ldots, l_m^2\right]$ is a diagonal matrix of the length scale hyperparameters. Values of these hyperparameters are conditioned upon the training data, the measurement noise matrix $\sigma^2 \boldsymbol{I}$, and the functional relationship associated with the kernel.

While such a probabilistic description for evaluations of a deterministic CFD model may seem superfluous, we argue it is not without rationale. At the application-level, practioners are all too aware of the limitations of CFD, and while perhaps not formally, they can attribute some degree of confidence based on the boundary conditions and geometry. For instance, there is likely greater confidence in RANS predictions of shear stresses at peak-efficiency than at stall. This *engineering judgement* can be expressed as a confidence interval or an error bar for CFD-yielded scalar fields depending on the boundary conditions. In cases where the boundary conditions themselves are not precisely known, aleatory uncertainty quantification techniques (see [26]) can be used. These provide a principled approach for estimating moments for scalar- and vector-valued quantities of interest. Finally, more recent epistemic uncertainty quantification efforts [27–29] can deliver either interval or distributions for a single CFD evaluation. This supports our perspective of interpreting each CFD evaluation as having a mean and variance.

C. Gaussian ridge functions

To idenfity both the dimension reducing subspace $\boldsymbol{W}$ and the hyperparameters $\theta = \left(\sigma_f, l_1, \ldots, l_m\right)$ associated with the Gaussian process model, we adopt the methodology in Seshadri et al. [17]. Here the authors use an alternating optimisation approach, altering between finding an appropriate dimension reducing subspace, and identifying the hyperparameters for the Gaussian distribution. The former objective is expressed as

$$\begin{aligned} \underset{\boldsymbol{W}}{\text{minimise}} \quad & \left\| \begin{bmatrix} f_1^* \\ \vdots \\ f_K^* \end{bmatrix} - \begin{bmatrix} \mu\left(\boldsymbol{W}^T \mathbf{x}_1^*\right) \\ \vdots \\ \mu\left(\boldsymbol{W}^T \mathbf{x}_K^*\right) \end{bmatrix} \right\|_2^2, \\ \text{subject to} \quad & \boldsymbol{W} \in \mathbb{S}t(m, d), \end{aligned} \tag{13}$$

where $\{\mathbf{x}_i^*, f_i^*\}_i^K$ represents a K-point *validation* data set. Problem (13) is a manifold optimisation problem, where the constraint corresponds to the Stiefel manifold: the set of all $d \times m$ orthogonal matrices [30]. Gradients for $\boldsymbol{W}$ along the Stiefel manifold can be readily derived as the objective function here is differentiable. Note that only the mean function of the Gaussian process is used in (13), for a given set of hyperparameters. The hyperparameters are in turn the outcome of the second (alternate) optimisation problem given by

$$\begin{aligned} &\underset{\theta}{\text{minimise}} \quad -\log \mathrm{p}\,(\theta) \\ &\text{subject to} \quad \log \mathrm{p}\,(\theta) = -\frac{1}{2}\begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_M \end{bmatrix}^T \left(\boldsymbol{C} + \sigma_m^2 \boldsymbol{I}\right)^{-1} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_M \end{bmatrix} - \frac{1}{2}\log\left|\boldsymbol{C} + \sigma_m^2 \boldsymbol{I}\right| - \frac{M}{2}\log\left(2\pi\right), \end{aligned} \tag{14}$$

where only the *training* data set is used[1]. Problem (14) is the standard maximum likelihood optimisation formulation (see Chapter 5 in [25]), which can be readily solved by a gradient-based optimiser.

III. Computational setup

The flowfield explored in this paper is the flow around the well known NACA0012 aerofoil, discretised with a 449x129 curvilinear C-mesh[2]. Various aerofoil designs are obtained by perturbing the baseline geometry using $d = 50$ Hicks-Henne bump functions [32]. Each bump function is given by

$$f_j(x) = \left[\sin\left(\pi x^{\frac{\log(0.5)}{\log(t_1)}}\right)\right]^{t_2} \tag{15}$$

where t_1 and t_2 control the x location and the width of the bump. The perturbed aerofoil coordinates are then obtained with

$$y(x) = y_{base}(x) + \sum_{j=1}^{d} \beta_j f_j(x) \tag{16}$$

where y_{base} are the y coordinates of the baseline aerofoil, x and y are normalised by the aerofoil's axial chord length C_x. The bump amplitudes $[\beta_1, \ldots, \beta_d]$ are then stored within the input vector $\mathbf{x}^{(m)} \in \mathbb{R}^d$ for each design.

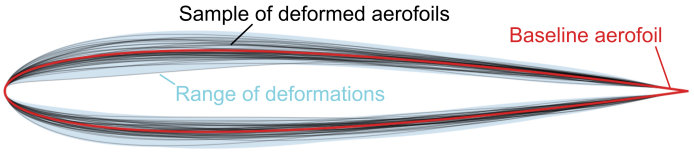


Fig. 1 Deformations made to the NACA0012 aerofoil. Fifty randomly selected deformed designs are shown.

To generate a dataset for training and testing, we create a ($M = 2000$)-point design of experiment with uniformly distributed Monte Carlo samples for $\mathbf{x}$. A randoly sample of the resulting aerofoil designs, as well as the full range of deformations, is shown in FIgure 1. Flowfields are simulated for each design using the incompressible solver of the SU2 CFD code [33]. The commonly used SA RANS model is used to represent the effects of turbulence, with the inlet turbulence viscosity ratio set to $\nu_t/\nu = 5$. The inlet velocity magnitude is set to $U_1 = 1$m/s, and the exit static pressure is set to zero. The laminar viscosity ν is then set to give a Reynolds number of $Re = U_1 C_x/\nu = 6 \times 10^6$. Each design is run at two angles of incidence $\alpha = 0^\circ$ and 10°, leading to a dataset consisting of 4000 flowfields in total.

The grid deformations performed when perturbing the baseline aerofoil to reach each new design means that, even away from the aerofoil surface, grid points are at a slightly different location for each design. Since we wish to learn a functional mapping for the flowfield variables at fixed points in space, we re-sample[3] each flowfield onto a single

[1]The *training* and *validation* data sets are obtained by further splitting the *training* data set referred to in the rest of this paper 70/30.

[2]This grid is used as a verification case by the AIAA Fluid Dynamics Technical Committee Turbulence Model Benchmarking Working Group (TMBWG) and is available from turbmodels.larc.nasa.gov/naca0012_grids.html. Diskin et al. [31] show lift and drag coefficients to be sufficiently grid independent at the 449x129 grid resolution.

[3]Resampling is performed with the pyvista python library [34], which uses linear interpolation for resampling.

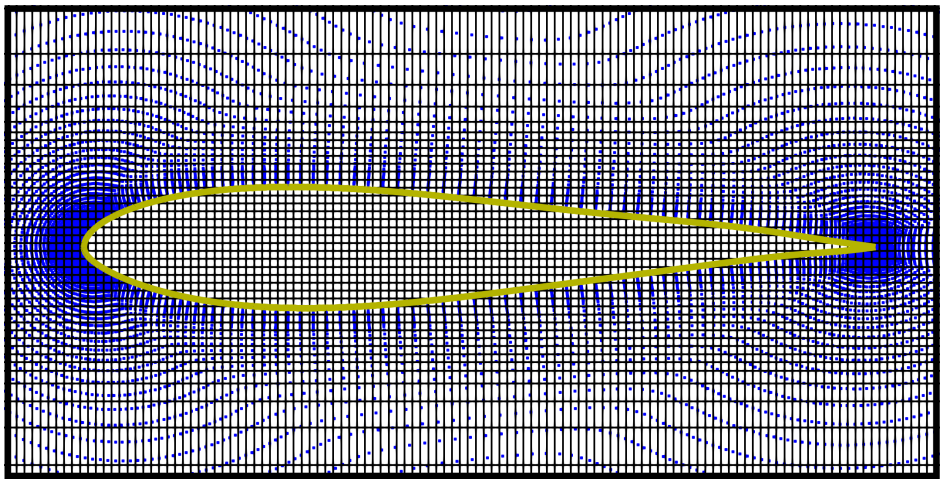
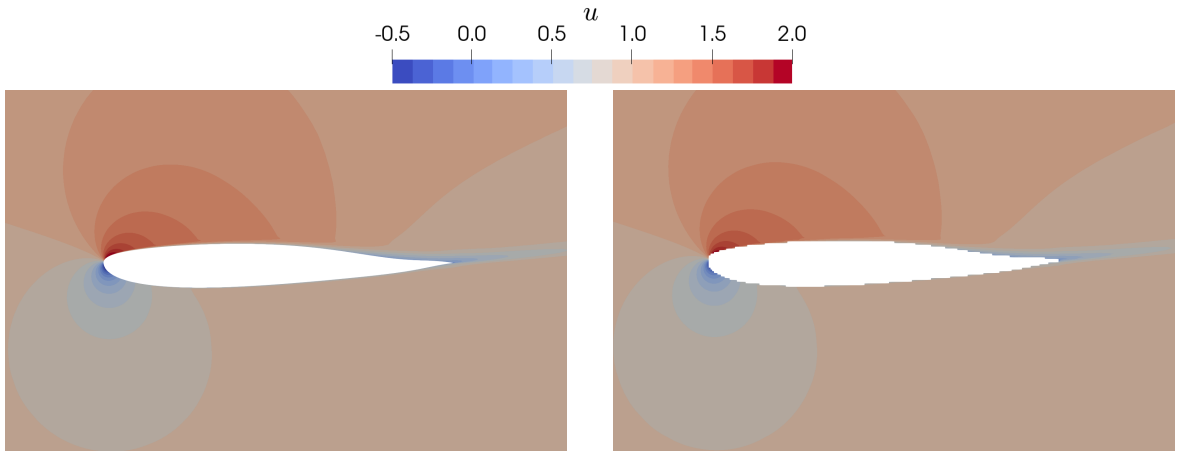


Fig. 2 Zoomed-in view of the discrete curvilinear C-grid representation of a deformed aerofoil (blue points), and the Cartesian grid (black) the flow variables are resampled onto. The aerofoil boundary is shown in yellow.

90x318 Cartesian grid, shown in Figure 2. grid points lying inside the solid region of the aerofoil are removed. As seen in Figure 3, the re-sampling procedure returns a reasonably good quality flow-field. We choose a Cartesian grid here since we are interested in full flowfield visualisations, but the framework can also be applied to individual points, surfaces (see Ref. [15]) or planes.



(a) Curvlinear C-grid. **(b) Cartesian grid.**

Fig. 3 Contours of u velocity for a deformed aerofoil at an angle of incidence of $\alpha = 10°$, before and after re-sampling.

After the above procedure we are left with $N = 90 \times 318 = 28620$ input/output pairs $(\mathbf{x}^{(m)}, f_i^{(m)})_{m=1}^{M}$, for each of the two angles of incidence investigated. The scalar $f_i^{(m)}$ is the field variable at the i^{th} grid point for the m^{th} design, where in this paper we take the static pressure p, axial velocity u, and turbulent viscosity ν_t as the field variables to predict. We normalise the aforementioned variables by taking the static pressure coefficient $C_p = (p - p_1)/(p_{01} - p_1)$, axial velocity ratio u/U_1, and turbulent viscosity ratio ν_t/ν. The $M = 2000$ designs are further split up into $M_{train} = 1000$ training designs and $M_{test} = 1000$ testing designs.

A. Downsampling and upsampling

Since obtaining ridge functions for all $N = 28620$ points in the Cartesian grid would be a rather time consuming operation, we first downsample the grid by taking a random subsample of $N_c = 1000$ points, at then obtain ridge

functions on the resulting *coarse* set of points. If full flowfield predictions are required, we need a way to upsample the ridge function predictions at the $N_c = 1000$ *coarse* points back to the $N_f = N - N_c = 27620$ remaining *fine* grid points.

To achieve upsampling, we exploit the covariance of the flowfields. For a given field variable $\mathbf{f} \in \mathbb{R}^N$, the covariance between f at the i^{th} and j^{th} points is given by

$$\Sigma_{ij} = \frac{1}{M} \sum_{m=1}^{M} \left(f_i - \mathbb{E}\left[\mathbf{f}\right]\right) \left(f_j - \mathbb{E}\left[\mathbf{f}\right]\right), \tag{17}$$

where the expectation of $\mathbf{f}$, $\mathbb{E}\left[\mathbf{f}\right]$, is the mean $\mathbf{f}$ field, averaged across the M number of designs. For computational efficiency this operation is vectorised, and applied to the entire flowfield for the $M_{train} = 1000$ training designs

$$\mathbf{\Sigma} = \frac{1}{M_{train}} \sum_{m=1}^{M_{train}} \left(\mathbf{f} - \mathbb{E}\left[\mathbf{f}\right]\right) \left(\mathbf{f} - \mathbb{E}\left[\mathbf{f}\right]\right)^T, \tag{18}$$

yielding a covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$, which contains the covariance between the field variable f at each and every grid point. This matrix is then rearranged into the form

$$\mathbf{\Sigma} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix} \tag{19}$$

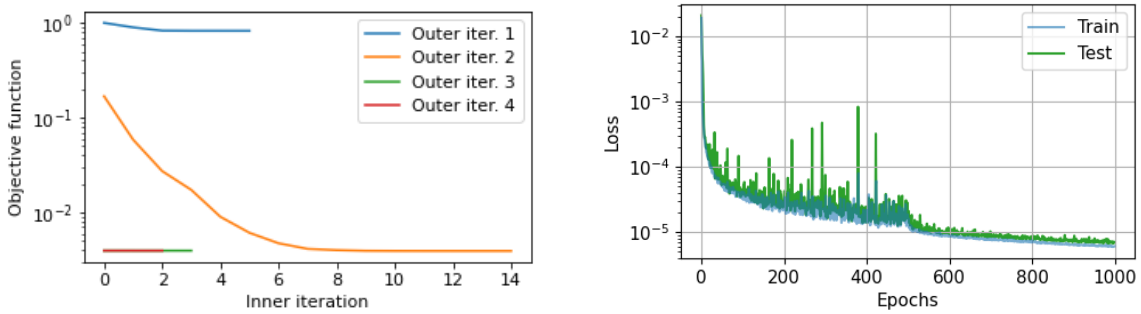
where $\boldsymbol{A} \in \mathbb{R}^{N_f \times N_f}$ is the covariance matrix for the *fine* points, $\boldsymbol{C} \in \mathbb{R}^{N_c \times N_c}$ is the covariance matrix for the *coarse* points, and $\boldsymbol{B} \in \mathbb{R}^{N_f \times N_c}$ is the covariance matrix between the two sets of points. Then, by taking advantage of the Schur complement of $\boldsymbol{C}$ [35], we can obtain the expected values of $\mathbf{f}$ at the *fine* points, $\mathbf{f}_f$, conditional on the values at the *coarse* points, $\mathbf{f}_c$:

$$\mathbb{E}\left[\mathbf{f}_f | \mathbf{f}_c\right] = \mathbb{E}\left[\mathbf{f}_f\right] + \boldsymbol{B}\boldsymbol{C}^{-1} \left(\mathbf{f}_c - \mathbb{E}\left[\mathbf{f}_c\right]\right). \tag{20}$$

where the expectations at the *fine* and *coarse* points, $\mathbb{E}\left[\mathbf{f}_f\right]$ and $\mathbb{E}\left[\mathbf{f}_c\right]$, are obtained by averaging the training data at each point. For a given design m, a predicted flowfield can now be obtained by replacing the true values at the *coarse* points, $\mathbf{f}_c$ in (20), with the posterior mean of the Gaussian ridge functions, $\bar{g}_i$, for $i = 1, \ldots, N_c$.

B. Numerical implementation

The framework described above is implemented in Python[4], with the code parallelised using the `joblib` python library (with the `loky` backend), and linear algebra operations performed with `numpy` [36]. Since each embedded subspace is independent of one another, the algorithm can be implemented in an embarrassingly parallel fashion, where computation of individual embedded ridge approximations for each subsampled point are computed by individual Python worker processes.



(a) GRF trained on data at a single subsampled point **(b) CNN trained on the entire dataset**

Fig. 4 Convergence histories for an embedded GRF and a CNN, trained on the static pressure coefficient data C_p, at an angle of incidence of $\alpha = 10$.

[4]The code is made publicly available at https://github.com/ascillitoe/gaussian_flowfield_approx.

Each parallel Python process is run concurrently on a separate computational core on a F72s_v2 virtual machine (72 virtual CPU's, 144 GiB memory) on the Microsoft Azure cloud computing service. For the 1000 sub-sampled points, with 1000 training designs, training takes approximately 30 minutes. A typical convergence history is shown in Figure 4a; the inner manifold optimisation, equation (13), usually terminates when a minimum step-size (1×10^{-10}) is reached. The outer iteration loop, which repeatedly performs equations (13) and (14), is halted once the normalised error falls below a given tolerance (1×10^{-5}).

C. Comparision with a convolutional neural network

To examine the effectiveness of the proposed framework for flowfield predictions, we compare its performance to a state-of-the-art convolutional neural network (CNN) architecture. CNN's use convolutional layers to take advantage of local spatial coherence in the input. These layers, combined with successive spatial resizing of the input data, can significantly reduce the number of weights relative to fully connected neural networks. Numerous studies [8–10] have recently used such an architecture for flowfield predictions with good success.

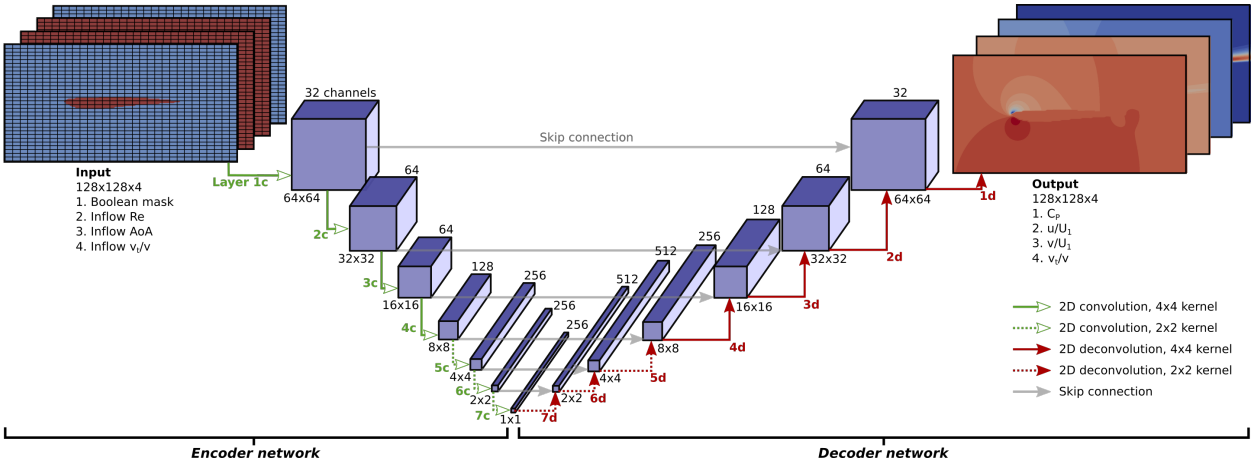


Fig. 5 The 488k weight U-net convolutional neural network architecture used for flowfield predictions. Arrows indicate the direction of forward operations. The network is fully convolutional with 14 layers. The convolutional blocks consist of an activation function, convolution filter, batch normalization (except for layers 1c and 1d) and dropout. For deconvolution, nearest neighbour upsampling is used, followed by a regular convolution.

Thuerey et al. [10] explored a number of CNN architectures for flowfield predictions, and found the U-Net architecture [37] to be the most successful. We implement a modified version of the framework proposed by Thuerey et al., shown in Figure 5, in order to predict the C_p, u, v and ν_t/ν fields. The input data consists of four 128×128 channels, a boolean mask to define the aerofoil geometries, and three uniform fields prescribing the inlet Reynolds number, angle of incidence, and turbulent viscosity ratio. The U-Net architecture consists of an encoder, which progressively down-samples the $128 \times 128 \times 4$ input channels with strided convolutions. This allows the network to extract increasingly large-scale and abstract information as the number of feature channels grows, until we are left with a $1 \times 1 \times 256$ derived feature vector. The decoder then does the opposite, with depooling layers reducing the number of features while increasing the spatial resolution. Skip connections help the network to consider low-level input information during the reconstruction of the solution in the decoding layers.

Learning the network's weights[5] involves an optimisation process to minimise a loss function. The $M_{train} = 1000$ flowfields are resampled onto a $N = 128^2 = 16384$ Cartesian grid, where they are then used in the loss function

$$\mathcal{L} = \frac{1}{M_{train}} \frac{1}{N} \sum_{m=1}^{M_{train}} \sum_{i=1}^{N} \phi_{hub}\left(\hat{p}_i^{(m)} - p_i^{(m)}\right) + \phi_{hub}\left(\hat{u}_i^{(m)} - u_i^{(m)}\right) + \phi_{hub}\left(\hat{v}_i^{(m)} - v_i^{(m)}\right) + \phi_{hub}\left(\hat{\nu_t}_i^{(m)} - \nu_{t_i}^{(m)}\right), \tag{21}$$

[5]We include both weights and biases under the term *weights*.

with $\hat{\cdot}$ denoting the field variables predicted by the network, and ϕ_{hub} being the Huber loss function [38]. In the encoder we use leaky ReLU activation functions to avoid vanishing gradients, whilst in the decoder we use standard ReLU's, with a slope of 0.2 for both. To help mitigate overfitting, a slight dropout rate of 0.02 is adopted for all layers.

The above CNN architecture is implemented in the PyTorch python library, and trained on a NVIDIA GTX970 1664 core GPU, with training times ranging from 50 seconds to 80 minutes depending on the network and dataset size. Numerous network sizes, ranging from 32k to 1.94M weights, were tested. A network with 488k weights was found to offer a good compromise between training times and accuracy, with a training time of approximately 50 minutes when $M_{train} = 1000$. The learning curve for the final CNN, shown in Figure 4b, shows that good convergence is achieved within 1000 Epochs.

IV. Results

Following the procedure outlined in Section II, the 1000 training designs are resampled onto a Cartesian grid, which is then downsampled by randomly selecting the $N_c = 1000$ points shown in Figure 6. At each subsampled point a Gaussian ridge function with $n = 1$ reduced dimensions is obtained for static pressure coefficient C_p, normalised axial velocity u/U_1, and turbulent viscosity ratio ν_t/ν. The three sets of embedded ridge functions, $g^{(p)}$, $g^{(u)}$ and $g^{(\nu_t)}$, are Gaussian process regressors, fit over the reduced input coordinates $r_i^{(m)} = \mathbf{W}_i^T \mathbf{x}^{(m)}$ at each point. Each one can be visualised as a *sufficient summary plot*, four of which are shown in Figure 6. The training and test designs collapse relatively tightly onto the posterior mean of the ridge functions, demonstrating that a low dimensional structure is present here. Comparing the design in question (highlighted by a gold circle in the summary plots), to the training and test designs, informs us how the design compares to the other designs in the dataset for different points in the flowfield. By moving around this summary plot, we see how C_p varies at the associated point as we alter a design's reduced input coordinate $r_i^{(m)}$.

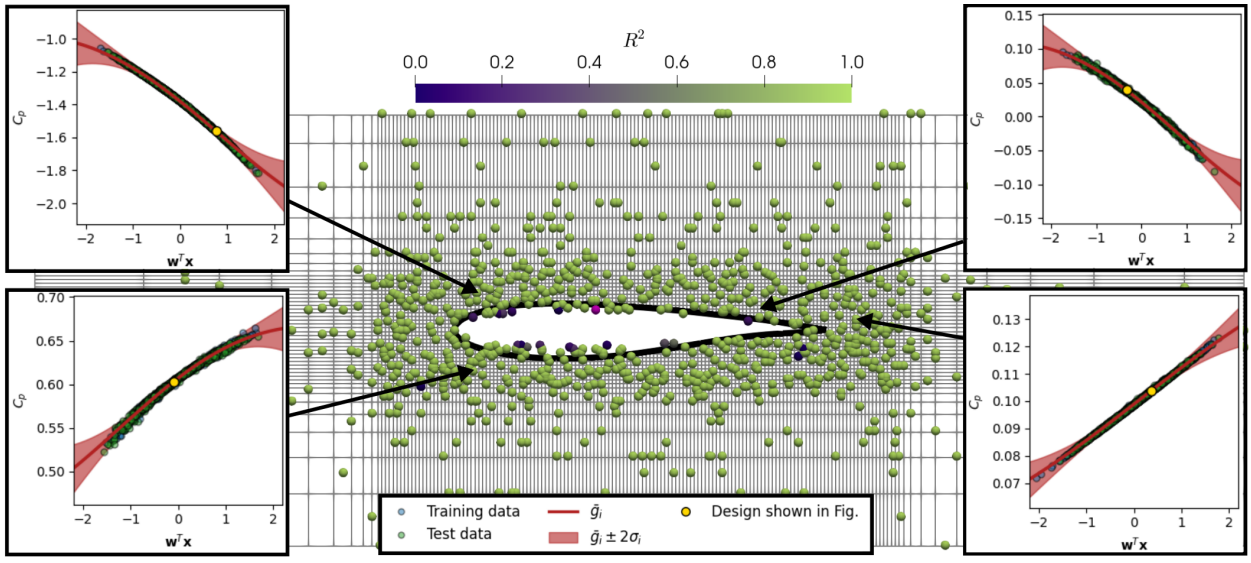


Fig. 6 The $N_c = 1000$ subsampled points, overlaid onto the $N = 28620$ point Cartesian grid, shown for a deformed aerofoil from the *training* set. At each of the subsampled points a GRF is fit to the *training* data for C_p (at $\alpha = 10°$), and the points are coloured by the GRFs' *test* R^2 scores. Sufficient summary plots for C_p at four locations are also shown.

Examining sufficient summary plots at selected points can provide valuable insights into a flow. However, in the present case, we are more concerned with flowfield predictions. The ridge functions can be evaluated for a given design, for example for a design m with design vector $\mathbf{x}^{(m)}$, C_p at the i^{th} point is obtained from the posterior mean $\bar{g}_i^{(p)}$

$$C_{p\,i}\left(\mathbf{x}^{(m)}\right) = \bar{g}_i^{(p)}\left(\mathbf{W}_i^{(p)^T}\mathbf{x}^{(m)}\right), \tag{22}$$

where $\mathbf{W}_i^{(p)}$ is the subspace matrix for C_p at the i^{th} point. In Figure 6, the subsampled points are coloured by the R^2

correlation coefficient

$$R_i^2 = 1 - \frac{\sum_{m=1}^{M}(f_i^{(m)} - \bar{g}_i(\mathbf{x}^{(m)}))^2}{\sum_{m=1}^{M}(f_i^{(m)} - \bar{f}_i)^2}, \tag{23}$$

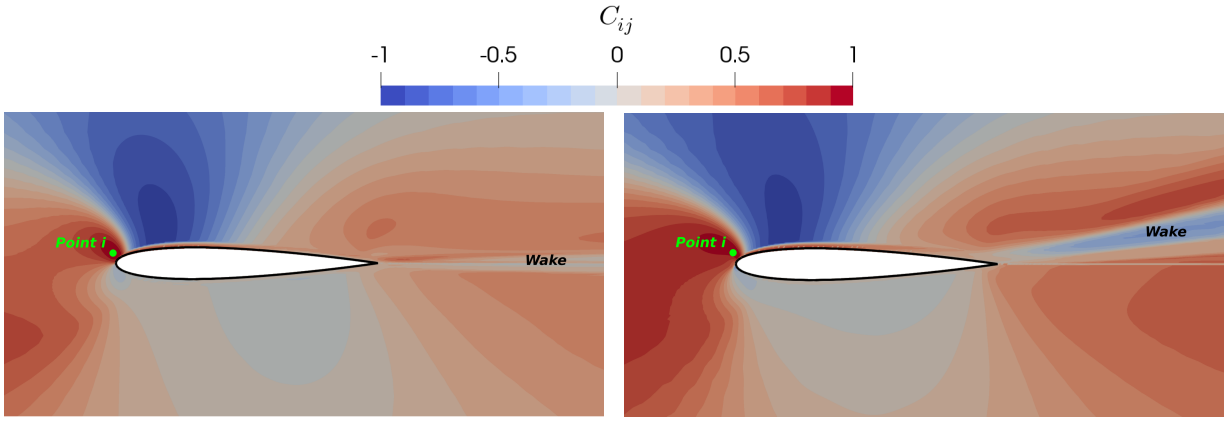
with $\bar{f}_i$ is the mean value of f_i across the M designs. For the majority of points, R_i^2 is close to one, implying the predicted values $\bar{g}_i(\mathbf{x}^{(m)})$ are well correlated with the true values $f_i^{(m)}$.

A. Exploiting covariance

The Gaussian ridge functions above can provide accurate predictions, and their sufficient summary plots can be examined to provide insights into the flow. However, for full flowfield predictions, we must still upsample their predictions back to the original Cartesian grid. As discussed in Section III.A, we can use the covariance matrices of the flowfields for this purpose. To understand this process in more detail, we will first examine the underlying physics of the covariances in more detail. To begin, the Pearson product-moment correlation coefficient between the field variable f at two points can be obtained from the covariance matrix

$$C_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}. \tag{24}$$

In Figure 7, we take the covariance matrix for the axial velocity, and display the correlation coefficient C_{ij} for a given *point i* near the leading edge. This visualises the correlation between the axial velocity at point i, and that at every other j^{th} point in the flowfield. Unsurprisingly, there is strong positive correlation between points close to i and itself. More interestingly, there is strong negative correlation between the velocity at points above the suction surface, immediately downstream of the leading edge. This implies that design perturbations which increase velocity at point i, tend to decrease velocity in this region. Along similar lines, the negative correlation in the wake in Figure 7b suggests that, at an angle of incidence of $\alpha = 10°$, increasing the velocity near the leading edge increases the velocity defect in the aerofoil's wake.



(a) $\alpha = 0°$ **(b)** $\alpha = 10°$

Fig. 7 Pearson product-moment correlation coefficient fields for axial velocity, showing correlation with respect to the highlighted point.

The above analysis suggests that the covariance matrices contain information on the underlying flow physics. With equation (20), we can exploit this encoded physics to obtain values of the flowfield variables at the remaining grid points in a more principled way compared to brute force approaches such as nearest neighbour interpolation. The downside is that the $N \times N$ covariance matrix for each flowfield must be stored. This introduces a storage requirement with an N^2 scaling, which could be problematic for larger grids. To mitigate this, it is desirable to find a low-rank approximation for Σ. To this end, we perform an eigendecomposition of each covariance matrix

$$\boldsymbol{\Sigma} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T. \tag{25}$$

The square matrix $Q \in \mathbb{R}^{N \times N}$ contains the eigenvectors $\boldsymbol{Q} = [v_1, \ldots, v_N]$, and $\boldsymbol{\Lambda}$ is a diagonal matrix holding the eigenvalues $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$. Two of the leading eigenvectors for the C_p covariance matrix are displayed in Figure 9. Each eigenvector can be thought of as representing a particular mode of the flowfield. The first 1000 eigenvalues for each of the covariance matrices are plotted in Figure 8. For all four field variables, especially the turbulent viscosity ν_t, the magnitude of their eigenvalues drops off quickly. This suggests that most of the flow physics is captured by only a small number, $l << N$, of the leading eigenvalues and eigenvectors.

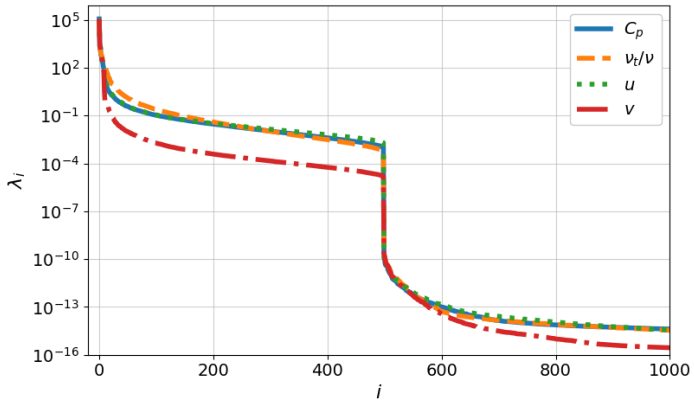
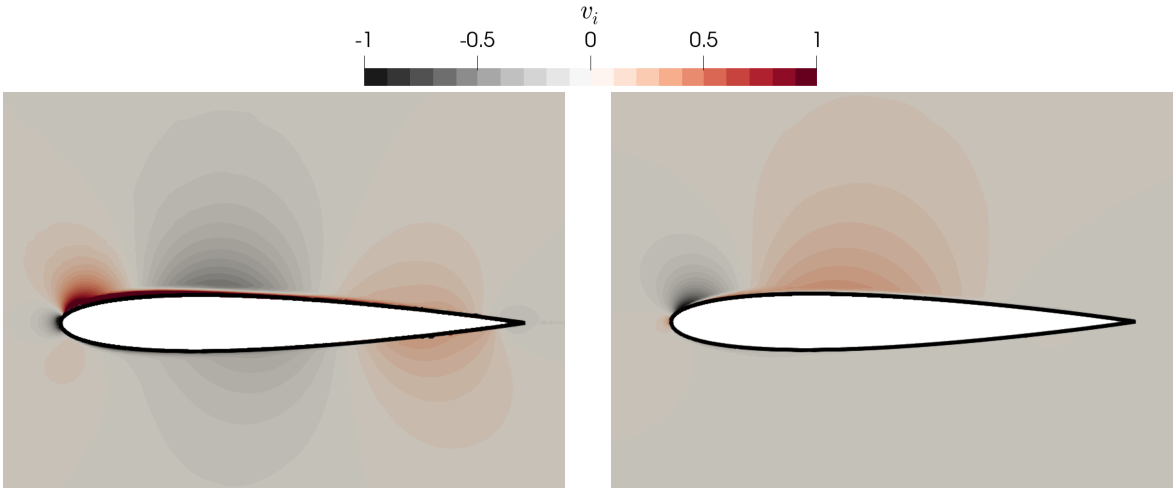


Fig. 8 The first 1000 eigenvalues of the field variables' covariance matrices.



(a) First **(b) Fifth**

Fig. 9 First and fifth eigenvectors of the covariance matrix for the static pressure coefficient fields at $\alpha = 10°$. The eigenvector fields are normalised by the maximum absolute value in each field.

By selecting the l number of leading eigenvalues and eigenvectors, so that $\boldsymbol{Q}^* \in \mathbb{R}^{N \times l}$ and $\boldsymbol{\Lambda}^* \in \mathbb{R}^{l \times l}$, a low-rank approximation for the covariance matrix can be obtained

$$\boldsymbol{\Sigma}^* = \boldsymbol{Q}^* \boldsymbol{\Lambda}^* \boldsymbol{Q}^{*T}. \tag{26}$$

After computing the full-rank covariance matrices from the training data, only $\boldsymbol{Q}^*$ and $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_l]$ must be saved, and the approximate Σ^* matrix can then be reconstructed at prediction time. If $l << N$, this offers a significant reduction in memory requirements relative to storing the full Σ matrix. In Figure 10, C_p predictions from Gaussian ridge functions at the $N_c = 1000$ *coarse* points have been upsampled to the remaining points using equations (19) and (20), with $\boldsymbol{\Sigma}$ replaced by $\boldsymbol{\Sigma^*}$. The $l = 20$ leading modes were used to construct $\boldsymbol{\Sigma^*}$, and as can be seen by comparing the prediction (colour contours) to the CFD solution (black iso-lines), this is sufficient to obtain a reasonably accurate flowfield prediction.

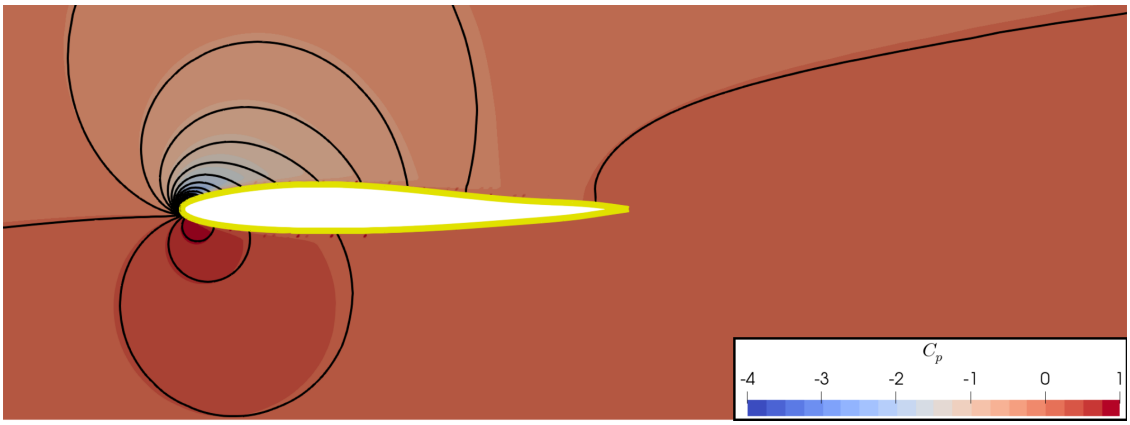
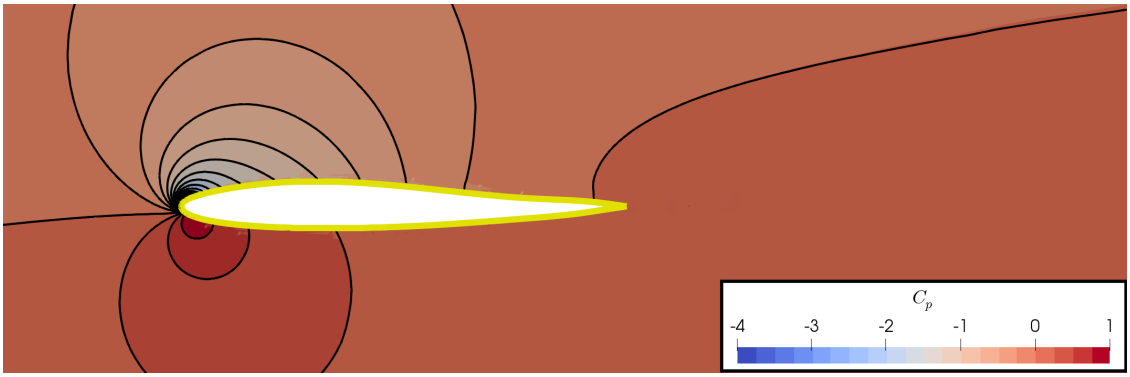
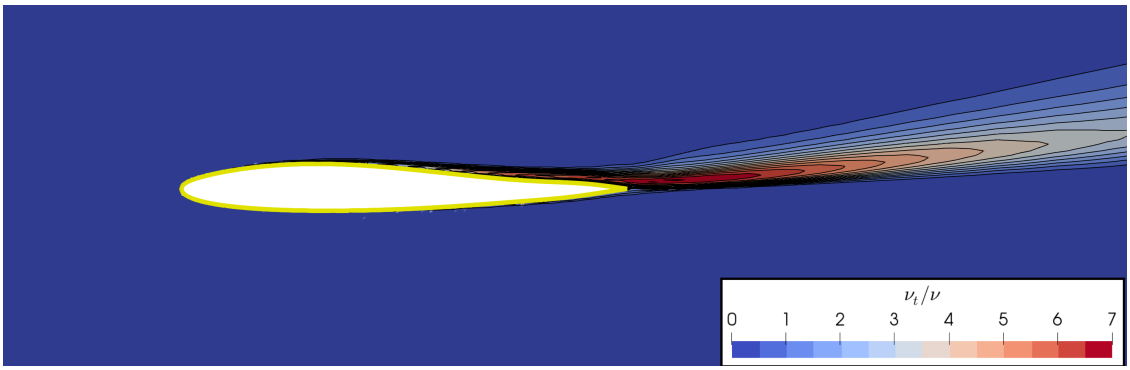


Fig. 10 Upsampled static pressure coefficient predictions from the embedded Gaussian ridge approximations, for a deformed aerofoil (from the test set) at an angle of incidence of $\alpha = 10°$. Colour contours show predictions, whilst isolines show the true CFD solutions. Upsampling is performed using a low-rank approximation of the covariance matrix for C_p, formed with the $l = 20$ leading modes.



(a) Static pressure coefficient, C_p



(b) Turbulent viscosity ratio, ν_t/ν

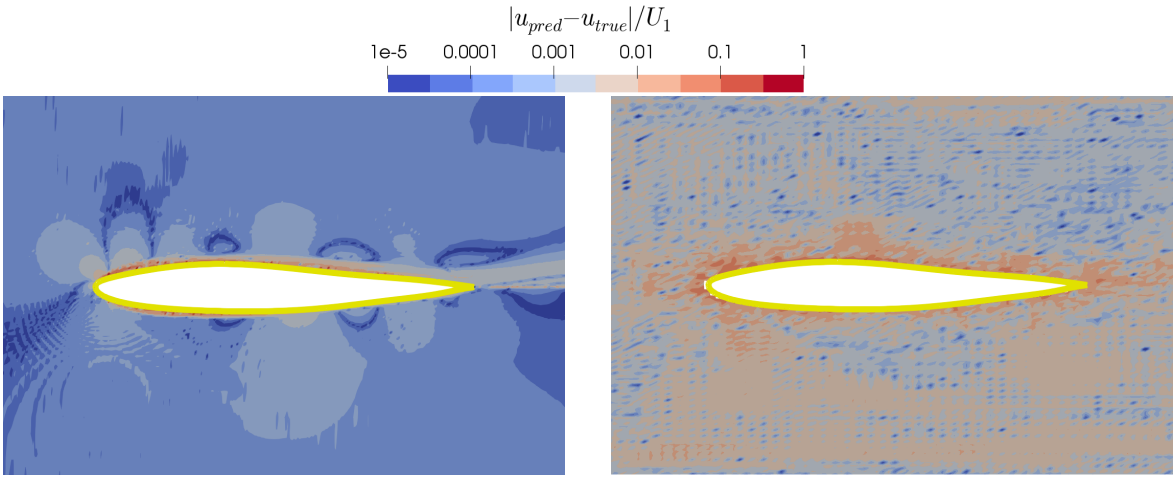
Fig. 11 Upsampled flowfield predictions from the embedded Gaussian ridge approximations, for the aerofoil shown in Figure 10, at an angle of incidence of $\alpha = 10°$. Colour contours show predictions, whilst isolines show the true CFD solutions. Upsampling is performed using the full-rank covariance matrix.

B. Accuracy of predictions

To increase the accuracy of full flowfield predictions further, one can increase the number of modes used to reconstruct the covariance matrices, bearing in mind that there is a trade-off between predictive accuracy and storage

demands. In this section, since we wish to make comparisons with the CNN approach, we choose to now use the full-rank covariance matrices. Figure 11 presents upsampled GRF predictions of C_p and ν_t/ν around an aerofoil randomly selected from the test set. The predictions for both field variables match the CFD solution closely, with only very minor errors visible near to the aerofoil surface. It is important to point out that this aerofoil belongs to the test set, so it has not been *seen* during training of the GRF's or computation of the covariance matrices.

Next, we compare the GRF framework's predictions to those given by the CNN framework outlined in Section III.C. The CNN is trained on the same $M_{train} = 1000$ data set used for the GRF's, after which it can be used to make flowfield predictions for a given design. Similarly to the GRF framework, predictions only take a number of seconds, with most of this time taken up by I/O operations. Both frameworks are used to predict the axial velocity field for a randomly selected test aerofoil at an angle of incidence of $\alpha = 10°$. Contours of the absolute error in these predictions relative to the CFD solution are shown in Figure 12. The upsampled Gaussian ridge approximations appear to be quite competitive with the CNN's predictions. The actually exhibits greater errors away from the aerofoil, whilst the GRFs' errors are slightly higher very close to the aerofoil surface.



(a) Embedded ridge approximations **(b) Convolutional neural network**

Fig. 12 Comparison of absolute error in predictions of normalised axial velocity from the embedded Gaussian ridge approximations and the convolutional neural network, for an aerofoil from the test set at $\alpha = 10°$.

For a more quantitative assessment of accuracy, the mean absolute error given by

$$MAE_\sigma = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\frac{1}{M}\sum_{m=1}^{M}\left|g_i(\mathbf{x}^{(m)}) - f_i^{(m)}\right|}{\sigma(\mathbf{f}_i)}\right] \tag{27}$$

is measured for all predictions. At each point, the absolute error is averaged for M number of designs, and it is normalised by the standard deviation of the true data $\mathbf{f}_i$ at that point. The result is then averaged over all N number of points. This is done for the M_{train} training designs and the M_{test} test designs, giving a *train error* and *test error*. Evaluating accuracy on designs unseen during training, the test designs, is necessary in order to determine if the models are being overfit to the training data. The resultant error scores are presented in Table 1. The same trends are observed at both angles of incidence; for the static pressure coefficient C_p and axial velocity u, the CNN predictions exhibit slightly lower errors. However, the upsampled GRF predictions are still competitive here. For the turbulent viscosity ratio ν_t/ν, the GRF predictions are actually superior the CNN predictions by a number of percentage points. These results are encouraging, as they suggest the GRF framework can achieve predictive accuracies which are competitive with a state-of-the-art CNN technique.

Flow Variable	$\alpha = 0°$				$\alpha = 10°$			
	Training MAE_σ		Test MAE_σ		Training MAE_σ		Test MAE_σ	
	GRF	CNN	GRF	CNN	GRF	CNN	GRF	CNN
C_p	3.2	**2.1**	3.5	**2.4**	4.0	**2.8**	4.4	**3.5**
u	3.0	**2.8**	**3.3**	3.4	4.6	**3.6**	5.1	**4.5**
ν_t	**12.5**	14.8	**15.4**	18.4	**11.1**	12.5	**14.8**	15.9

Table 1 Normalised mean absolute errors (as percentages) for the GRF and CNN predictions of three flow variables, at two different angles of incidence. The lowest errors are highlighted in bold.

C. Incorporating prediction confidence

Each Gaussian ridge function is a Gaussian process. Therefore, in addition to providing a posterior mean $\bar{g}_i$, which provides a predicted value for the field variable, each ridge function also provides a posterior variance $\mathrm{Cov}\left[g_i\right]$. Seshadri et al. [17] argue that this posterior variance serves as a heuristic for identifying the suitability of the dimension reducing subspace. To take advantage of this point, the posterior variance is propagated through the covariance matrix of each flowfield. All of the upsampled predictions previously presented in this paper were obtained with the covariance matrix $\boldsymbol{C}$ in equation (20) replaced by the modified matrix $\tilde{\boldsymbol{C}}$. This matrix is constructed by adding the ridge functions' posterior variances to their corresponding elements in the matrix

$$\tilde{\boldsymbol{C}} = \boldsymbol{C} + \mathrm{diag}\left(\mathrm{Cov}\left[\mathbf{g}\right]\right). \tag{28}$$

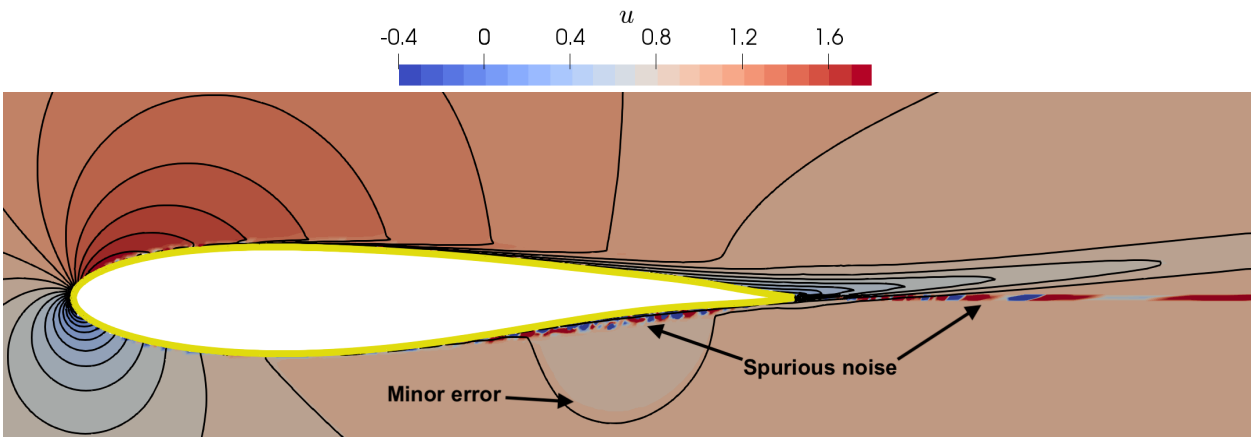
Since the inverse of $\boldsymbol{C}$ is taken in equation (20), replacing $\boldsymbol{C}$ with $\tilde{\boldsymbol{C}}$ has the effect of weighting the contribution of each Gaussian ridge function by the inverse of its posterior variance. As the posterior variance increases, the upsampled predictions at the *fine* points tends towards $\mathbb{E}\left[\mathbf{f}_f\right]$, the mean value of f at those points from the training data. This can be interpreted as the predictions falling back on *prior knowledge* when confidence in the GRF's prediction is low. At a small number of subsampled points, the GRF algorithm fails to converge, or it converges but with a poor fit, as seen by the small number of points with a low test R^2 in Figure 6. Figure 13a shows that if the unmodified $\boldsymbol{C}$ matrix is used, these occasional poor predictions result in spurious noise when they are upsampled to the remaining grid points. However, if $\tilde{\boldsymbol{C}}$ is used, as in Figure 13b, this problem is resolved. The contribution of GRF's with high posterior variances is reduced, which prevents their associated predictive errors being propagated into the upsampled flowfield predictions.

The posterior variances can also be used to provide a measure of prediction confidence to the user. In a similar fashion to how the posterior means at the subsampled points are upsampled to the remaining points via equation (20), the posterior variances can also be upsampled. The conditional covariance of $\mathbf{f}_f$ given $\mathbf{f}_c$ is the Schur complement [35] of $\tilde{\boldsymbol{C}}$ in $\boldsymbol{\Sigma}$

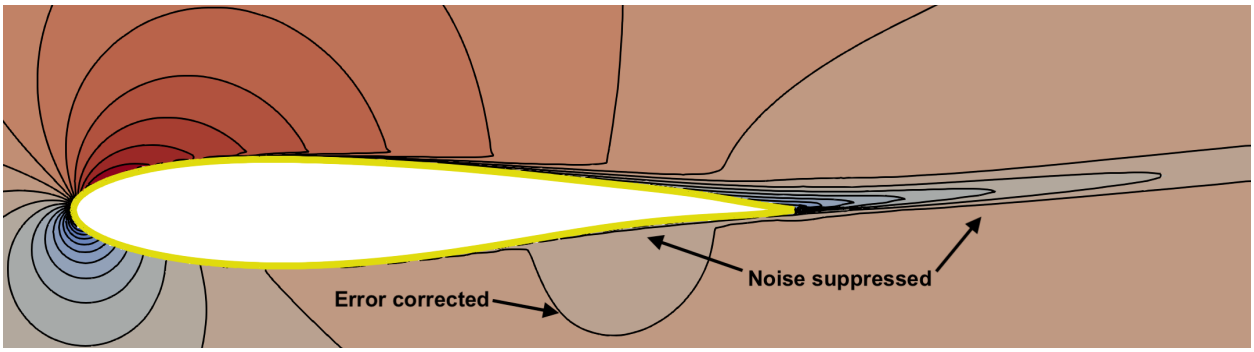
$$\mathrm{Cov}\left[\mathbf{f}_f|\mathbf{f}_c\right] = \boldsymbol{A} - \boldsymbol{B}\tilde{\boldsymbol{C}}^{-1}\boldsymbol{B}^T. \tag{29}$$

In Figure 14, this is done with the posterior variance for GRF's of axial velocity, and the resulting contours of standard deviation $\sigma_i = \sqrt{\mathrm{Cov}\left[f_i\right]}$ are plotted. This information could be invaluable to the end user, as it provides a confidence bound, informing the user where they can or cannot trust the predictions. For example, at the point labelled in Figure 14, the moderately high standard deviation (relative to the rest of the flowfield) indicates that predictions may be less reliable in this region. However, as the sufficient summary plot for this point (inset in Fig. 14) shows, the confidence bounds for the selected design are actually quite small relative to the rest of the training set, implying the predictive uncertainty is generally low for this design.

As mentioned in Section II.B, the aforementioned ideas allow for the possibility of incorporating uncertainties in the CFD data during training. The estimated aleatoric and epistemic uncertainties in the CFD data can be included in the measurement noise matrix $\sigma^2\boldsymbol{I}$ for each GRF, allowing for the CFD uncertainties to be accounted for in the GRF framework's final predictions.



(a) Without propagation



(b) With propagation

Fig. 13 Predictions of static pressure coefficient C_p, for an aerofoil from the test set at $\alpha = 10°$. The upsampled predictions are shown with and without the GRF's posterior variance propagated into the covariance matrix.

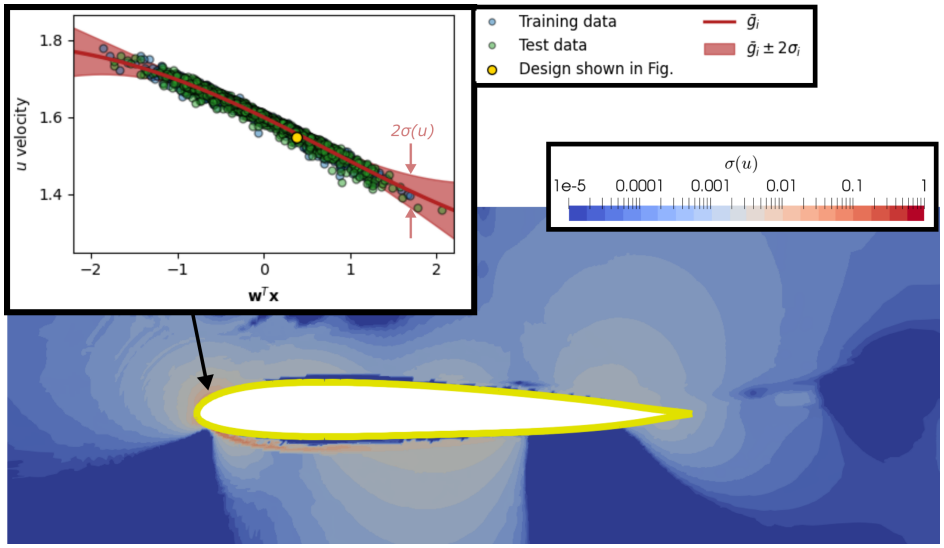


Fig. 14 Contours of posterior standard deviation of axial velocity, $\sigma(u)$, for a deformed aerofoil (from the test set) at an angle of incidence of $\alpha = 10°$. A sufficient summary plot for a node near the leading edge is shown.

V. Conclusions

This paper extends *embedded ridge functions*, first proposed by Wong et al. [15], to full flowfield predictions. The resulting data-driven framework could be trained on existing CFD data where available, or it could be integrated within a wider design of experiment, with new CFD data generated specifically for training. Once trained, the framework provides rapid flowfield predictions, which can be used for design space exploration, design optimisation tasks, and for performance predictions.

By reducing the dimensionality of the problem, embedded ridge functions allow for a high degree of model interpretability, in addition to potentially reducing the amount of training data required. Since computing embedded ridge functions for every computational grid point within a flowfield would be computationally expensive, we instead choose to obtain ridge functions at only a small number of randomly subsampled points. This paper then demonstrates how the flow physics encoded within covariance matrices, computed from the training data, can be used to upsample the ridge functions' predictions back to the rest of the flowfield. Furthermore, the independent nature of each ridge function means that their training can be viewed as an embarrassingly parallel task. This makes the embedded ridge function framework trivial to implement in a parallel fashion, allowing for excellent scaling with problem size. Additionally, the computation of the covariance matrices, and the subsequent use of them, involves a variety of linear algebra operations. This allows for efficient implementation, taking advantage of linear algebra libraries such as BLAS and LAPACK. Performing an eigendecomposition of the covariance matrices allows for a further efficiency gain. Only a small fraction of the total number of modes are shown to be necessary to capture the flow physics. This allows for low rank approximations of the covariance matrices to be used, significantly shrinking storage requirements and the cost of input/output operations.

To provide a point of reference, a state-of-the-art convolutional neural network (CNN) was also trained on the same aerofoil data. Comparing the predictive errors of both frameworks on unseen test data, the proposed embedded ridge function approach was found to be competitive with the CNN. This is encouraging, since the ridge function framework offers a number of other advantages over the CNN. Firstly, it is arguably more interpretable. Although methods do exist to interpret CNN's [14], they may not be particularly accessible for those without a degree of deep learning knowledge. On the other hand, the ridge functions enable *sufficient summary plots* to be viewed for any given point. The reduced dimensional nature of these plots lends itself to easy visualisation, allowing for easy comparison between designs, and new physical insights. Secondly, the use of Gaussian ridge functions allows for principled uncertainty quantification. Adding the ridge functions' posterior variances to the covariance matrices means the upsampled predictions return to their prior knowledge when uncertainty is high, which has the effect of removing spurious noise from upsampled predictions. The user may also visualise this uncertainty information, giving the user increased confidence in predictions. With the growing interest in uncertainty quantification methods for CFD, the possibility of assimilating the resulting uncertainties into the proposed framework could also prove invaluable.

This paper has exposed the capability of the embedded ridge function framework by using a dataset of incompressible aerofoil flows. To continue development, further investigations on more complex flows are necessary in the future. For example, flows with large regions of flow separation, compressible flows, and three dimensional flows. The inclusion of multiple boundary conditions into the input data would also be beneficial. Finally, the use of covariance matrices constructed from the training data to upsample predictions to other grid points raises the interesting prospect of such an approach could be used for other sources of data. For example, it is common for experimental measurements to only be available at a limited number of locations, but full flowfield estimates would be desirable. In such a situation, covariance matrices could be trained on a CFD dataset, and then used to approximate the full flowfield conditional on a finite number of experimental measurements.

Acknowledgments

This research was supported in part through computational resources provided by The Alan Turing Institute and with the help of a generous gift from Microsoft Corporation. The authors are 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).

References

[1] Knight, D., "Design Optimization in Computational Fluid Dynamics," *Encyclopedia of Optimization*, Springer US, 2009, pp. 666–677. https://doi.org/10.1007/978-0-387-74759-0_121.

[2] Kim, K.-Y., Samad, A., and Benini, E., *Design Optimization of Fluid Machinery: Applying Computational Fluid Dynamics and Numerical Optimization*, John Wiley & Sons, Inc, Hoboken, NJ, 2019.

[3] Zang, T. A., "Airfoil/Wing Optimization," *Encyclopedia of Aerospace Engineering*, edited by R. Blockley and W. Shyy, John Wiley & Sons, Ltd, 2010. https://doi.org/10.1002/9780470686652.eae500.

[4] Ding, F., Kareem, A., and Wan, J., "Aerodynamic Tailoring of Structures Using Computational Fluid Dynamics," *Structural Engineering International*, Vol. 29, No. 1, 2019, pp. 26–39. https://doi.org/10.1080/10168664.2018.1522936.

[5] Agostini, L., "Exploration and prediction of fluid dynamical systems using auto-encoder technology," *Physics of Fluids*, Vol. 32, No. 6, 2020, p. 067103. https://doi.org/10.1063/5.0012906, URL https://doi.org/10.1063/5.0012906.

[6] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics Informed Deep Learning (Part I): Data-driven Discovery of Nonlinear Partial Differential Equations," , No. Part I, 2017, pp. 1–22.

[7] Raissi, M., and Karniadakis, G. E., "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, Vol. 357, 2018, pp. 125–141. https://doi.org/10.1016/j.jcp.2017.11.039.

[8] Guo, X., Li, W., and Iorio, F., "Convolutional Neural Networks for Steady Flow Approximation," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016. https://doi.org/10.1145/2939672.2939738.

[9] Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S., "Prediction of aerodynamic flow fields using convolutional neural networks," *Comput. Mech.*, 2019. https://doi.org/10.1007/s00466-019-01740-0.

[10] Thuerey, N., Weißenow, K., Prantl, L., and Hu, X., "Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows," *AIAA J.*, Vol. 58, No. 1, 2020, pp. 25–36. https://doi.org/10.2514/1.j058291.

[11] Jin, X., Cheng, P., Chen, W.-L., and Li, H., "Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," *Physics of Fluids*, Vol. 30, No. 4, 2018, p. 047105. https://doi.org/10.1063/1.5024595, URL https://doi.org/10.1063/1.5024595.

[12] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D., "Weight Uncertainty in Neural Network," *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 37, edited by F. Bach and D. Blei, PMLR, Lille, France, 2015, pp. 1613–1622.

[13] Lakshminarayanan, B., Pritzel, A., and Blundell, C., "Simple and scalable predictive uncertainty estimation using deep ensembles," *Adv. Neural Inf. Process. Syst.*, 2017.

[14] Zhang, Q., Wu, Y. N., and Zhu, S., "Interpretable Convolutional Neural Networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8827–8836.

[15] Wong, C. Y., Seshadri, P., Parks, G., and Girolami, M., "Embedded Ridge Approximations," , 2019.

[16] Pinkus, A., *Ridge Functions*, Cambridge University Press, 2015. https://doi.org/10.1017/cbo9781316408124.

[17] Seshadri, P., Yuchi, S., and Parks, G. T., "Dimension Reduction via Gaussian Ridge Functions," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 7, No. 4, 2019, pp. 1301–1322. https://doi.org/10.1137/18m1168571.

[18] Constantine, P. G., *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, Vol. 2, SIAM, 2015.

[19] Cook, R. D., and Ni, L., "Sufficient dimension reduction via inverse regression: A minimum discrepancy approach," *Journal of the American Statistical Association*, Vol. 100, No. 470, 2005, pp. 410–428.

[20] Hokanson, J. M., and Constantine, P. G., "Data-driven polynomial ridge approximation using variable projection," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2018, pp. A1566–A1589.

[21] Lukaczyk, T. W., Constantine, P., Palacios, F., and Alonso, J. J., "Active subspaces for shape optimization," *10th AIAA multidisciplinary design optimization conference*, 2014, p. 1171.

[22] Seshadri, P., Shahpar, S., Constantine, P., Parks, G., and Adams, M., "Turbomachinery active subspace performance maps," *Journal of Turbomachinery*, Vol. 140, No. 4, 2018.

[23] Seshadri, P., Yuchi, S., Parks, G., and Shahpar, S., "Supporting multi-point fan design with dimension reduction," *The Aeronautical Journal*, Vol. 124, No. 1279, 2020, p. 1371–1398. https://doi.org/10.1017/aer.2020.50.

[24] Scillitoe, A. D., Ubald, B., Seshadri, P., and Shahpar, S., "Design Space Exploration Of Stagnation Temperature Probes Via Dimension Reduction," *Proc. ASME Turbo Expo*, 2020.

[25] Rasmussen, C. E., and Williams, C. K., *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[26] Smith, R. C., *Uncertainty quantification: theory, implementation, and applications*, Vol. 12, Siam, 2013.

[27] Scillitoe, A., Seshadri, P., and Girolami, M., "Uncertainty Quantification for Data-driven Turbulence Modelling with Mondrian Forests," *arXiv preprint arXiv:2003.01968*, 2020.

[28] Duraisamy, K., Iaccarino, G., and Xiao, H., "Turbulence modeling in the age of data," *Annual Review of Fluid Mechanics*, Vol. 51, 2019, pp. 357–377.

[29] Gorlé, C., Zeoli, S., Emory, M., Larsson, J., and Iaccarino, G., "Epistemic uncertainty quantification for Reynolds-averaged Navier-Stokes modeling of separated flows over streamlined surfaces," *Physics of Fluids*, Vol. 31, No. 3, 2019. https://doi.org/10.1063/1.5086341.

[30] Absil, P.-A., Mahony, R., and Sepulchre, R., *Optimization algorithms on matrix manifolds*, Princeton University Press, 2009.

[31] Diskin, B., Thomas, J. L., Rumsey, C. L., and Schwöppe, A., "Grid-Convergence of Reynolds-Averaged Navier–Stokes Solutions for Benchmark Flows in Two Dimensions," *AIAA Journal*, Vol. 54, No. 9, 2016, pp. 2563–2588. https://doi.org/10.2514/1.j054555.

[32] Hicks, R. M., and Henne, P. A., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412. https://doi.org/10.2514/3.58379.

[33] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, Vol. 54, No. 3, 2016, pp. 828–846. https://doi.org/10.2514/1.j053813.

[34] Sullivan, C., and Kaszynski, A., "PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)," *Journal of Open Source Software*, Vol. 4, No. 37, 2019, p. 1450. https://doi.org/10.21105/joss.01450.

[35] von Mises, R., *Mathematical Theory of Probability and Statistics*, Elsevier, 1964. https://doi.org/10.1016/c2013-0-12460-9.

[36] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E., "Array programming with NumPy," *Nature*, Vol. 585, No. 7825, 2020, pp. 357–362. https://doi.org/10.1038/s41586-020-2649-2, URL https://doi.org/10.1038/s41586-020-2649-2.

[37] Ronneberger, O., Fischer, P., and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Lecture Notes in Computer Science*, Springer International Publishing, 2015, pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28, URL https://doi.org/10.1007/978-3-319-24574-4_28.

[38] Huber, P. J., "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, Vol. 35, No. 1, 1964, pp. 73–101. https://doi.org/10.1214/aoms/1177703732, URL https://doi.org/10.1214/aoms/1177703732.