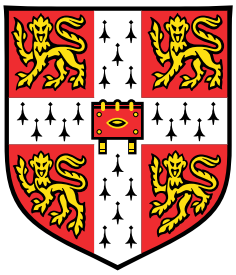
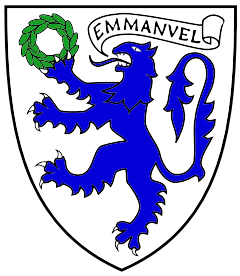


Probabilistic Operator Learning for Climate Model Parameterisation



Thomas Cowperthwaite

Supervisor: Dr. Henry Moss
Prof. Colm-Cille Caulfield

Department of Earth Sciences
University of Cambridge

4996 words

This dissertation is submitted for the degree of
Master of Research

Emmanuel College June 2024

Declaration

This report is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text and/or bibliography. This dissertation 5,000 words or fewer, including the abstract and figure captions, but excluding the front matter, references, and appendices.

Thomas Cowperthwaite
June 2024

Acknowledgements

I'm hugely grateful to Henry for his guidance and advice throughout this project, and I'm excited to see what the next few years bring — it's going to be a great journey. I'm also very thankful to the UKRI and the AI4ER team for making this project possible, with special thanks to Annabelle for her help throughout the year. Lastly I thank my friends, family, and especially my parents, for their unwavering support.

Abstract

Accurate climate simulations are essential for making predictions of the state of the physical Earth system over long timescales. A significant limitation of modern numerical climate simulations is the huge computational expense incurred when resolving phenomena that exist over small spatiotemporal scales. Modelling these processes is essential for making accurate climate predictions as they play a consequential role in the flow of heat and carbon across the planet. Calculating corrections, or parameterisations, that can be included in low-resolution simulations to capture true properties of the climate system is an active area of research. Here, we present a novel probabilistic operator learning framework which aims to learn a mesh-invariant mapping from a low-resolution climate model output to a corresponding parameterisation, and demonstrate its competitive performance on a number of benchmark problems. Our methodology is unique because it offers closed-form, principled estimates of uncertainty, and is unreliant on neural network architectures. We show that our method is competitive compared to state-of-the-art models in a low-data regime, which is particularly relevant when training data is costly to produce. This work provides a foundation for a new generation of probabilistic operator learning methods, and motivates future work on quantifying uncertainty in climate model parameterisations.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The scientific community relies on numerical climate models to make predictions of the state of the Earth system over long timescales [2]. Models which are used to inform national and international policy regarding climate change are typically comprised of coupled sub-models of the atmosphere, oceans, carbon cycle, ecosystems, ice sheets, and aerosols [40]. These coupled climate models have proved successful in predicting the Earth's future climate under prescribed anthropogenic and external forcings [21, 11].

The most important components of a climate model represent atmospheric and oceanic processes, and operate by numerically solving the partial differential equations (PDEs) which govern geophysical fluid flow [40, 37]. The physical bases for these processes are well-understood for large-scale behaviour [37], while some aspects, such as cloud microphysics, sub-mesoscale processes in ocean circulations, and fluid-structure interactions remain active areas of research [38]. The underpinnings of these models are largely well-founded in fundamental physics [37], however it remains a formidable challenge to simulate these systems numerically to a high degree of accuracy. Atmospheric and oceanic processes on a the mesoscale (10s–100s km) and smaller have a significant impact on the Earth's climate [36, 39]. Current state-of-the-art climate models are adept at representing large scale phenomena, however, while much of the smaller-scale physics has been described theoretically, computational constraints effectively limit the spatiotemporal resolutions that can be achieved when making climate predictions [2, 16, 38].

Our work aims to use gaussian processes to develop a novel machine learning framework which is able to circumvent these computational constraints by learning statistical relationships between high- and low-resolution fluid-dynamical models and leveraging them to improve predictions [42, 7, 41].

1.1 Organisation of this Thesis

- In Chapter 2, we introduce the technical background to this work, including a brief literature review.
- Chapter 3 details the methodology we employ to advance the state-of-the-art in probabilistic operator learning.
- In Chapter 4 we outline our results on a number of benchmark problems, as well as applications of our model to a realistic fluid-dynamical system.
- In Chapter 5 we place our results into the context of the broader fields of operator learning and climate model parameterisation, and evaluate some limitations of our approach.
- Finally, in Chapter 6, we outline some promising directions for future work and reiterate our contributions.
- Supplementary materials, including a FAIR data and reproducibility statement, are provided in Appendix A.

Chapter 2

Background

2.1 The Operator Learning Problem

Outside of climate modelling, many problems in engineering and applied science rely on finding solutions of (often non-linear) PDEs. As we have seen in the context of geophysical fluid dynamics, numerically obtaining accurate solutions to these equations is often computationally expensive. Operator learning is concerned with finding, in a data-driven way, a mapping (or *operator*) between a pair of functions, which often represent the initial conditions of a system and the solution of the PDE that governs it [6].

Formally, we consider a pair of separable Banach spaces $\mathcal{U}$ and $\mathcal{V}$, which represent the spaces of possible initial conditions and PDE solutions respectively [5]. The PDE that governs the system of interest is then represented by the operator

$$\mathcal{G}^\dagger : \mathcal{U} \to \mathcal{V}. \tag{2.1}$$

In an applied setting, it is not, in general, possible to observe all elements of $\mathcal{U}$ and $\mathcal{V}$, therefore the operator learning problem is that of approximating $\mathcal{G}^\dagger$ from a finite number of samples, which forms the training set for our model. The operator learning problem is therefore defined as follows:

Let $\{U_i, V_i\}_{i=1}^N$ be a set of N elements of $\mathcal{U} \times \mathcal{V}$ such that

$$V_i = \mathcal{G}^\dagger(U_i) \;\forall\; i = 1, \ldots, N. \tag{2.2}$$

A training set $\{\boldsymbol{u_i}, \boldsymbol{v_i}\}_{i=1}^N$ is generated by sampling each function $U_i \in \mathcal{U}$ at a set of locations $\{\boldsymbol{\omega_j} \in \Omega\}_{i=1}^{s_\Omega^2}$, resulting in vectors $\boldsymbol{u_i} \in \mathbb{R}^{s_\Omega^2}$ — a corresponding sample is

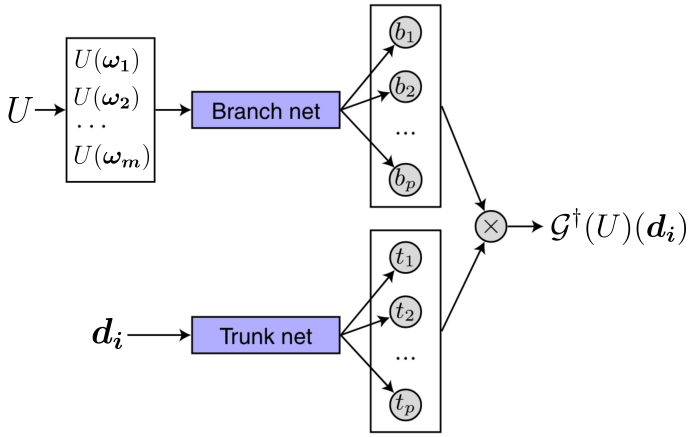


Fig. 2.1 A schematic of DeepONet, illustrating the branch and trunk networks. Adapted from [25].

taken from the target functions V_i in their domain $\mathcal{D}$. Here, we define the *resolution* in each domain as $s^2_{\Omega,\mathcal{D}}$, and $s^2_\Omega \neq s^2_\mathcal{D}$ in general.

The requirement of learning a mapping between *functions* rather than vectors makes this setup distinct from many conventional machine learning problems. As a result, a satisfactory mapping must be *mesh-invariant* and require no further training when applied to a dataset with a different resolution.

2.1.1 Literature Review

DeepONet: The seminal approach to the operator learning problem is known as the deep operator network (DeepONet), which operates by taking an entire discretised function as input, and returning the value of the output function evaluated at a single point [25, 26]. DeepONet is comprised of two deep neural networks, the "branch" net, learning a latent representation of the operator in question, while the "trunk" net encodes the output coordinates into a second latent space. These latent spaces are then combined to make a prediction (Figure 2.1). As a result of the architecture used, the model is only capable of making predictions of individual locations in $\mathcal{D}$ in a single inference step. DeepONet also relies on large training datasets, does not quantify uncertainty [4], and lacks a theoretical guarantee of convergence [12, 18]. Our work removes these limitations.

Neural Operator: More recently, a class of models known as neural operators have been developed [20]. These are an extension of discrete neural networks, where instead of each layer simply performing affine transformations on vectors, followed by an elementwise activation, each neural operator layer also performs a learnable kernel

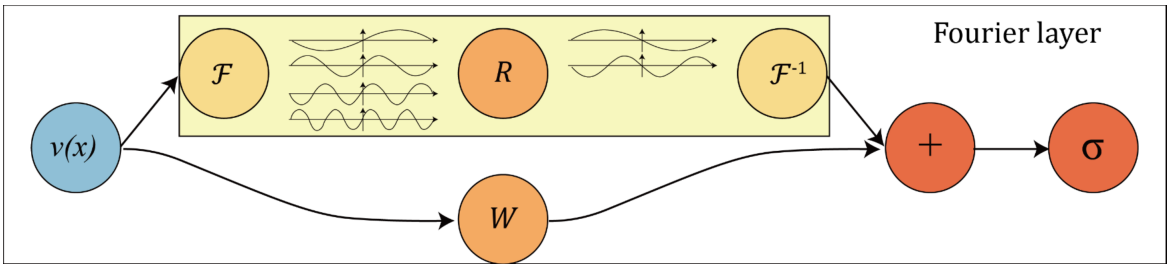


Fig. 2.2 A schematic of a single layer in a FNO model. R and W are trainable parameters, and $\mathcal{F}$ is the fourier transform. Adapted from [22].

integral transformation which acts on any discretisation of the input function. Of particular interest is the Fourier neural operator (FNO), which uses a fourier transform as its integral transformation [22]. Here, a fourier decomposition of the input function is computed, and a learnable linear map selects the most relevant fourier modes. These modes undergo an inverse transform and an activation function, before being passed to the next layer (Figure 2.2). This framework is effective at emulating PDEs, yet does not provide estimates of uncertainty, and suffers from a lack of interpretability [22].

Kernel methods: Other recent findings show that kernel regression methods are also competitive on operator learning benchmarks [5]. In this work, Batlle et al. use principle component analysis to project input functions to a low-dimensional latent space, before using kernel regression to predict pointwise evaluations of the output function. In this study, we extend this framework to include uncertainty quantification and increase prediction efficiency by additionally using PCA in the output space.

We suggest that our approach is able to match the current state-of-the-art on all major operator learning properties, while also providing uncertainty quantification, although detailed proofs are outside the scope of this work (Table 2.1).

Property	Model			
	NNs	DeepONets	Neural Operators	Ours
Discretization Invariance	✗	✗	✓	✓
Is the output a function?	✗	✓	✓	✓
Can query output at any point?	✗	✓	✓	✓
Can take input at any point?	✗	✗	✓	✓
Universal Approximation	✗	✓	✓	✓
Uncertainty Quantification	✗	✗	✗	✓

Table 2.1 Comparison of operator learning properties between models.

2.2 Gaussian Processes

A gaussian process (GP) is a set of random variables, any subset of which have a joint gaussian distribution, that can be used for non-linear regression [33]. A GP is comprised of a *prior* mean function $\mu(x)$, and a kernel function $k(x, x')$, where $x,\ x'$ are points in the domain of the GP. These functions can be used to encode any prior knowledge we have about the problem at hand. By introducing training data, a covariance matrix $\boldsymbol{K} = k(\boldsymbol{X}_{train}, \boldsymbol{X}_{train})$ is computed, and this can then be combined analytically with test data $\boldsymbol{X}_{test}$ to generate a joint distribution. This distribution is then conditioned using $\boldsymbol{Y}_{train}$ to make predictions (with uncertainty estimates) for the input test data.

2.3 Climate Model Parameterisations

We aim to assess our model's utility for improving climate model outputs. For this, we took inspiration from Perezhogin et al. [31], who used an idealised ocean model to generate a dataset suitable for calculating sub-grid scale parameterisations.

Formally, we define a potential vorticity[1] (PV) *parameterisation* as a map from a PV field (obtained from a low-resolution simulation) to a field of subgrid PV *forcing*. The PV forcing is defined as

$$S = \overline{\nabla} \cdot \overline{\boldsymbol{u}}\ \overline{q} - \overline{(\nabla \cdot \boldsymbol{u}\ q)} \tag{2.3}$$

where $\overline{(\ \cdot\ )}$ represents a coarsegrained variable, $\boldsymbol{u}$ is the fluid velocity field, and q is the PV. The PV forcing is defined as the difference between the divergence of the PV flux in a low-resolution simulation and the "ideal" PV flux divergence that could be recovered at that resolution. The latter is obtained by taking the output of a high-resolution, expensive simulation, and artificially coarse-graining the field [34]. Here, we choose to use the "sharp" coarse-graining operator focused on by Perezhogin et al. [31].

Perezhogin et al. [31] used variational autoencoders and generative adversarial networks to learn a parameterisation for a two-layer quasigeostrophic (QG) flow. They show that probabilistic machine learning techniques can learn accurate stochastic parameterisations. However, their approach is restricted to learning a vector mapping for the resolution present in the training set. Our work aims to improve upon this result by progressing towards learning a mesh-invariant operator.

[1] a standardised measure of the "twistiness" of a fluid flow. Formally is the *vorticity* ($\nabla \times \boldsymbol{u}$), normalised to the vorticity at a reference latitude. See [35, 15] for more details.

Chapter 3

Methodology

Our approach to the general operator learning problem is inspired by Batlle et al. [5], and is comprised of two machine-learning methods, principle component analysis (PCA) [17] and gaussian process (GP) regression [33]. The code used to implement our model can be found under an MIT licence at https://github.com/tomcjackc/Probabilistic-Operator-Learning-for-Climate-Model-Parameterisation. We detail our model below.

3.1 Principle Component Analysis

First, each sample point in $\boldsymbol{X}_{train} = \{\boldsymbol{u_i}\}_{i=1}^{N_{train}}$ and $\boldsymbol{Y}_{train} = \{\boldsymbol{v_i}\}_{i=1}^{N_{train}}$ is standardised across the dataset. PCA is then used to reduce the dimensionality of the vectors $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$. The variance-maximisation of PCA ensures that the relationships between the functions U_i and V_i and their domains are efficiently encoded, in the limit of $s_\Omega^2, s_\Omega^2 \to \infty$. The number of principle components (PCs) retained is n and m in each case, respectively.

3.2 Gaussian Process Regression

Using the *low-dimensional* vectors resulting from the PCA transformations, denoted as $\hat{\boldsymbol{u}}_i \in \mathbb{R}^n$ and $\hat{\boldsymbol{v}}_i \in \mathbb{R}^m$ (again standardised), we train a set of GP regression models, which aims to represent the action of the operator $\mathcal{G}^\dagger$ between the reduced-dimension latent spaces (here denoted as $f^\dagger$), as shown in Figure 3.1.

In this work, we choose to use a radial basis function (RBF) kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right). \tag{3.1}$$

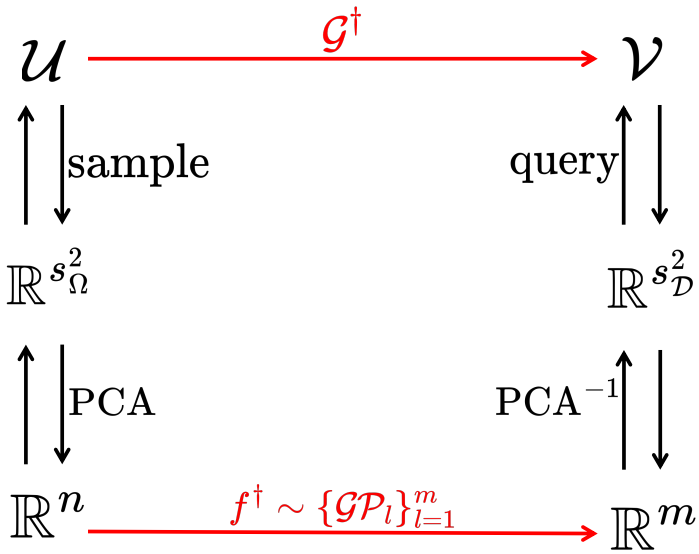


Fig. 3.1 A schematic of the model structure. Here, $f^\dagger$ acts as a latent representation of $\mathcal{G}^\dagger$. The arrows labelled PCA may be reversed using an inverse PCA transform, and sampling steps may be inverted by querying the model at any point in the domain $\mathcal{D}$.

Here, there are two kernel hyperparameters, a characteristic lengthscale l and variance σ^2. The RBF kernel encodes the loose requirement that functions represented by the GP must be infinitely-differentiable. We use a zero prior mean function, such that out-of-distribution predictions will tend to zero. We also set the observation noise for our data σ^2_{noise} as a likelihood hyperparameter, giving us three hyperparameters in total.

3.2.1 Hyperparameter Tuning and Loss Function

Optimally training each GP requires two steps. First, the posterior mean function $\boldsymbol{\mu}^*$ and covariance matrix $\boldsymbol{\Sigma}^*$ are determined analytically for a given set of hyperparameters, and the negative log marginal likelihood loss function

$$\log p(\mathbf{Y}_{train}|\mathbf{X}_{train}) = \frac{1}{2}\mathbf{Y}_{train}^\top(\mathbf{K}+\sigma^2_{noise}\mathbf{I})^{-1}\mathbf{Y}_{train} + \frac{1}{2}\log|\mathbf{K}+\sigma^2_{noise}\mathbf{I}| + \frac{N_{train}}{2}\log 2\pi \tag{3.2}$$

is calculated, where $\mathbf{K}$ is the covariance matrix of the training points. This is then minimised using a gradient-based method (here, BFGS), requiring multiple calculations of the mean function and covariance matrix.

To ensure the optimisation procedure remains stable, we enforce constraints on the three hyperparameters using a `SoftClip` bijector, with constraints: $10^{-3} < l$, $10^{-3} < \sigma^2 < 20$, and $10^{-3} < \sigma^2_{noise}$ [32].

3.3 Performance Metrics

The primary metric used to measure the performance of our model is the mean relative L^2 loss across a testing dataset. This metric is hence defined as

$$\overline{L^2 \text{ loss}} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{||\mathcal{G}^\dagger(\boldsymbol{u_i}) - \mathcal{G}(\boldsymbol{u_i})||_2}{||\mathcal{G}^\dagger(\boldsymbol{u_i})||_2} \tag{3.3}$$

where N_{test} is the number of testing examples, $\mathcal{G}^\dagger$ is the target operator, and $\mathcal{G}$ is the operator learned by the model. Each $||\cdot||_2$ is the L^2 norm.

We also use the mean coefficient of determination

$$\overline{R^2} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left(1 - \frac{\sum_{j=1}^{s^2_{\mathcal{D}}} \left(\mathcal{G}^\dagger(\boldsymbol{u_i})_j - \mathcal{G}(\boldsymbol{u_i})_j\right)^2}{\sum_{j=1}^{s^2_{\mathcal{D}}} \left(\mathcal{G}^\dagger(\boldsymbol{u_i})_j - \overline{\mathcal{G}^\dagger(\boldsymbol{u_i})}\right)}\right), \quad \overline{\mathcal{G}^\dagger(\boldsymbol{u_i})} = \frac{1}{s^2_{\mathcal{D}}} \sum_{j=1}^{s^2_{\mathcal{D}}} \mathcal{G}^\dagger(\boldsymbol{u_i})_j \tag{3.4}$$

as a measure of the fraction of variance explained in the operator output.

Finally, we use the mean negative log predictive density

$$\overline{\text{NLPD}} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left(-\sum_{j=1}^{s^2_{\mathcal{D}}} \log p\left(\mathcal{G}(\boldsymbol{u_i})_j = \mathcal{G}^\dagger(\boldsymbol{u_i})_j | \boldsymbol{u_i}\right)\right) \tag{3.5}$$

to quantify the uncertainty calibration of our model. This represents the probability of predicting the observed value for each feature, as if they were predicted independently, given the input data $\boldsymbol{u_i}$, averaged over all test examples and negated. When reporting, we normalise by a factor of $s^2_{\mathcal{D}}$ to allow comparison between problems.

3.4 Choice of GP Framework

Previously, we stated that $f^\dagger$ would be represented by a set of gaussian processes (one for each dimension of the vectors $\hat{\boldsymbol{v}}_{\boldsymbol{i}}$); there are a number of ways in which this can be achieved, and time complexity scalings are presented in Section A.2.

3.4.1 Option 1: Multi-input GPs

A clear option is to use m single-output GPs with multi-dimensional inputs, each taking the full $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ vector as an input, and responsible for predicting a single PC of the output. When used in conjunction with an Automatic Relevance Determination (ARD) kernel, the most important PCs of the input are detected by assigning them a short length scale. The component of the posterior covariance function corresponding to an irrelevant PC is assigned a very long length scale through hyperparameter tuning.

This flexible approach allows for relevant PCs of the input to be learned directly from the data. This could be important for future work on interpreting whether domain-specific principles are being leveraged by the model.

Models set up in this $\mathbb{R}^n \to \mathbb{R}$ configuration are labelled as `multiinput = True`.

3.4.2 Option 2: Single input GPs

In contrast, we consider using a set of m single-output GPs with one-dimensional inputs. Each l-th model is again responsible for predicting the l-th principle component of the output, but using only the l-th principle component of the input to make the prediction.

This approach is more computationally-efficient than the previous case, as it generally requires fewer optimisation steps in the hyperparameter tuning by virtue of the one-dimensional parameter space.

There is no guarantee *a priori* that a PC in the input will contain sufficient information for a GP to make an accurate prediction of its corresponding output component. This limitation motivated the PCA adaptation proposed in Section A.7.

Models set up in this $\mathbb{R} \to \mathbb{R}$ configuration are labelled as `multiinput = False`.

3.4.3 Option 3: Multi-output GPs

A third option is that of multi-output GPs, which would be capable of fully representing the latent operator $f^\dagger$ in a single model which captures structure across its outputs [3]. We do not make use of this setup here due to its undesirable complexity scaling, but it constitutes an interesting direction for future work.

3.5 Mesh-invariance

Once trained, our model is able to use inputs sampled at any set of locations in the input domain to make predictions at any set of locations in the output domain. We achieve this by constructing new PCA models by linearly interpolating the PCA component fields shown in Section A.3.2 at the new set of sample locations. The testing points should lie in a region spanned by training points for this to be effective, and this method can be used for both up- and down-sampling results.

Once interpolated, the pre-trained latent GP models can be used, without any further modification, to make predictions.

Chapter 4

Results

4.1 Data Sources

Data used for the benchmark experiments presented here can be found at https://doi.org/10.5281/zenodo.12529654 [10, 26], and those used for the PV parameterisation problem can be found at https://doi.org/10.5281/zenodo.7622683 [31, 30]. We defer to these references for full explanations of how the datasets were produced.

4.2 Benchmark Problems

Previous general approaches to operator learning (some of which are surveyed in Section 2.1) have been benchmarked on a number of standardised problems. Training and testing data are obtained by numerically solving a known PDE over a mesh of a given resolution. In this section, we present four benchmark problems used to compare our model to state-of-the-art approaches.

In each case shown, $\mathcal{U}$ and $\mathcal{V}$ are spaces of real-valued functions with domains $\Omega = \mathcal{D} \subset \mathbb{R}^k$ where $k = 2$ in all cases except Burgers' Equation, where $k = 1$. Final results were obtained by hand-tuning model hyperparameters (n, m, `ARD`), using training and testing datasets of 1000 and 200 instances, respectively. Results of the PCA for each problem can be found in Appendix Section A.3 and the model hyperparameters used for each result are shown in Section A.4.

4.2.1 Burgers' Equation

Burgers' equation is a convection-diffusion PDE commonly used in the study of fluid mechanics [19], non-linear acoustics [13], and traffic flow [29]. The input function U

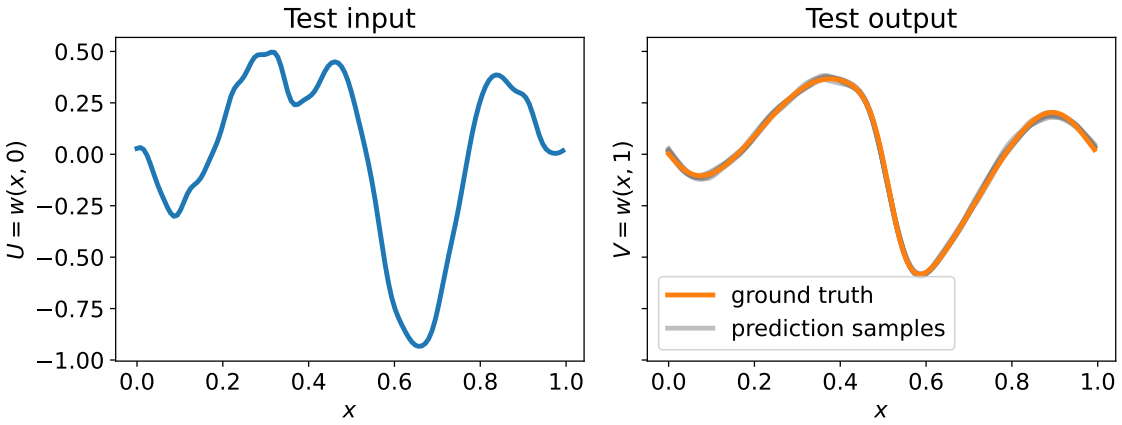


Fig. 4.1 Example of our model's test performance on one instance of the Burgers' Equation problem. The input function is shown on the **left**, and the target output function, as well as eight samples drawn from our model, are shown on the **right**.

represents an initial one-dimensional profile of some arbitrary physical quantity existing on the domain $\mathcal{D} = (0, 1)$ with periodic boundary conditions (BCs). This field is then evolved under Burgers' Equation

$$\frac{\partial w}{\partial t} + w\frac{\partial w}{\partial x} = \nu\frac{\partial^2 w}{\partial x^2}, \quad (x, t) \in (0, 1) \times (0, 1], \tag{4.1}$$

$$w(x, 0) = U(x), \quad x \in (0, 1) \tag{4.2}$$

until obtaining $V(x) = w(x, 1)$. The operator we aim to learn is therefore $\mathcal{G}^\dagger : w(x, 0) \to w(x, 1)$. Training data is comprised of different randomly-generated initial functions U_i, along with their time-evolved counterparts V_i (Figure 4.1). The constant ν is set to 0.1 in each case. As in [26], our data is discretised with resolution $s_{\mathcal{D}}^2 = 128$ across the domain (equally-spaced). Table 4.1 shows the best results obtained, and Figure 4.2 shows an example uncertainty quantification.

n	m	L^2 loss ↓	R^2 ↑	NLPD ↓
7	8	1.43%	0.9987	1.66

Table 4.1 Required n and m to capture 99% of variance in Burgers' equation input and output functions, respectively. Performance measured on a test set.

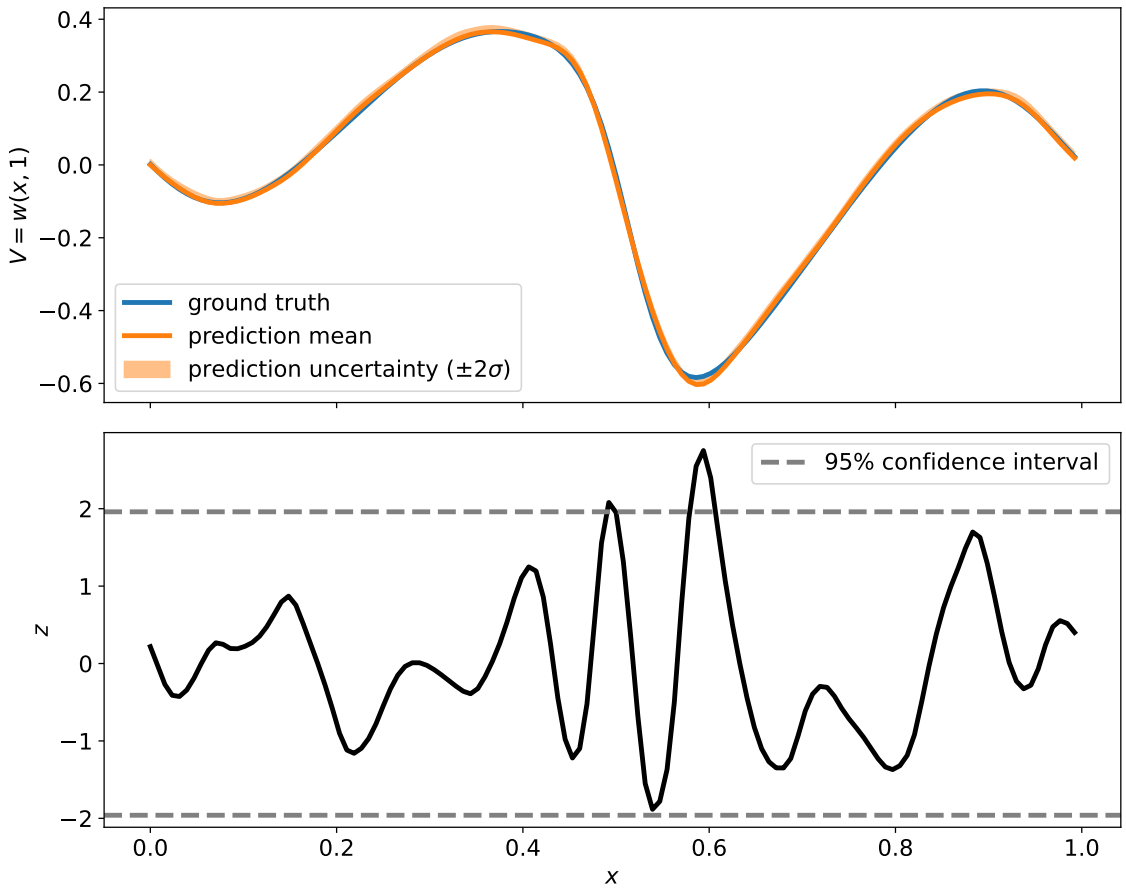


Fig. 4.2 Example of test result on the Burgers' Equation problem (**upper**), and an example of uncertainty calibration using z-scores (**lower**), we see that almost all points lie within the 95% confidence interval.

4.2.2 Darcy Flow

Darcy's law is a PDE that describes the flow of fluid through a porous medium, finding extensive applications in hydrogeology and petroleum engineering [27]. We consider the two-dimensional Darcy flow, governed by

$$\begin{aligned} -\nabla \cdot \left(e^U \nabla V\right) &= w, \quad \text{in } \mathcal{D} \\ V &= 0, \quad \text{on } \partial\mathcal{D}. \end{aligned} \tag{4.3}$$

Here, $\mathcal{D} = (0,1)^2$, U represents a permeability field, V represents the volumetric flux of a fluid in the medium, and w represents a forcing term which is fixed in our dataset. The training set was generated by randomly sampling different permeability fields and numerically solving for the "solution" V (Figure 4.3). The resolution in the training dataset is $s_{\mathcal{D}}^2 = 29 \times 29$ and the sampling points are arranged in a square lattice across the domain. Our best numerical results are presented in Table 4.2 and example test performance is shown in Figure 4.3.

n	m	L^2 loss $\downarrow$	R^2 $\uparrow$	NLPD $\downarrow$
~ 450	28	2.96%	0.9375	2.32

Table 4.2 Required n and m to capture 99% of variance in Darcy Flow input and output functions, respectively. Performance measured on a test set.

4.2.3 Helmholtz Equation

The Helmholtz Equation is the eigenvalue problem for the Laplacian operator and is therefore ubiquitous across physical science. The formulation used in this work is motivated by the study of waves, and is represented by

$$\left(-\nabla^2 - \frac{\omega^2}{U^2(x)}\right) V = 0, \quad x \in (0,1)^2 \tag{4.4}$$

with BCs

$$\frac{\partial V}{\partial n} = 0, \quad x \in \{0,1\} \times [0,1] \cup [0,1] \times \{0\} \quad \text{and} \quad \frac{\partial V}{\partial n} = v_N, \quad x \in [0,1] \times \{1\}. \tag{4.5}$$

Here, U and V represent the wave speed and excitation fields, respectively, and ω is a constant. The resolution of both the input and output domains is $s_{\mathcal{D}}^2 = 100 \times 100$.

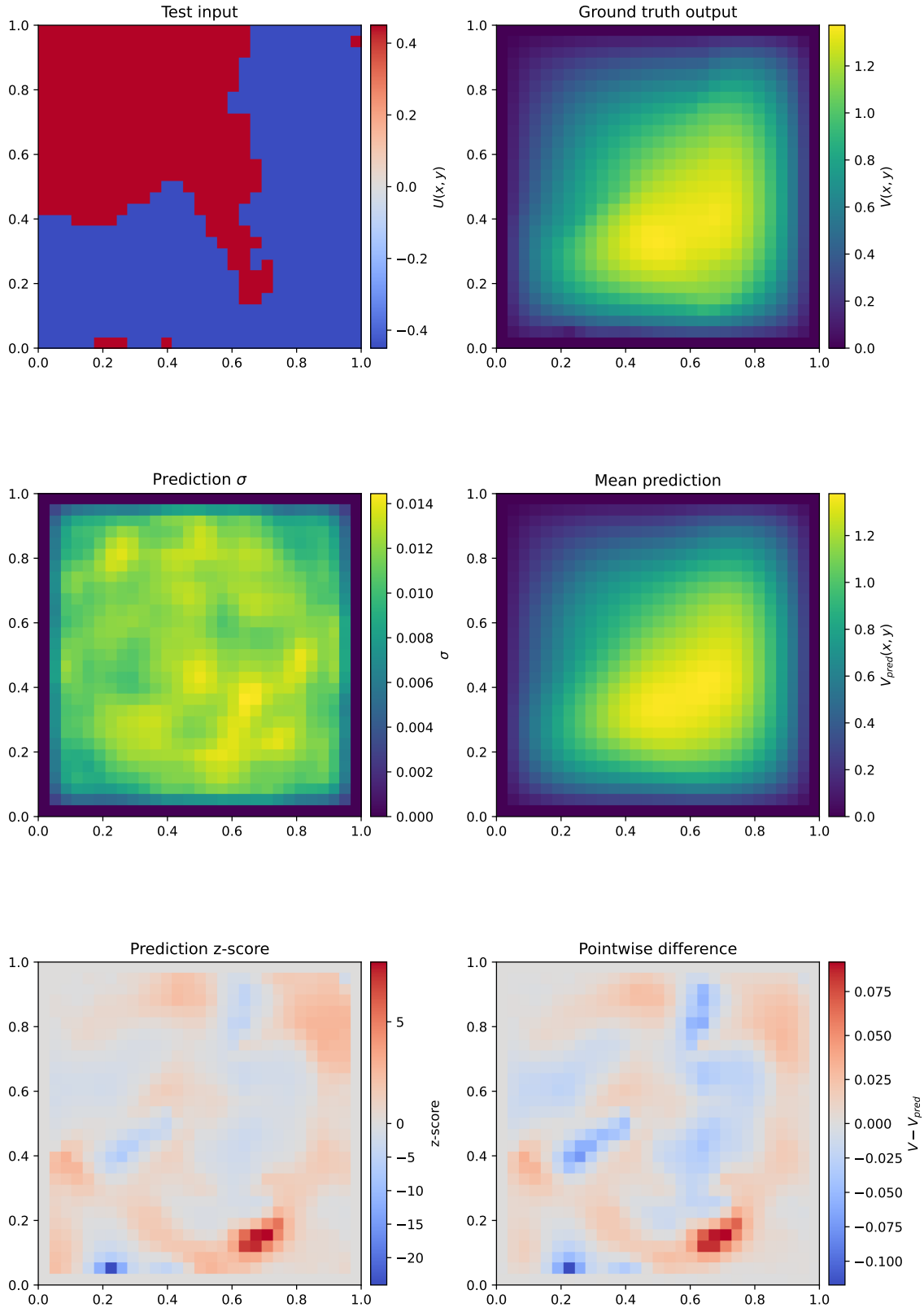


Fig. 4.3 Example of our model's test performance on the Darcy Flow problem.

An example test result is shown in Figure 4.4 and numerical results are provided in Table 4.3.

n	m	L^2 loss $\downarrow$	R^2 $\uparrow$	NLPD $\downarrow$
46	8	9.32%	0.9351	1.47

Table 4.3 Required n and m to capture 99% of variance in Helmholtz input and output functions, respectively. Performance measured on a test set.

4.2.4 Navier-Stokes Equations

The Navier-Stokes (NS) Equations are the fundamental equations governing fluid flow, making them an appropriate benchmark for our model, which ultimately aims to be applied to geophysical fluid flows in the climate system. This work uses data generated according to the vorticity-stream function formulation of the NS Equations

$$\frac{\partial \omega}{\partial t} + (c \cdot \nabla)\omega - \nu \Delta \omega = U, \quad \omega = -\Delta \psi, \quad \int_{\mathcal{D}} \psi = 0, \quad c = \left(\frac{\partial \psi}{\partial y}, -\frac{\partial \psi}{\partial x} \right) \tag{4.6}$$

where ω and ψ are the vorticity and stream function of the fluid flow, respectively. The initial vorticity field $\omega(x, y, t = 0)$ and fluid viscosity ν were held fixed. The equations were solved numerically on the domain $\mathcal{D} = (0, 2\pi)^2$, described by coordinates x and y, which was given periodic BCs. The domain was discretised to a resolution of $s_{\mathcal{D}}^2 = 64 \times 64$. Our model aims to learn the mapping from U to V, where U is the forcing term for the flow, and $V = \omega(x, y, t = T)$. An example test result is shown in Figure 4.5, along with uncertainty quantification. Our best numerical results are given in Table 4.4.

n	m	L^2 loss $\downarrow$	R^2 $\uparrow$	NLPD $\downarrow$
79	36	7.06%	0.9953	2.40

Table 4.4 Required n and m to capture 99% of variance in NS input and output functions, respectively. Performance measured on a test set.

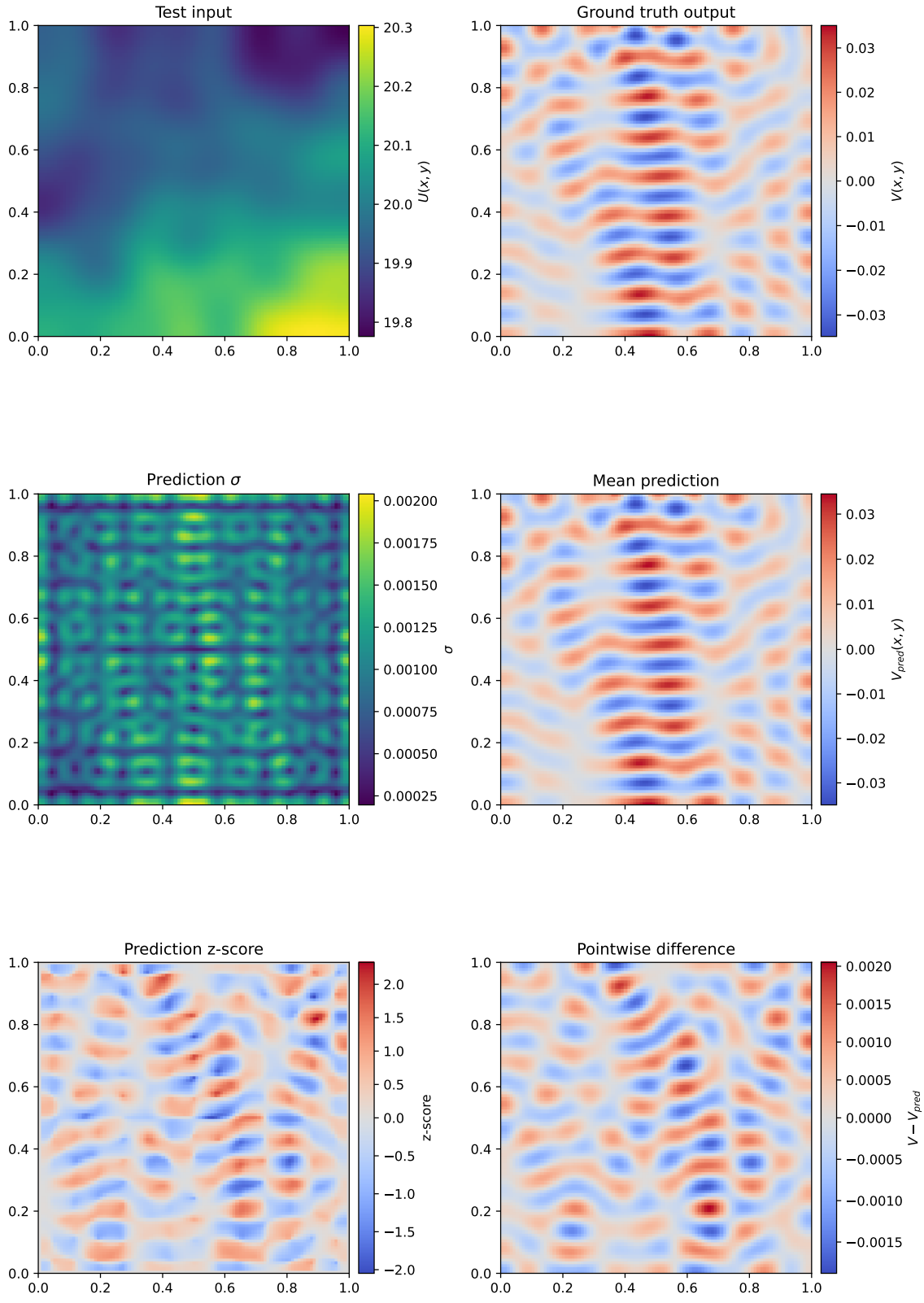


Fig. 4.4 Example of our model's test performance on the Helmholtz problem.

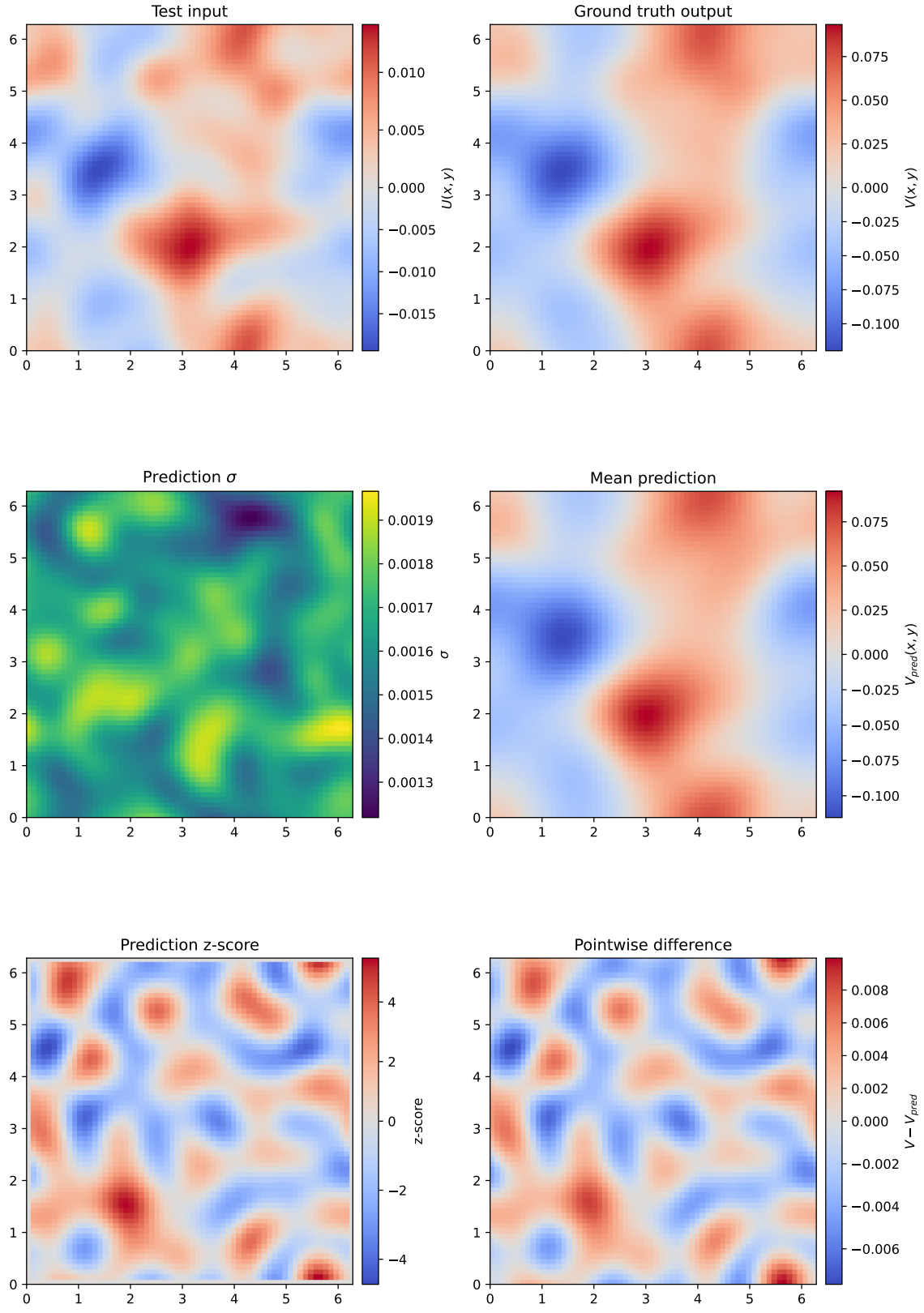


Fig. 4.5 Example of our model's test performance on the NS problem.

4.2.5 Aggregated Results

We now compare our results on each benchmark problem with six state-of-the-art operator learning frameworks: DeepONet [25], POD-DeepONet [26], FNO [22, 20], PCA-Net [14], PARA-Net [10], and the kernel regression approach from [5] — using the mean relative L^2 loss (Table 4.5). We also provide a comparison of data efficiency and uncertainty quantification across a range of N_{train} in Figure 4.6.

Model	**Burgers'**	**Darcy Flow**	**Helmholtz**	**Navier-Stokes**
DeepONet	2.15%	2.91%	18.67%	17.45%
POD-DeepONet	1.94%	**2.32%**	n/a	n/a
FNO	1.93%	2.41%	**4.29%**	**0.50**%
PCA-Net	n/a	n/a	5.95%	18.02%
PARA-Net	n/a	n/a	13.54%	44.39%
[5]	2.15%	2.75%	9.61%	0.81%
Our best result	**1.43%**	2.96%	9.36%	7.06%
Our uncertainty calibration	1.66	2.32	5.35	1.57

Table 4.5 Summary of our best results obtained on each problem, benchmarked against six state-of-the-art models. Elements marked n/a show where the relevant model has not yet been applied to the given problem. The relative L^2 test error (↓) is reported, and the best result for each problem is highlighted in **bold**. Data obtained from [10].

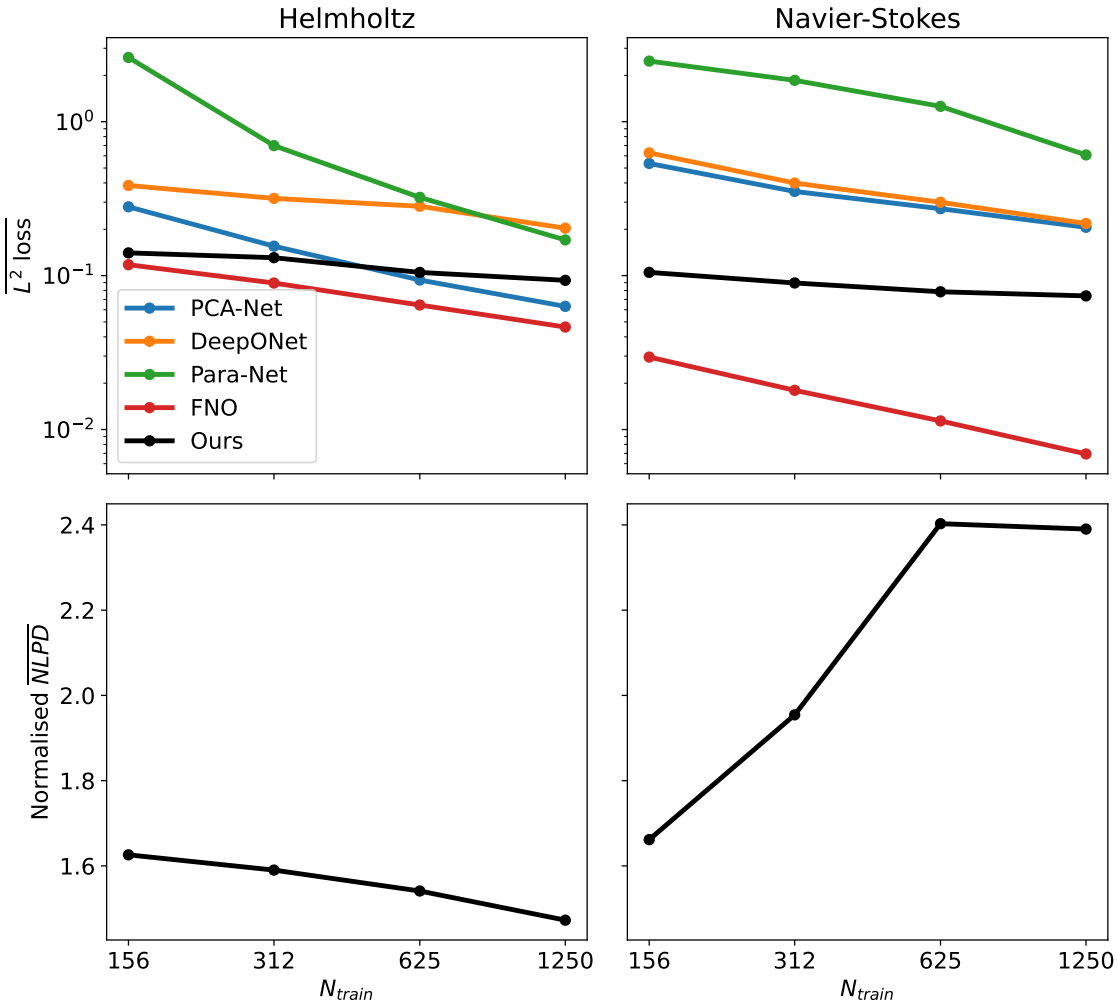


Fig. 4.6 Comparison of test loss (**top**) and uncertainty calibration (**bottom**) for a range of N_{train}, across five different operator learning models.

4.3 Parameterisation of a Quasigeostrophic Flow

Data for our PV parameterisation problem were generated by numerically solving the two-layer QG equations for PV (see Section A.5) and subtracting the PV resulting from the mean flow, to obtain a PV anomaly field for each time step.

In the context of our general operator learning problem, we denote a given discretised PV field (flattened to one dimension) as $\boldsymbol{u_i}$, and the resulting subgrid PV forcing $\boldsymbol{v_i}$. Our model aims to use these vectors to learn the mesh-invariant mapping $\mathcal{G}^\dagger : \mathcal{U} \to \mathcal{V}$. Functions U_i and V_i are defined on the same spatial domain $\mathcal{D} = (0, 10^6)^2$.

Our dataset was obtained by integrating the two-layer QG equations for 10 years at a high spatial (256×256) and temporal ($\Delta t = 1$ hr) resolution. Snapshots of the q and $\boldsymbol{u}$ fields were then taken every $1,000$ hours. We use 10 runs (860 snapshots) for training, due to computational constraints imposed by the complexity scaling of the GPs, and 5 runs for testing. Each snapshot was used to generate a training input vector $\overline{q(x, y, t = t_i, \text{run} = j)} \equiv \boldsymbol{u}_{i+86j}$, and an output vector $\overline{\nabla} \cdot \overline{\boldsymbol{u}}_{ij}\ \overline{q}_{ij} - \overline{(\nabla \cdot \boldsymbol{u}_{ij}\ q_{ij})} \equiv \boldsymbol{v}_{i+86j}$, where $i = 0, \ldots, 85$ for each $j = 0, \ldots, 9$. The $\overline{(\ \cdot\ )}$ represents a filtered and coarse-grained variable ($s^2_{\mathcal{D}} = 48 \times 48$).

Although our model was able to learn some spatial features of the output function in training (Figure 4.7), the magnitude of the PV forcing was consistently underestimated. Importantly, this performance did not generalise to the test set (Table 4.6), and we propose that a more flexible model (greater n, m, `ARD = True`) trained on more runs could improve this performance significantly. Model parameters are given in Section A.6.

n	m	L^2 loss $\downarrow$	R^2 $\uparrow$	% z-scores within 95% confidence interval
>500	>500	100.89%	-0.01792	46%

Table 4.6 Required n and m to capture 99% of variance in PV parameterisation input and output functions, respectively. Performance measured on a test set.

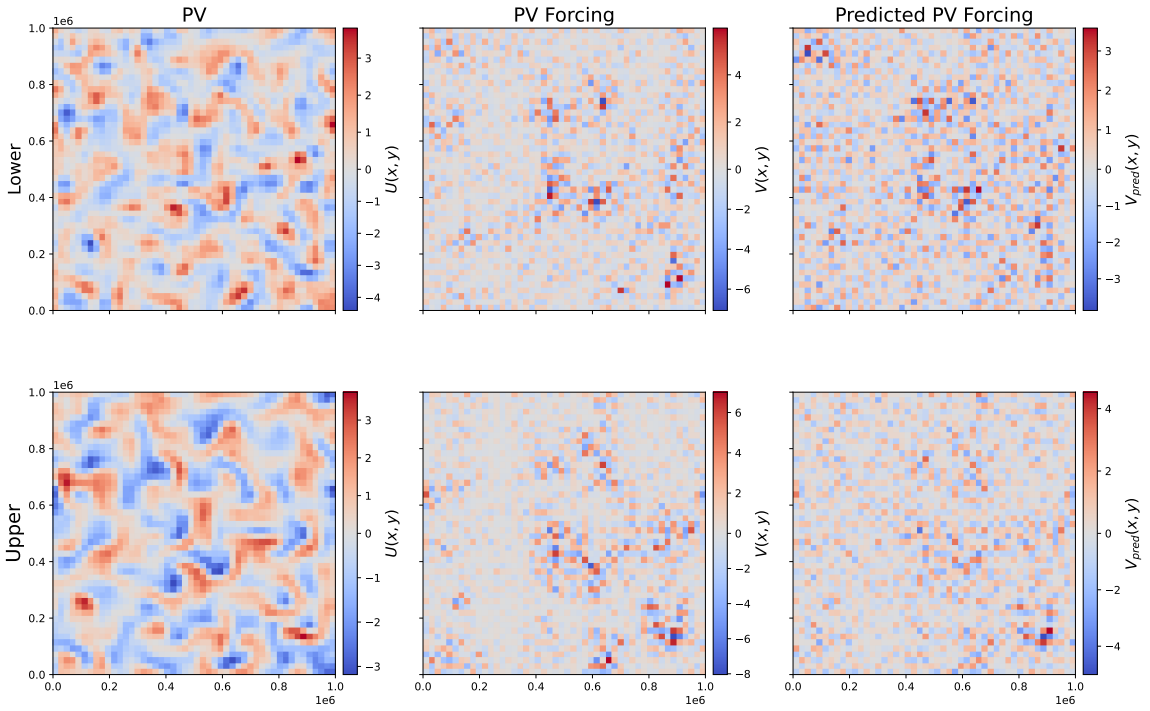


Fig. 4.7 An example of our model's **training** performance on the PV parameterisation problem.

4.4 Mesh-invariant Results

Here, we briefly demonstrate the mesh-invariance of our model using the Navier-Stokes problem as an example. In Figure 4.8, we show test results from our model which was trained at a high resolution of $s_{\mathcal{D}}^2 = 64 \times 64$. We then coarsegrained the input and output testing data to $s_{\mathcal{D}}^2 = 32 \times 32$ and made predictions using the model with no re-training. In this example, the L^2 test accuracy at the high resolution was 23.82%, and at the lower resolution there was a small drop in performance, to 26.09%.

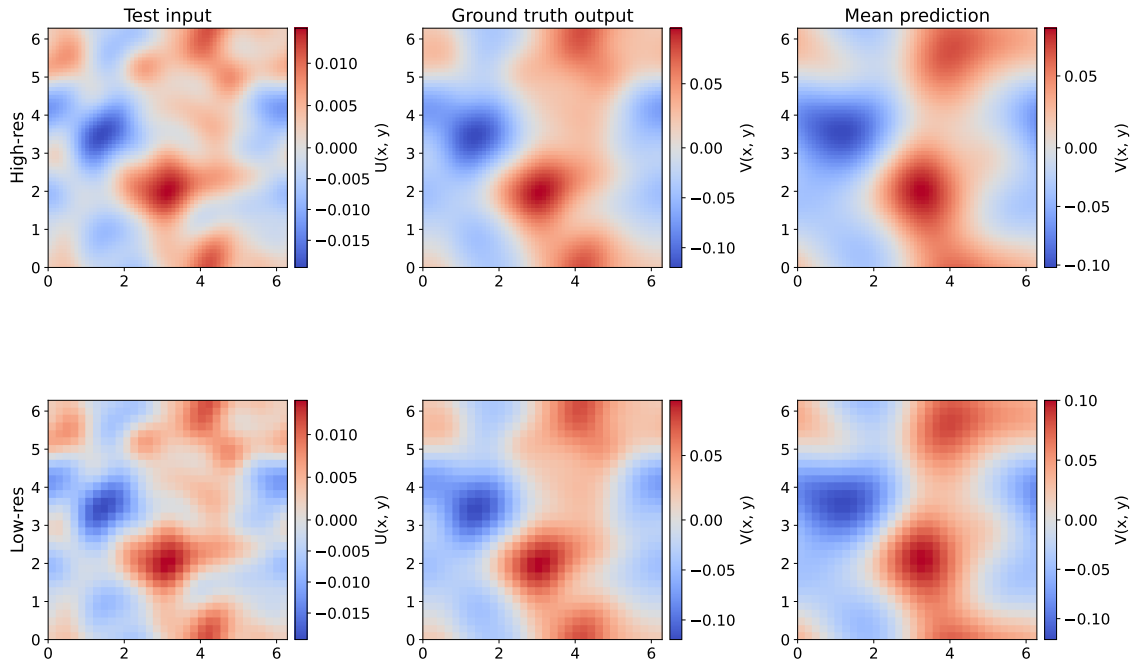


Fig. 4.8 A demonstration of the mesh-invariant property of our model. High-res: 64×64, low-res: 32×32.

Chapter 5

Discussion

5.1 Benchmark Problems

We tested our probabilistic operator learning framework on four benchmarks, each requiring the model to learn a different operator. Overall, we found that our method made competitive predictions compared to the current state-of-the-art across each of the problems tested, while providing principled estimates of uncertainty; to our knowledge, we are the first to have achieved these two goals simultaneously using a non-neural network-based approach.

On each benchmark problem we found that our model's uncertainty estimates were generally well-calibrated, with 90%, 83%, 93%, and 91% of test points lying within the 95% confidence interval for the four benchmarks, respectively. This implies slightly overconfident predictions.

Our model performs significantly better than the current state-of-the-art on the one-dimensional Burgers' problem. Our model assumes nothing about the domain from which its training data was drawn, each pixel is treated as independent, with no spatial correlations encoded *a priori*. This contrasts with the FNO, which is set up differently depending on the dimensionality D of the problem's domains (making use of the D-dimensional fourier transform), and therefore benefiting from an additional bias when compared to our model, which has to discover the space's geometry from scratch (Section A.3.2).

The PCA is inefficient at capturing variance in the input functions of the Darcy flow problem, requiring over 400 components to capture 99% of the variance (Section A.3). This inefficiency resulted in a much more complex model being required to obtain competitive results (Section A.4). As we showed, the input field is discrete and is contains many discontinuities which occur over very short length scales ($\sim$ 1 pixel).

Many PCs are required to represent these high-frequency features (Section A.3.2). We therefore conclude that vanilla PCA is more suitable for continuous-valued functions.

In contrast with the FNO, our model performed relatively poorly on the Helmholtz and NS problems, with best mean relative L^2 losses of 9.36% and 7.06%, respectively. We hypothesise that this is because in fourier space, the Laplacian operator (present in both problems) becomes diagonal and thus linear, significantly simplifying its representation in the model [8]. This is clearest in the Helmholtz case, with visible periodicity in the output functions.

5.2 Evaluating our PV Parameterisation

We also applied our model to a much more challenging problem, that of recovering PV parameterisations from a low-resolution PV field. Here, our model displayed some ability to capture the spatial structure of the PV forcing during training, and only slightly underestimated the magnitude of the forcing required. This promising performance did not generalise to the test set.

When comparing our methodology to that of Perezhogin et al. [31], we note that the largest difference in approach is that of the scale of training data used (10 vs 250 simulation runs). In Section A.2, we note that our model scales poorly with the volume of training data used, so this is a clear avenue for improving our model's performance on this more challenging problem. Numerous approaches are available for improving the scalability of GPs [23]. Alternatively, one could train the PCA and GP components of the model independently, leveraging the manageable scaling of PCA with N_{train} by training the PCA on the full dataset to ensure the optimal principle components are identified, before fitting the GP models to a smaller subset of the data to learn the latent operator.

As mentioned, there is a theoretical basis to expect fourier-based models to perform well in the context of fluid dynamics. We also note that PCA struggled to efficiently reduce the dimensionality of the input and output functions, further motivating the hypothesis that using a spectral decomposition such as a discrete fourier transform may prove to be more effective for this problem, especially when combined with the uncertainty quantification provided by the GP latent operator.

Chapter 6

Conclusions and Future Work

In this study, we pioneered a novel approach to probabilistic operator learning using purely classical (non-neural network-based) methods. We have shown (Section 4.2) that our framework is competitive with modern methods in a low-data regime, while crucially providing well-calibrated estimates of uncertainty on all benchmark problems. We also demonstrated the mesh-invariance of our setup.

Following this success, we applied our approach to a realistic fluid-dynamics problem (Section 4.3), and found that the spatial dependence of the output function led to poor performance of the PCA in that domain — hence limiting the effectiveness of our model overall, and motivating some interesting avenues for future work.

We propose that using a spectral decomposition, such as a multi-dimensional discrete fourier transform, may simplify the effect of differential operators such as the Laplacian. This will maintain the GP uncertainty quantification while offering performance improvements for the model overall.

It would be interesting to find ways of coupling the two dimensionality-reduction steps together, whether it be PCA or otherwise, in order to optimally train the latent GP models. Neural network methods achieve this through backpropagation, however, there is limited literature on coupled PCA methods. We consider developing a hybrid method between PCA and canonical-correlation analysis in Section A.7 and further work on this may yield improvements [1, 24].

More broadly, we believe our framework will ultimately find utility in providing probabilistic subgrid parameterisations for fluid flow, and future work could work towards interpreting model predictions, leveraging recent developments in equation discovery [28, 43, 42].

References

[1] (2007). Canonical Correlation Analysis. In Härdle, W. and Simar, L., editors, *Applied Multivariate Statistical Analysis*, pages 321–330. Springer, Berlin, Heidelberg.

[2] Alizadeh, O. (2022). Advances and challenges in climate modeling. *Climatic Change*, 170(1):18.

[3] Álvarez, M. A. and Lawrence, N. D. (2011). Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12(41):1459–1500.

[4] Ayhan, M. S. and Berens, P. (2022). Test-time Data Augmentation for Estimation of Heteroscedastic Aleatoric Uncertainty in Deep Neural Networks.

[5] Batlle, P., Darcy, M., Hosseini, B., and Owhadi, H. (2024). Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496:112549.

[6] Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. M. (2021). Model Reduction And Neural Networks For Parametric PDEs. *The SMAI Journal of computational mathematics*, 7:121–157.

[7] Bolton, T. and Zanna, L. (2019). Applications of Deep Learning to Ocean Data Inference and Subgrid Parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1):376–399. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2018MS001472.

[8] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (1988). Spectral Approximation. In Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., editors, *Spectral Methods in Fluid Dynamics*, pages 31–75. Springer, Berlin, Heidelberg.

[9] de Hoop, M. and Huang (2022). The cost-accuracy trade-off in operator learning with neural networks.

[10] de Hoop, M. V., Huang, D. Z., Null, E. Q., and Stuart, A. M. (2022). The Cost-Accuracy Trade-Off in Operator Learning with Neural Networks. *Journal of Machine Learning*, 1(3):299–341.

[11] Fan, X., Miao, C., Duan, Q., Shen, C., and Wu, Y. (2020). The Performance of CMIP6 Versus CMIP5 in Simulating Temperature Extremes Over the Global Land Surface. *Journal of Geophysical Research: Atmospheres*, 125(18):e2020JD033031. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020JD033031.

[12] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[13] Hamilton, M. F. and Blackstock, D. T. (1998). *Nonlinear Acoustics*. Academic Press.

[14] Hesthaven, J. S. and Ubbiali, S. (2018). Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78. ADS Bibcode: 2018JCoPh.363...55H.

[15] Holton, J. R. and Hakim, G. J. (2013). Chapter 4 - Circulation, Vorticity, and Potential Vorticity. In Holton, J. R. and Hakim, G. J., editors, *An Introduction to Dynamic Meteorology (Fifth Edition)*, pages 95–125. Academic Press, Boston.

[16] Jakob, C. (2010). Accelerating Progress in Global Atmospheric Model Development through Improved Parameterizations: Challenges, Opportunities, and Strategies. *Bulletin of the American Meteorological Society*, 91(7):869–876.

[17] Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202. Publisher: Royal Society.

[18] Kawaguchi, K. and Sun, Q. (2021). A Recipe for Global Convergence Guarantee in Deep Neural Networks. arXiv:2104.05785 [cs, math, stat].

[19] Kim, Y. J. and Tzavaras, A. E. (2001). Diffusive N-Waves and Metastability in the Burgers Equation. *SIAM Journal on Mathematical Analysis*, 33(3):607–633.

[20] Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2023). Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97.

[21] Lee, J.-Y., Marotzke, J., Bala, G., Cao, L., Corti, S., Dunne, J. P., Engelbrecht, F., Fischer, E., Fyfe, J. C., Jones, C., Maycock, A., Mutemi, J., Ndiaye, O., Panickal, S., and Zhou, T. (2021). Future global climate: scenario-based projections and near-term information. In Masson-Delmotte, V., Zhai, P., Pirani, A., Connors, S. L., Péan, C., Berger, S., Caud, N., Chen, Y., Goldfarb, L., Gomis, M. I., Huang, M., Leitzell, K., Lonnoy, E., Matthews, J. B. R., Maycock, T. K., Waterfield, T., Yelekçi, , Yu, R., and Zhou, B., editors, *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, pages 553–672. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

[22] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:2010.08895 [cs, math].

[23] Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2020). When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423.

[24] Lock, E. F., Hoadley, K. A., Marron, J. S., and Nobel, A. B. (2013). Joint and individual variation explained (JIVE) for integrated analysis of multiple data types. *The Annals of Applied Statistics*, 7(1):523–542. Publisher: Institute of Mathematical Statistics.

[25] Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229. Publisher: Nature Publishing Group.

[26] Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. (2022). A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778.

[27] Martin, F. D. and Colpitts, R. M. (1996). Reservoir Engineering. In Lyons, W. C., editor, *Standard Handbook of Petroleum and Natural Gas Engineering*, pages 1–362. Gulf Professional Publishing, Houston.

[28] Meng, Y. and Qiu, Y. (2024). Sparse discovery of differential equations based on multi-fidelity Gaussian process. arXiv:2401.11825 [cs, math].

[29] Musha, T. and Higuchi, H. (1978). Traffic Current Fluctuation and the Burgers Equation. *Japanese Journal of Applied Physics*, 17(5):811. Publisher: IOP Publishing.

[30] Perezhogin, P. (2023). Dataset for paper Pavel Perezhogin, Laure Zanna, Carlos Fernandez-Granda "Generative data-driven approaches for stochastic subgrid parameterizations in an idealized ocean model" submitted to JAMES.

[31] Perezhogin, P., Zanna, L., and Fernandez-Granda, C. (2023). Generative Data-Driven Approaches for Stochastic Subgrid Parameterizations in an Idealized Ocean Model. *Journal of Advances in Modeling Earth Systems*, 15(10):e2023MS003681. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2023MS003681.

[32] Pinder, T. and Dodd, D. (2022). Gpjax: A gaussian process framework in jax. *Journal of Open Source Software*, 7(75):4455.

[33] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass. OCLC: ocm61285753.

[34] Ross, A., Li, Z., Perezhogin, P., Fernandez-Granda, C., and Zanna, L. (2023). Benchmarking of Machine Learning Ocean Subgrid Parameterizations in an Idealized Model. *Journal of Advances in Modeling Earth Systems*, 15(1):e2022MS003258. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2022MS003258.

[35] Rossby, C.-G. (1940). Planetary flow patterns in the atmosphere. *Q. J. R. Meteorol. Soc.*, 66:68–87. Num Pages: 87 OCLC: 679175492.

[36] Seo, H., O'Neill, L. W., Bourassa, M. A., Czaja, A., Drushka, K., Edson, J. B., Fox-Kemper, B., Frenger, I., Gille, S. T., Kirtman, B. P., Minobe, S., Pendergrass, A. G., Renault, L., Roberts, M. J., Schneider, N., Small, R. J., Stoffelen, A., and Wang, Q. (2023). Ocean Mesoscale and Frontal-Scale Ocean–Atmosphere Interactions and Influence on Large-Scale Climate: A Review. *Journal of Climate*, 36:1981–2013. Section: Journal of Climate.

[37] Smith, D. M., Scaife, A. A., and Kirtman, B. P. (2012). What is the current state of scientific knowledge with regard to seasonal and decadal forecasting? *Environmental Research Letters*, 7(1):015602.

[38] Stensrud, D. J. (2007). *Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models*. Cambridge University Press, Cambridge.

[39] Stevens, B. and Bony, S. (2013). What Are Climate Models Missing? *Science*, 340(6136):1053–1054.

[40] Troccoli, A. (2010). Seasonal climate forecasting. *Meteorological Applications*, 17(3):251–268. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/met.184.

[41] Warner, T. T. (2010). *Numerical Weather and Climate Prediction*. Cambridge University Press, Cambridge.

[42] Zanna, L. and Bolton, T. (2020). Data-Driven Equation Discovery of Ocean Mesoscale Closures. *Geophysical Research Letters*, 47(17):e2020GL088376. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020GL088376.

[43] Zhu, Y.-C., Gardner, P., Wagg, D. J., Barthorpe, R. J., Cross, E. J., and Fuentes, R. (2022). Robust equation discovery considering model discrepancy: A sparse Bayesian and Gaussian process approach. *Mechanical Systems and Signal Processing*, 168:108717.

Appendix A

Supplementary Material

A.1 FAIR Data and Reproducibility

Complete, runnable code required to reproduce our results may be found at https://github.com/tomcjackc/Probabilistic-Operator-Learning-for-Climate-Model-Parameterisation. A subset of the data sources below was used for our benchmarking experiments, which may be found here https://doi.org/10.5281/zenodo.12529654. PV parameterisation training data generated by Perezhogin et al. [31] was accessed through a Globus data store: https://g-402b19.00888.8540.data.globus.org (immediate download), and we recommend using the code above to use this data. All attribution for data generation remains with the original authors.

Data used for the Burgers' and Darcy flow problems were obtained from https://tinyurl.com/burgersdarcy, and those for the Helmholtz and Navier-Stokes problems were obtained from https://data.caltech.edu/records/fp3ds-kej20 [9, 10]. Data generated to conduct the PV parameterisation experiments can be obtained from https://zenodo.org/records/7622683 [30, 31]. All data is publicly available.

A.2 Complexity

An important factor to consider is the computational complexity of our model. In Table A.1, we compare the time complexity scaling for a single evaluation with five different models, excluding any hyperparameter tuning.

Our model scales competitively with the resolution of the output, and scales better than the neural network-based models with respect the their hyperparameters. Neural

Architecture	Evaluation Cost scaling
PCA-Net	$\mathcal{O}\left(s_{\mathcal{D}}^2 + w^2\right)$
DeepONet	$\mathcal{O}\left(s_{\mathcal{D}}^2 + w^2\right)$
PARA-Net	$\mathcal{O}\left(s_{\mathcal{D}}^2 w^2\right)$
FNO	$\mathcal{O}\left(d_f s_{\mathcal{D}}^2 \log\left(s_{\mathcal{D}}^2\right) + s_{\mathcal{D}}^2 d_f^2\right)$
[5]	$\mathcal{O}\left(s_\Omega^2 n + s_{\mathcal{D}}^2\left(N_{train} s_\Omega^2 + N_{train}^2\right)\right)$
Our model (Op. 1)	$\mathcal{O}\left(s_\Omega^2 n + m\left(n N_{train} + N_{train}^2\right) + s_{\mathcal{D}}^2 m\right)$
Our model (Op. 2)	$\mathcal{O}\left(s_\Omega^2 n + m\left(N_{train} + N_{train}^2\right) + s_{\mathcal{D}}^2 m\right)$
Our model (Op. 3)	$\mathcal{O}\left(s_\Omega^2 n + \left(m^3 N_{train} + N_{train}^2\right) + s_{\mathcal{D}}^2 m\right)$

Table A.1 Evaluation cost scaling of different architectures. Here, w is the width of the neural network and d_f is the number of features used in the FNO.

networks scale quadratically with the width of the layers, while our model scales linearly with the number of PCs, and thus with the flexibility of the model.

As our model calculates the covariance of each test point with every training point to determine prediction uncertainty, our evaluation cost scales quadratically with the number of training points. For problems where data curation is computationally expensive, this is not expected to be a significant issue, although numerous approaches may be taken to mitigate this poor scaling [23].

A.3 Benchmark Problem PCA Results

A.3.1 Determining good values for n and m

Here we provide the results of principle component analysis (PCA) conducted on each benchmark problem presented in Section 4.2. Figure A.1 shows the cumulative variance explained by the PCA in each case. A subset of components are shown for clarity, and the **red dashed** and **solid** lines indicate 95% and 99% variance explained, respectively.

A.3.2 Principle Components

In Figures A.2, A.3, A.4, and A.5 we show visual representations of a selection of principle components for each problem. **Colours** (**vertical coordinate** for Burgers' Equation) represent the value of the weight vector at that sampling location in the given principle component, and thus indicate important "directions" in the high-dimensional space that maximise variance across the training dataset. In each case, we see that with successive principle components, increasingly high-frequency features of the functions U and V are being captured by this dimensionality-reduction.

A.3.3 Predictive skill in the Latent Space

We accessed the predictions being made in the latent space (displayed in Figures A.6, A.7, and A.8) to understand where accuracy was being lost in overall model predictions.

A.4 Model Parameters for Benchmark Problems

Model hyperparameters required to reproduce results shown in Section 4.2.5 are provided in Table A.2.

Problem	n	m	`ARD`	`multiinput`	`standardise`	`n_test`	`n_train`
Burgers'	10	50	`True`	`True`	`False`	200	1000
Darcy	50	50	`True`	`True`	`True`	200	1000
Helmholtz	20	20	`False`	`True`	`True`	200	1000
NS	50	50	`False`	`True`	`True`	200	1000

Table A.2 Model parameters used to obtain our best benchmark results, presented in Section 4.2.5. `ARD` determines whether an ARD kernel is used for the GP regressors, `multiinput` toggles whether the overall model operates in the multi-input configuration, as outlined in Section 3.4, `standardise` determines whether principle components are transformed to have a mean of 0 and standard deviation of 1. `n_test` and `n_train` denote the sizes of the test and train set, respectively.

A.5 Two-layer Quasigeostrophic Equations

The two-layer QG equations are given by

$$\partial_t q_m + \nabla \cdot (\mathbf{u}_m q_m) + \beta_m \partial_x \psi_m + U_m \partial_x q_m = -\delta_{m,2} r_{ek} \nabla^2 \psi_m + \text{ssd} \circ q_m, \tag{A.1}$$

$$q_m = \nabla^2 \psi_m + (-1)^m \frac{f_0^2}{g' H_m}(\psi_1 - \psi_2), \quad m \in \{1, 2\} \tag{A.2}$$

where q_m is the PV in layer m, where $m = 1,\ 2$ indicates the upper and lower layers, respectively. Here, ∇ denotes the horizontal gradient operator, $\boldsymbol{u}_m$ is the velocity field, ψ_m is the stream function, U_m is the prescribed mean flow in the x (zonal) direction, δ is the Kronecker delta, r_{ek} is the surface drag coefficient, and $ssd \circ q_m$ represents the effect of small-scale energy dissipation, which is prescribed to suppress numerical noise in the simulation. The meridional (y direction) gradient of PV due to simulated differential rotation and the prescribed mean flow is given as $\beta_m = \beta + (-1)^{m+1} \frac{f_0^2}{g' H_m}(U_1 - U_2)$ for a constant β (the β-plane approximation of fluid dynamics). Here, f_0 is the coriolis parameter at the latitude of interest, g' is the reduced gravity, and $H = H_1 + H_2$ is the vertical thickness of the entire system.

A.6 Model Parameters for PV Parameterisation

Model hyperparameters required to reproduce results shown in Section 4.3 are provided in Table A.3.

n	m	`ARD`	`multiinput`	`standardise`	`n_test`	`n_train`
50	50	`False`	`True`	`True`	430 (5 runs)	860 (10 runs)

Table A.3 Model parameters used to obtain our best results on the PV parameterisation problem, presented in Section 4.3. `ARD` determines whether an ARD kernel is used for the GP regressors, `multiinput` toggles whether the overall model operates in the multi-input configuration, as outlined in Section 3.4, `standardise` determines whether principle components are transformed to have a mean of 0 and standard deviation of 1. `n_test` and `n_train` denote the sizes of the test and train set, respectively.

A.7 Joint PCA

Principle component analysis is used to identify orthogonal directions in a high-dimensional space, maximising the variance explained in a given dataset which is represented by X, an $N \times d$ matrix. Here, we provide an overview of the standard PCA procedure, before outlining the approach we have taken to developing a *joint* PCA.

In the case of the first principle component, the problem reduces to finding

$$\boldsymbol{w^1} = \underset{||\boldsymbol{w}||=1}{\arg\max}\, \mathrm{Var}\,[t] = \underset{||\boldsymbol{w}||=1}{\arg\max} \sum_{i=0}^{N} t_i^2 = \underset{||\boldsymbol{w}||=1}{\arg\max} \sum_{i=0}^{N} (\boldsymbol{X_i} \cdot \boldsymbol{w})^2 \tag{A.3}$$

where $\boldsymbol{w^1}$ is the weight vector representing the d-dimensional direction in which variance in the dataset is maximised. The projection of sample i onto a weight vector $\boldsymbol{w}$ is shown as $t_i = (\boldsymbol{X_i} \cdot \boldsymbol{w})$.

Identifying this expression with its matrix form and using the constraint $||\boldsymbol{w}|| = 1$, we obtain

$$\boldsymbol{w^1} = \underset{||\boldsymbol{w}||=1}{\arg\max}\, ||X\boldsymbol{w}||^2 = \underset{||\boldsymbol{w}||=1}{\arg\max}\, \boldsymbol{w}^T X^T X \boldsymbol{w} = \underset{||\boldsymbol{w}||=1}{\arg\max}\, \frac{\boldsymbol{w}^T X^T X \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{w}} \tag{A.4}$$

which satisfies the form for a Rayleigh quotient and thus may be solved analytically. The $\boldsymbol{w}$ that maximises the quotient is the eigenvector which corresponds to the largest eigenvalue of $X^T X$.

We propose a framework which simultaneously reduced the dimensionality of two datasets $X_1,\ X_2$ with the equal numbers of samples. We do not however, require $d^1 = d^2$ (superscripts denoting dataset index). Our procedure should simultaneously capture the variance of the two datasets individually, while also maximising a measure of their joint variation — we choose to use the covariance. We therefore set up the problem as follows:

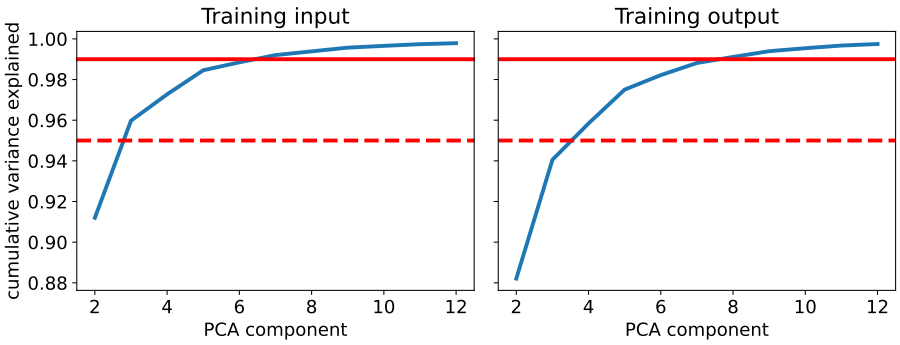
$$\boldsymbol{w_1^1}, \boldsymbol{w_2^1} = \underset{||\boldsymbol{w^1}||=||\boldsymbol{w^2}||=1}{\arg\max} \{\mathrm{Var}\,[t_1] + \mathrm{Var}\,[t_2] + 2\rho\ \mathrm{Cov}\,(t_1, t_2)\} \tag{A.5}$$

with ρ acting as a tunable parameter to determine the weighting placed on individual variance capture or covariance capture. The case $\rho = 0$ recovers the two individual PCA models.

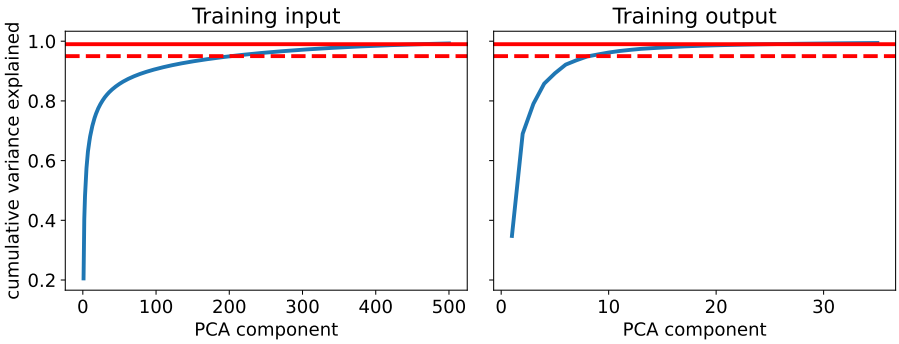
In matrix form, Equation A.5 becomes

$$\boldsymbol{w_1^1}, \boldsymbol{w_2^1} = \underset{||\boldsymbol{w_1}||=||\boldsymbol{w_2}||=1}{\arg\max} \left\{ \boldsymbol{w_1}^T X_1^T X_1 \boldsymbol{w_1} + \boldsymbol{w_2}^T X_2^T X_2 \boldsymbol{w_2} + 2\rho \boldsymbol{w_1}^T X_1^T X_2 \boldsymbol{w_2} \right\} \tag{A.6}$$

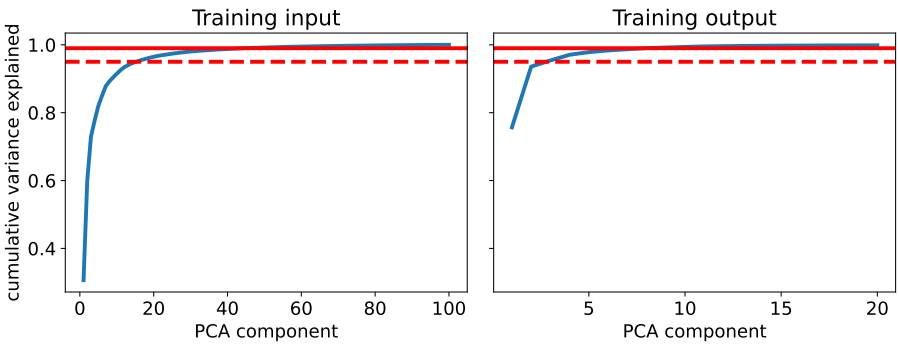
which is a constrained joint optimisation problem. An alternating optimisation scheme could be considered, first fixing $\boldsymbol{w_1}$ and optimising $\boldsymbol{w_2}$, then doing the reverse, repeating until convergence.



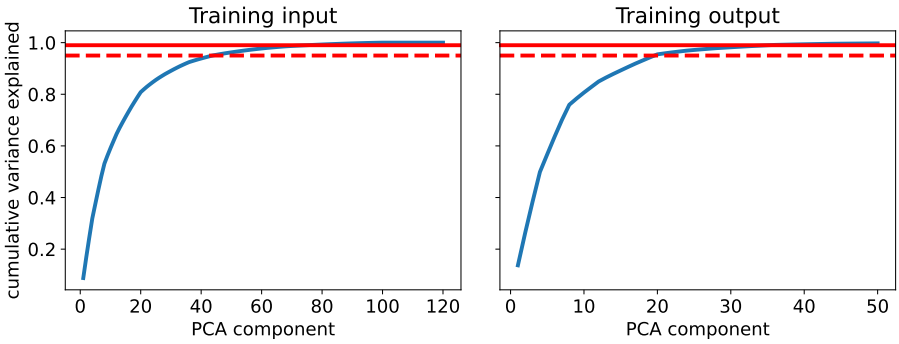
(a) Burgers' Equation



(b) Darcy Flow



(c) Helmholtz Equation



(d) Navier-Stokes Equations

Fig. A.1 Figure showing the cumulative variance explained by PCA for the input (**left**) and output (**right**) training datasets for each of the four benchmark problems.

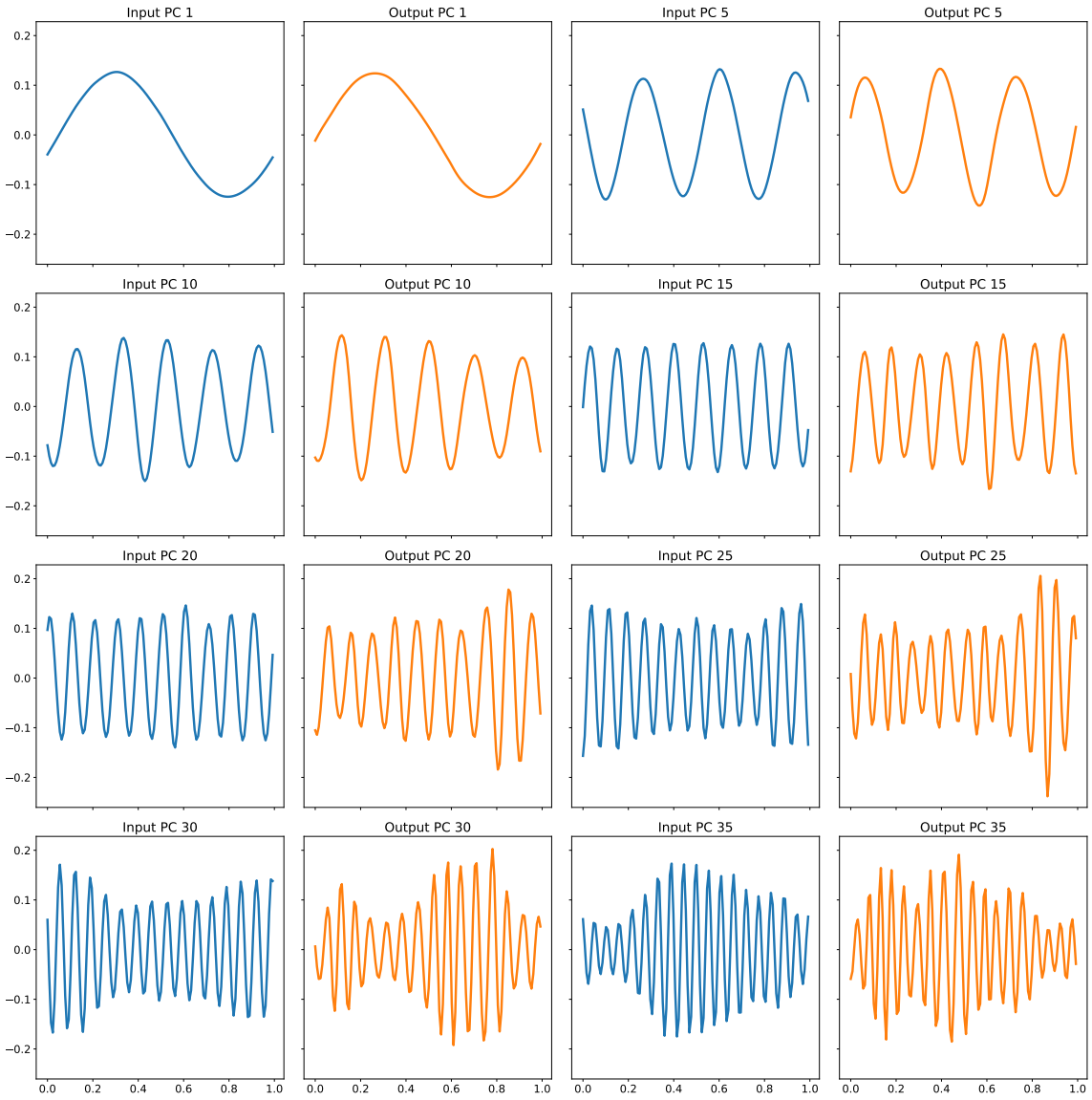


Fig. A.2 A subset of principle components identified by the PCA algorithm for the Burgers' Equation dataset. Components identified for the input functions are shown in **blue** and those for the output functions are shown in **orange**.

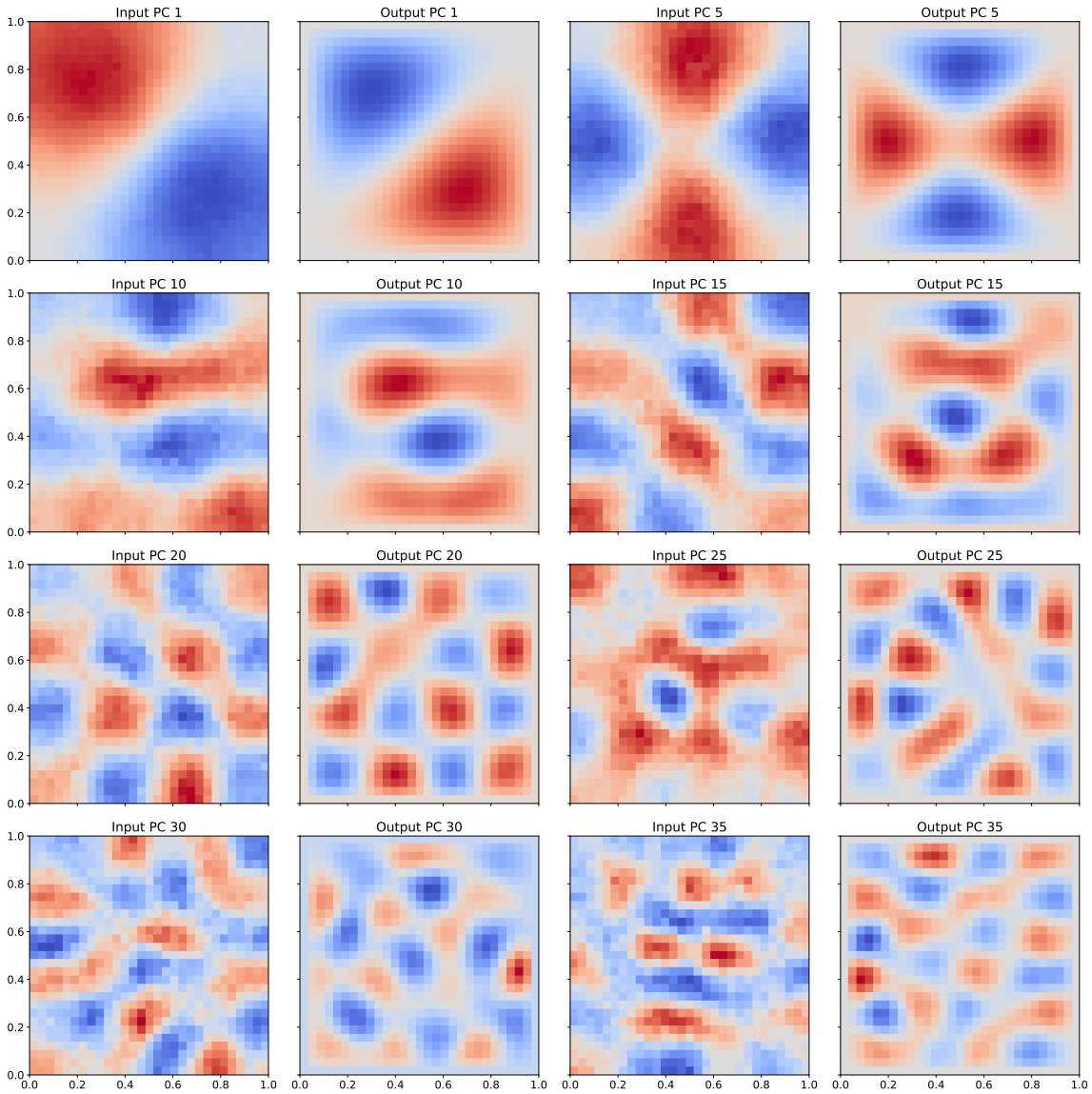


Fig. A.3 A subset of principle components identified by the PCA algorithm for the Darcy flow dataset. Components identified for the input functions are shown in columns 1 and 3, while corresponding components for output functions are in columns 2 and 4.

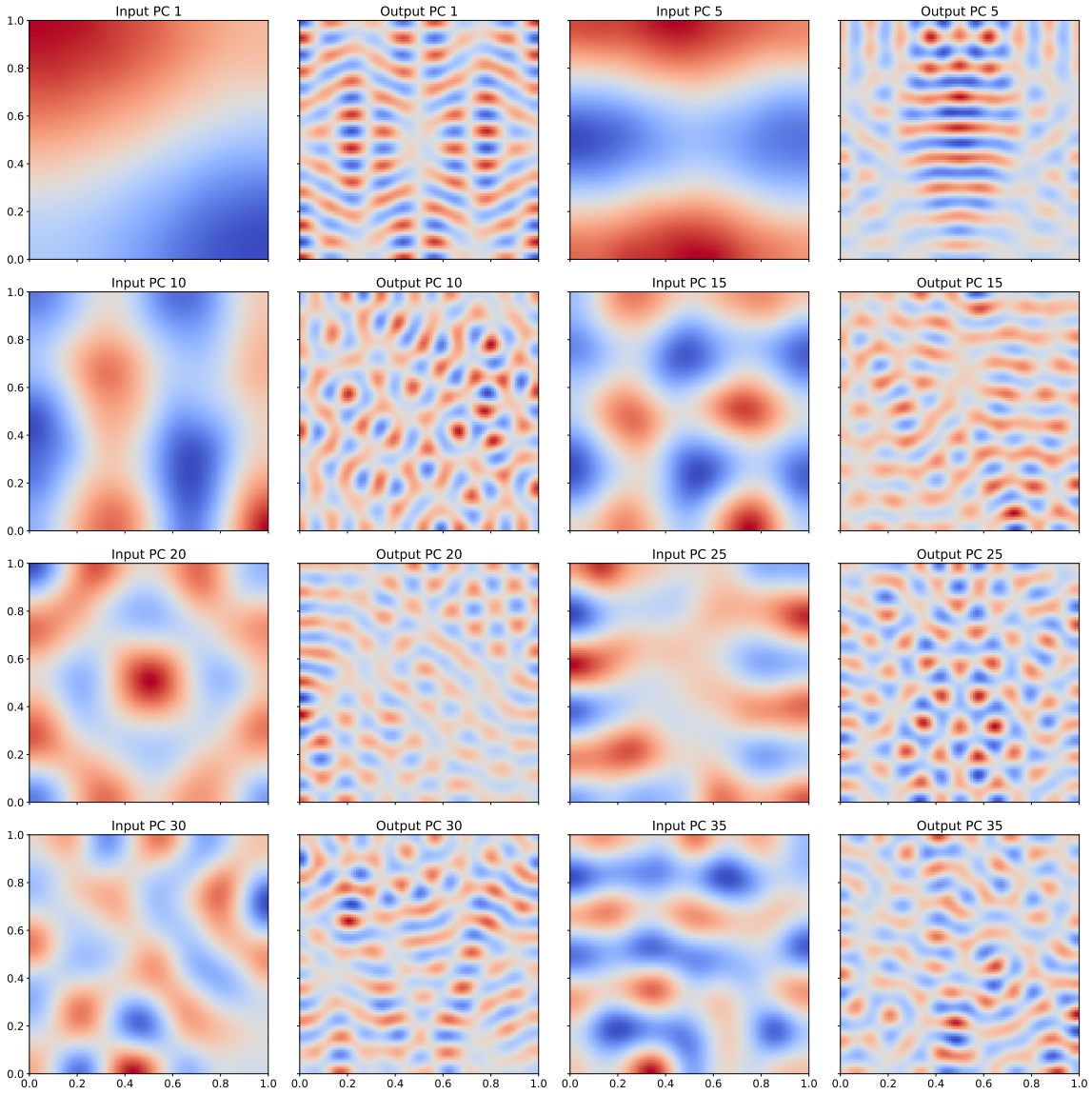


Fig. A.4 A subset of principle components identified by the PCA algorithm for the Helmholtz Equation dataset. Components identified for the input functions are shown in columns 1 and 3, while corresponding components for output functions are in columns 2 and 4.

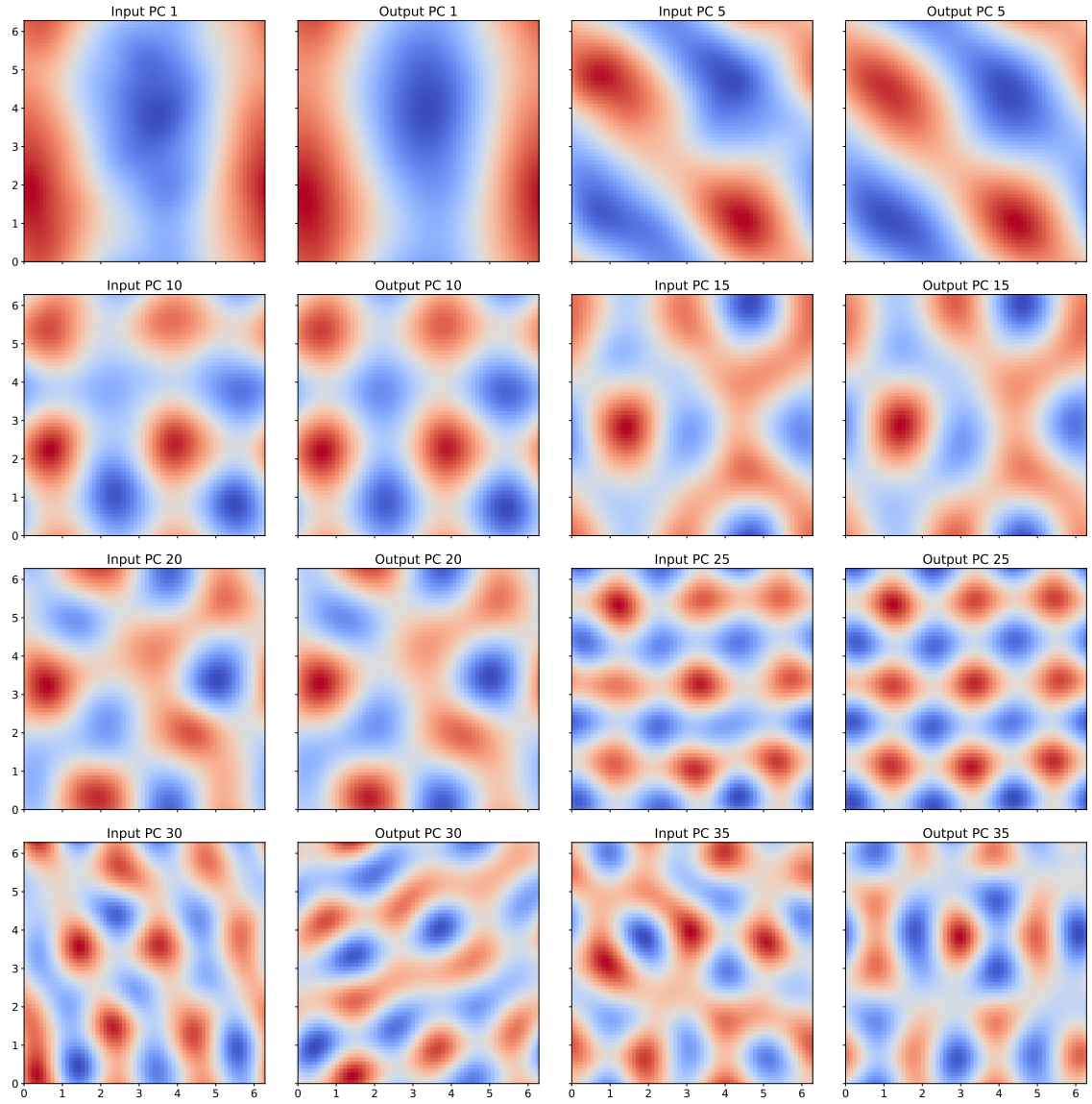


Fig. A.5 A subset of principle components identified by the PCA algorithm for the Navier-Stokes Equations dataset. Components identified for the input functions are shown in columns 1 and 3, while corresponding components for output functions are in columns 2 and 4.

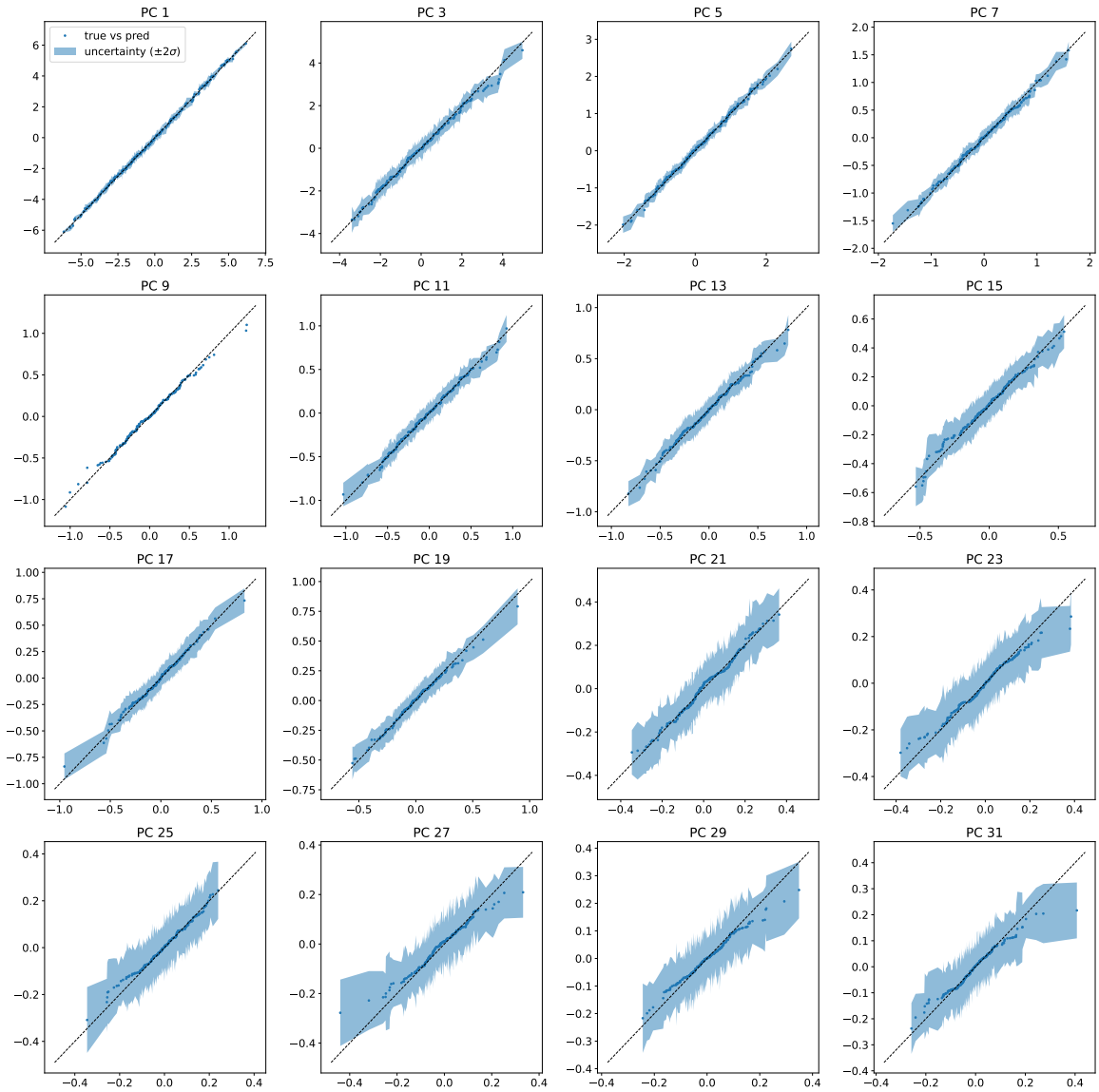


Fig. A.6 Darcy Flow latent space predictions. Plots of predicted (y-axis) against true (x-axis) principle component for a subset of principle components. Uncertainty ($\pm 2\sigma$) shown as shaded region.

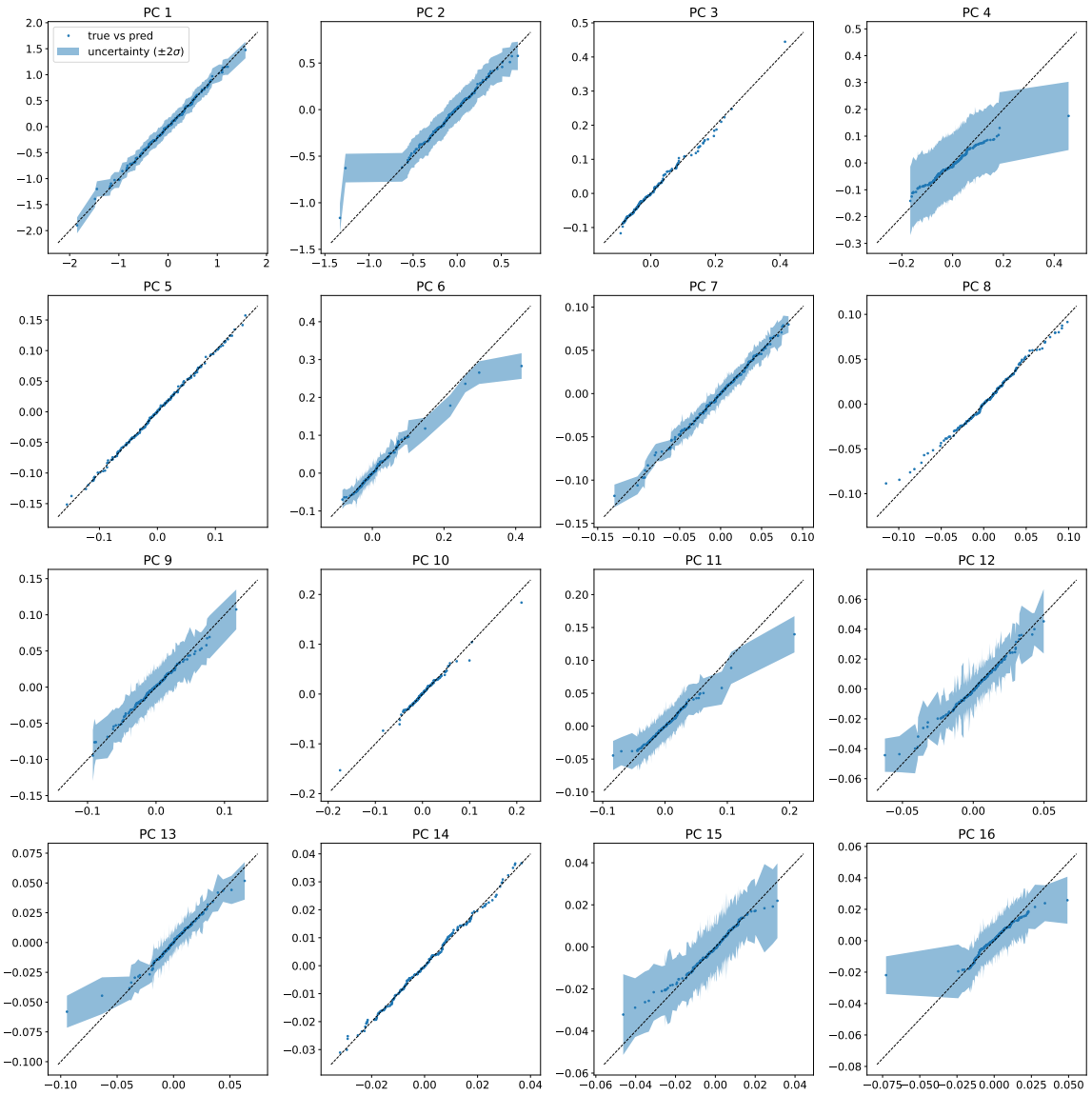


Fig. A.7 Helmholtz Equation latent space predictions. Plots of predicted (y-axis) against true (x-axis) principle component for a subset of principle components. Uncertainty ($\pm 2\sigma$) shown as shaded region.

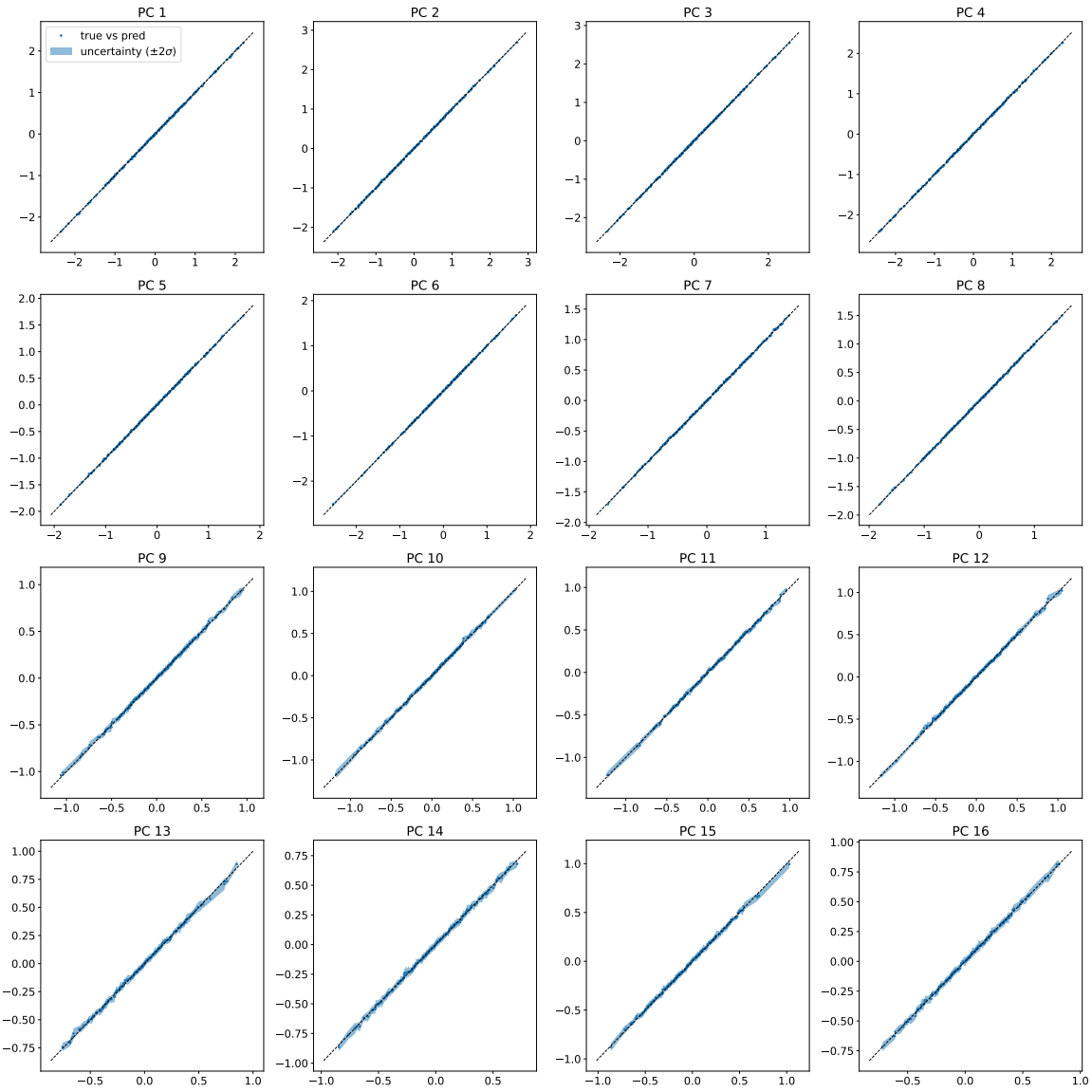


Fig. A.8 Navier-Stokes Equations latent space predictions. Plots of predicted (y-axis) against true (x-axis) principle component for a subset of principle components. Uncertainty ($\pm 2\sigma$) shown as shaded region.