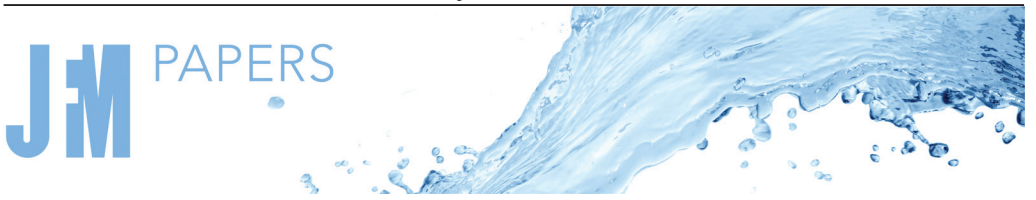
J. Fluid Mech. (2023), *vol*. 973, A42, doi:10.1017/jfm.2023.720



Dynamics of a data-driven low-dimensional model of turbulent minimal Couette flow

Alec J. Linot[1] **and Michael D. Graham**[1],†

[1]Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA

(Received 12 January 2023; revised 18 August 2023; accepted 26 August 2023)

Because the Navier–Stokes equations are dissipative, the long-time dynamics of a flow in state space are expected to collapse onto a manifold whose dimension may be much lower than the dimension required for a resolved simulation. On this manifold, the state of the system can be exactly described in a coordinate system parameterising the manifold. Describing the system in this low-dimensional coordinate system allows for much faster simulations and analysis. We show, for turbulent Couette flow, that this description of the dynamics is possible using a data-driven manifold dynamics modelling method. This approach consists of an autoencoder to find a low-dimensional manifold coordinate system and a set of ordinary differential equations defined by a neural network. Specifically, we apply this method to minimal flow unit turbulent plane Couette flow at $Re = 400$, where a fully resolved solutions requires $O(10^5)$ degrees of freedom. Using only data from this simulation we build models with fewer than 20 degrees of freedom that quantitatively capture key characteristics of the flow, including the streak breakdown and regeneration cycle. At short times, the models track the true trajectory for multiple Lyapunov times and, at long times, the models capture the Reynolds stress and the energy balance. For comparison, we show that the models outperform POD-Galerkin models with $\sim$2000 degrees of freedom. Finally, we compute unstable periodic orbits from the models. Many of these closely resemble previously computed orbits for the full system; in addition, we find nine orbits that correspond to previously unknown solutions in the full system.

Key words: machine learning

1. Introduction

A major challenge in dealing with chaotic fluid flows, whether it be performing experiments, running simulations or interpreting the results, is the high-dimensional nature of the state. Even for simulations in the smallest domains that sustain turbulence

† Email address for correspondence: mdgraham@wisc.edu

(a minimal flow unit (MFU)), the state dimension may be $O(10^5)$ (Jiménez & Moin 1991; Hamilton, Kim & Waleffe 1995). However, despite this nominal high-dimensionality, the dissipative nature of turbulent flows leads to the expectation that long-time dynamics collapse onto an invariant manifold of much lower dimension than the ambient dimension (Hopf 1948). By modelling the dynamics in a manifold coordinate system, simulations could be performed with a drastically lower-dimensional state representation, significantly speeding up computations. In addition, such a low-dimensional state representation is highly useful for downstream tasks such as control or design. Finding a low-dimensional, or ideally a minimal-dimensional, parameterisation of the manifold and an evolution equation for this parameterisation are both challenges. In this work we aim to address these challenges with a data-driven model, specifically for the task of reconstructing turbulent plane Couette flow (PCF).

The classic way to perform dimension reduction from data is to use the proper orthogonal decomposition (POD), also known as principal component analysis (PCA) or Karhunen–Loève decomposition (Holmes *et al.* 2012). This is a linear dimension reduction technique in which the state is projected onto the set of orthogonal modes that capture the maximum variance in the data. The POD is widely used for flow phenomena, some examples of which include turbulent channel flow (Moin & Moser 1989; Ball, Sirovich & Keefe 1991), flat-plate boundary layers (Rempfer & Fasel 1994) and free shear jet flows (Arndt, Long & Glauser 1997). Smith, Moehlis & Holmes (2005) showed how to incorporate system symmetries into the POD modes, the details of which we elaborate on in § 3.

Although the POD has seen wide use and is easy to interpret, more accurate reconstruction can be achieved with nonlinear methods, a result we highlight in § 3. Some popular methods for nonlinear dimension reduction include kernel PCA (Schölkopf, Smola & Müller 1998), diffusion maps (Coifman *et al.* 2005), local linear embedding (LLE) (Roweis & Saul 2000), isometric feature mapping (Isomap) (Tenenbaum, de Silva & Langford 2000) and t-distributed stochastic neighbour embedding (tSNE) (Hinton & Roweis 2003). These methods are described in more detail in Linot & Graham (2022), and an overview of other dimension reduction methods can be found in Van Der Maaten, Postma & Van Den Herik (2009). One drawback of all of these methods, however, is that they reduce the dimension, but do not immediately provide a means to move from a low-dimensional state back to the full state. A popular dimension reduction method without these complications is the undercomplete autoencoder (Hinton & Salakhutdinov 2006), which uses a neural network (NN) to map the input data into a lower-dimensional 'latent space' and another NN to map back to the original state space. We describe this structure in more detail in § 2. Some examples where autoencoders have been used for flow systems include flow around a cylinder (Murata, Fukami & Fukagata 2020), flow around a flat plate (Nair & Goza 2020), Kolmogorov flow (Page, Brenner & Kerswell 2021; Pérez De Jesús & Graham 2023) and channel flow (Milano & Koumoutsakos 2002). Although we will not pursue this approach in the present work, it may be advantageous for multiple reasons to parametrise the manifold with overlapping local representations, as done in Floryan & Graham (2022).

After reducing the dimension, the time evolution for the dynamics can be approximated from the equations of motion or in a completely data-driven manner. The classical method is to perform a Galerkin projection wherein the equations of motion are projected onto a set of modes (e.g. POD modes) (Holmes *et al.* 2012). However, in this approach, all the higher POD modes are neglected. An extension of this idea, called nonlinear Galerkin, is to assume that the time derivative of the coefficients of all of the higher modes is zero, but not

the coefficients themselves (Foias *et al.* 1988; Titi 1990; Graham, Steen & Titi 1993); this is essentially a quasisteady state approximation for the higher modes. This improves the accuracy, but comes at a higher computational cost than the Galerkin method, although this can be somewhat mitigated by using a postprocessing Galerkin approach (García-Archilla, Novo & Titi 1998). Wan *et al.* (2018) also showed a recurrent neural network (RNN), a NN that feeds into itself, can be used to improve the nonlinear Galerkin approximation. This RNN structure depends on a history of inputs, making it non-Markovian. In addition to these linear dimension reduction approaches, an autoencoder can be used with the equations of motion in the so-called manifold Galerkin approach, which Lee & Carlberg (2020) developed and applied to the viscous Burgers equation.

When the equations of motion are assumed to be unknown, and only snapshots of data are available, a number of different machine learning techniques exist to approximate the dynamics. Two of the most popular techniques are RNNs and reservoir computers. Vlachas *et al.* (2020) showed both these structures do an excellent job of capturing the chaotic dynamics of the Lorenz-96 equation and Kuramoto–Sivashinsky equation (KSE). For fluid flows, autoencoders and RNNs (specifically long short-term memory (LSTM) networks) have been used to model flow around a cylinders (Eivazi *et al.* 2020; Hasegawa *et al.* 2020*a*), pitching airfoils (Eivazi *et al.* 2020), bluff bodies (Hasegawa *et al.* 2020*b*) and MFU plane Poiseuille flow (PPF) (Nakamura *et al.* 2021). Although these methods often do an excellent job of predicting chaotic dynamics, the models are not Markovian, so a true assessment of the dimension of the state in these models has to account for the degrees of freedom used to store past history of the dynamics. These methods are also implemented in discrete rather than continuous time. These two properties are undesirable, because the underlying dynamics are Markovian and continuous in time, and not modelling them as such complicates applications and interpretations of the model. In particular, we want to use the model for state space analyses such as determination of periodic orbits (POs), where standard tools are available for ODEs that do not easily generalise to non-Markovian dynamic models.

Owing to these issues, we use neural ordinary differential equations (ODEs) (Chen *et al.* 2019). In neural ODEs, the right-hand side of an ODE is represented as a NN that is trained to reconstruct the time evolution of the data from snapshots of training data. Linot & Graham (2022) showed that this is an effective method for modelling the chaotic dynamics of the KSE. Rojas, Dengel & Ribeiro (2021) used neural ODEs to predict the periodic dynamics of flow around a cylinder, and Portwood *et al.* (2019) used neural ODEs to predict the kinetic energy and dissipation of decaying turbulence.

In this work we investigate the dynamics of MFU Couette flow. The idea behind the MFU, first introduced by Jiménez & Moin (1991), is to reduce the simulation domain to the smallest size that sustains turbulence, thus isolating the key components of the turbulent nonlinear dynamics. Using an MFU for Couette flow at transitional Reynolds number, Hamilton *et al.* (1995) outlined the regeneration cycle of wall-bounded turbulence called the 'self-sustaining process' (SSP), which we describe in more detail in § 3. This system was later analysed with covariant Lyapunov analysis by Inubushi, Takehiro & Yamada (2015), who found a Lyapunov time (the inverse of the leading Lyapunov exponent) of $\sim$48 time units.

Many low-dimensional models have been developed to recreate the dynamics of the SSP. The first investigation of this topic was by Waleffe (1997), who developed an 8-mode model for shear flow between free-slip walls generated by a spatially sinusoidal forcing. He selected the modes based on intuition from the SSP and performed a Galerkin projection onto these modes. Moehlis, Faisst & Eckhardt (2004) later added an additional mode to

Waleffe's model which enables modification of the mean profile by the turbulence, and made some modifications to the chosen modes. In this 'MFE' (Moehlis, Faisst, Eckhardt) model, Moehlis *et al.* found exact coherent states (ECS), which we discuss in the following, that did not exist in the 8-mode model. In addition, Moehlis *et al.* (2002) also used the POD modes on a domain slightly larger than the MFU to generate POD-Galerkin models. These low-dimensional models have been used as a starting point for testing data-driven models. For example, both LSTMs (Srinivasan *et al.* 2019) and a Koopman operator method with nonlinear forcing (Eivazi *et al.* 2021) have been used to attempt to reconstruct the MFE model dynamics. Borrelli *et al.* (2022) then applied these methods to PPF.

Finally, we note that a key approach to understanding complex nonlinear dynamical phenomena, such as the SSP of near-wall turbulence, is through study of the underlying state space structure of fixed points and POs. In the turbulence literature these are sometimes called ECS (Kawahara, Uhlmann & van Veen 2012; Graham & Floryan 2021). Turbulence organises around ECS in the sense that trajectories chaotically move between different such states. The first ECS found were fixed-point solutions in PCF (Nagata 1990). Following this work, Waleffe (1998) was able to connect ECS of PCF and PPF to the SSP. Later, more fixed point ECS were found in MFU PCF and visualised by Gibson, Halcrow & Cvitanović (2008*a*). Unlike fixed points, which cannot capture dynamic phenomena at all, POs are able to represent key aspects of turbulent dynamics such as bursting behaviour. Kawahara & Kida (2001) found the first two POs for MFU PCF, one of which had statistics that agreed well with the SSP. Then, Viswanath (2007) found another PO and 4 new relative periodic orbits (RPOs) in this domain, and Gibson made these solutions available in (Gibson *et al.* 2008*b*), along with a handful of others.

In the present work, we use autoencoders and neural ODEs, in a method we call 'data-driven manifold dynamics' (DManD) (Linot *et al.* 2023*a*), to build a reduced-order model (ROM) for turbulent MFU PCF (Hamilton *et al.* 1995). Section 2 outlines the details of the DManD framework. We then describe the details of the Couette flow in § 3.1, the results of the dimension reduction in § 3.2, and the DManD model's reconstruction of short- and long-time statistics in §§ 3.3 and 3.4, respectively. After showing that the models accurately reproduce these statistics, we compute RPOs for the model in § 3.5, finding several that are similar to previously known RPOs, as well as several that seem to be new. Finally, we summarise the results in § 4.

2. Framework

The fundamental ideas of the DManD framework are (1) that when modelling long-time dynamics of a dissipative system, only the finite-dimensional manifold $\mathcal{M}$ on which the dynamics lie needs to be considered, not the full state space, and (2) a low-dimensional coordinate system for this manifold and the dynamics in this coordinate system can be determined from data for the system. In general, the training data for the development of a DManD model come in the form of snapshots $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M\}$, which are either the full state or measurements diffeomorphic to the full state (e.g. time delays (Takens 1981; Young & Graham 2023)). Here we consider full-state data $\boldsymbol{u}$ that live in an ambient space $\mathbb{R}^d$. We generate a time series of data by evolving this state forward in time according to

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{u}). \tag{2.1}$$

(In the present context, this equation represents a fully resolved direct numerical simulation (DNS).) With the full state, we can then define a mapping to a low-dimensional

state representation

$$\boldsymbol{h} = \boldsymbol{\chi}(\boldsymbol{u}), \tag{2.2}$$

where $\boldsymbol{h} \in \mathbb{R}^{d_h}$ is a state in the manifold coordinate system. Finally, we define a mapping back to the full state

$$\tilde{\boldsymbol{u}} = \check{\boldsymbol{\chi}}(\boldsymbol{h}). \tag{2.3}$$

For data that lie on a finite-dimensional invariant manifold of dimension $d_{\mathcal{M}}$, these functions can, in principle, reconstruct the state exactly (i.e. $\tilde{\boldsymbol{u}} = \boldsymbol{u}$). However, if the dimension d_h is too low, or there are errors in the approximation of these functions, then $\tilde{\boldsymbol{u}}$ approximates the state. Here we also note that we refer to $\boldsymbol{h}$ as a state in the manifold coordinate system, but this coordinate representation is non-unique, and we seek one of an infinite number of diffeomorphic representations. Then, with this low-dimensional state representation, we can define an evolution equation

$$\frac{\mathrm{d}\boldsymbol{h}}{\mathrm{d}t} = \boldsymbol{g}(\boldsymbol{h}). \tag{2.4}$$

The DManD model consists of the three functions $\boldsymbol{\chi}$, $\check{\boldsymbol{\chi}}$ and $\boldsymbol{g}$. By approximating these functions, the evolution of trajectories on the manifold can be performed entirely in the manifold coordinates, which requires far fewer operations than a full simulation, as $d_h \ll d$. We choose to approximate all of these functions using NNs, but other representations could be used. The parameters (weights and biases) of these NNs ($\boldsymbol{\theta}_E, \boldsymbol{\theta}_D, \boldsymbol{\theta}_g, \boldsymbol{\theta}_\phi$) will be updated from data to minimise losses described in the following.

Before continuing to the details of the approach, we pause here to emphasise some specific aspects of the model. In traditional ROMs, approximations are made to reduce the number of degrees of freedom, e.g. in POD-Galerkin the governing equations are linearly projected onto a low-dimensional subspace, with the inevitable loss of information. By contrast, if the dynamics indeed lie on a manifold embedded in the ambient state space, then in principle there is an *exact* mapping between coordinates in the ambient space and the manifold, i.e. a reduced-dimensional representation that involves no approximation. Similarly, on the invariant manifold, the dynamics are tangent to the manifold and can be exactly represented by an ODE on it. Thus, in this context there is no issue of 'asymptotic convergence' as the number of degrees of freedom in the model increases: the invariant manifold has a definite dimension $d_{\mathcal{M}}$, and once the manifold representation has that dimension, no further degrees of freedom should be necessary. In practice, determining $d_{\mathcal{M}}$ from data is challenging (see Zeng & Graham 2023), and even if it is known, numerical errors will be present in whatever method (here NNs) is used to approximate the multidimensional nonlinear functions $\boldsymbol{\chi}$, $\check{\boldsymbol{\chi}}$, and $\boldsymbol{g}$ that define the model.

Now we return to the implementation of the DManD approach. First, we train $\boldsymbol{\chi}$ and $\check{\boldsymbol{\chi}}$ using an undercomplete autoencoder. This is a NN structure consisting of an encoder, which reduces dimension ($\boldsymbol{\chi}$), and a decoder, which expands dimension ($\check{\boldsymbol{\chi}}$). (An *undercomplete* autoencoder generates a lower-dimensional representation of the data ($d_h < d$) (Goodfellow, Bengio & Courville 2016). *Overcomplete* autoencoders also exist.) As mentioned in § 1, a common approach to dimension reduction is to project onto a set of POD modes so we will work with the data expressed in the POD basis, as detailed in § 3.2. Two specific approaches are used. POD gives the optimal *linear* projection in terms of reconstruction error, so in one approach we use this fact to train a so-called *hybrid autoencoder*. Here the encoder is written as the sum of POD and a correction in the form

of a NN:

$$\boldsymbol{h} = \boldsymbol{\chi}(\boldsymbol{u}; \boldsymbol{\theta}_E) = \boldsymbol{U}_{d_h}^{\mathrm{T}} \boldsymbol{u} + \boldsymbol{\mathcal{E}}(\boldsymbol{U}_r^{\mathrm{T}} \boldsymbol{u}; \boldsymbol{\theta}_E). \tag{2.5}$$

In this equation, $\boldsymbol{U}_k \in \mathbb{R}^{d \times k}$ is a matrix whose k columns are the first k POD modes as ordered by variance and $\boldsymbol{\mathcal{E}}$ is a NN. The first term ($\boldsymbol{U}_{d_h}^{\mathrm{T}} \boldsymbol{u}$) is the projection onto the leading d_h POD modes, and the second term is the NN correction. The matrix $\boldsymbol{U}_r$ in this term may either be a full change of basis with no approximation ($r = d$), or involve some dimension reduction ($d > r > d_h$).

For mapping back to the full state (decoding), we again sum POD with a correction

$$\tilde{\boldsymbol{u}} = \check{\boldsymbol{\chi}}(\boldsymbol{h}; \boldsymbol{\theta}_E) = \boldsymbol{U}_r([\boldsymbol{h}, 0]^{\mathrm{T}} + \boldsymbol{\mathcal{D}}(\boldsymbol{h}; \boldsymbol{\theta}_D)). \tag{2.6}$$

Here, $[\boldsymbol{h}, \boldsymbol{0}]^{\mathrm{T}}$ is the $\boldsymbol{h}$ vector zero-padded to the correct size, and $\boldsymbol{\mathcal{D}}$ is a NN. The first term is the POD mapping back to the full space, and the second term is a NN correction. These hybrid autoencoder operations act as a shortcut connection on the optimal linear dimension reduction, which we (Linot & Graham 2020) found useful for representing the data and achieving accurate reconstruction of $\boldsymbol{u}$. Yu *et al.* (2021) also took a similar approach with POD shortcut connections over each layer of the network. We determine the NN parameters $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}_D$ by minimising

$$\begin{aligned} L = &\frac{1}{dK} \sum_{i=1}^{K} \|\boldsymbol{u}(t_i) - \check{\boldsymbol{\chi}}(\boldsymbol{\chi}(\boldsymbol{u}(t_i); \boldsymbol{\theta}_E); \boldsymbol{\theta}_D)\|^2 \\ &+ \frac{1}{d_h K} \sum_{i=1}^{K} \alpha \|\boldsymbol{\mathcal{E}}(\boldsymbol{U}_r^{\mathrm{T}} \boldsymbol{u}(t_i); \boldsymbol{\theta}_E) + \boldsymbol{\mathcal{D}}_{d_h}(\boldsymbol{h}(t_i); \boldsymbol{\theta}_D)\|^2. \end{aligned} \tag{2.7}$$

The first term in this loss is the mean-squared error (MSE) of the reconstruction $\tilde{\boldsymbol{u}}$, and the second term is a penalty that promotes accurate representation of the leading d_h POD coefficients. In the second term, $\boldsymbol{\mathcal{D}}_{d_h}$ denotes the leading d_h elements of the decoder output. When this term vanishes (with $\alpha > 0$), the autoencoder exactly matches the first d_h POD coefficients. This penalty does not reduce the size of the correction by the encoder, rather it promotes the removal of the encoder correction by the decoder. Here, and in the following, the norm is defined to be the L_2 norm $\|\boldsymbol{q}\|^2 = \sum_{i=1}^{N} q_i^2$, so the normalisation in front of the terms is to average over elements in the vector and the batch K. For selecting α and other hyperparameters of the NNs we swept over the parameters to find the best. We discuss the details of the hyperparameter selection and the minimisation procedure in § 3.

The second approach we use is a standard autoencoder, where, while still working in the POD basis, we do not learn the encoder and decoder in terms of deviations from POD projections but simply set $\boldsymbol{h} = \boldsymbol{\chi}(\boldsymbol{u}; \boldsymbol{\theta}_E) = \boldsymbol{\mathcal{E}}(\boldsymbol{U}_r^{\mathrm{T}} \boldsymbol{u}; \boldsymbol{\theta}_E)$ and $\tilde{\boldsymbol{u}} = \check{\boldsymbol{\chi}}(\boldsymbol{h}; \boldsymbol{\theta}_E) = \boldsymbol{U}_r \boldsymbol{\mathcal{D}}(\boldsymbol{h}; \boldsymbol{\theta}_D)$. In § 3.2 we contrast the standard and hybrid approaches. In Linot & Graham (2020) we found lower autoencoder error by performing the POD change of basis, and even lower error by taking the hybrid approach for chaotic data from the KSE. In the present context, we found the performance of the two approaches in the POD basis to be very similar, as further discussed in the following.

Next, we approximate $\boldsymbol{g}$ using a neural ODE. A drawback of training a single dense NN for $\boldsymbol{g}$ is that the resulting dynamics may become weakly unstable, with linear growth at long times (Linot & Graham 2022; Linot *et al.* 2023*a*). To avoid this, we use a 'stabilised'

neural ODE approach by adding a linear damping term onto the output of the NN, giving

$$\boldsymbol{g}(\boldsymbol{h}(t_i);\boldsymbol{\theta}_g) = \boldsymbol{g}_{NN}(\boldsymbol{h}(t_i);\boldsymbol{\theta}_g) + \boldsymbol{A}\boldsymbol{h}. \tag{2.8}$$

For training $\boldsymbol{g}_{NN}$ we integrate (2.8) over some time window τ from time t_i to $t_i + \tau$ yielding

$$\tilde{\boldsymbol{h}}(t_i + \tau) = \boldsymbol{h}(t_i) + \int_{t_i}^{t_i+\tau} \boldsymbol{g}_{NN}(\boldsymbol{h}(t);\boldsymbol{\theta}_g) + \boldsymbol{A}\boldsymbol{h}(t)\,\mathrm{d}t. \tag{2.9}$$

We use the standard Dormand–Prince method (Dormand & Prince 1980) to perform this integration, but we note that this method does not depend on the integration scheme and different methods can be used when training and deploying the model. Depending on the situation, one may either learn $\boldsymbol{A}$ from data or fix it. Here we set it to the diagonal matrix

$$A_{ij} = -\beta\delta_{ij}\sigma_i(\boldsymbol{h}), \tag{2.10}$$

where $\sigma_i(\boldsymbol{h})$ is the standard deviation of the ith component of $\boldsymbol{h}$, β is a tunable parameter and δ_{ij} is the Kronecker delta. In Linot & Graham (2022) and Linot *et al.* (2023*a*), we found that without this stabilising linear term, small errors in the dynamics would eventually grow resulting in $\boldsymbol{g}_{NN}$ outputting a constant value, which led to continuous linear growth away from the attractor. Including the linear damping term effectively eliminates this behaviour by attracting trajectories back to the origin, preventing them from moving far away from the training data. In § 3.4 we show that this approach drastically improves the long-time performance of these models.

We then determine the parameters $\boldsymbol{\theta}_g$ by minimising the difference between the predicted state $\tilde{\boldsymbol{h}}(t_i + \tau)$ and the true state $\boldsymbol{h}(t_i + \tau)$, averaged over the data:

$$J = \frac{1}{d_h K}\sum_{i=1}^{K}(\|\boldsymbol{h}(t_i + \tau) - \tilde{\boldsymbol{h}}(t_i + \tau)\|^2). \tag{2.11}$$

For clarity, we show the specific loss we use, which sums over only a single snapshot forward in time at a fixed τ. More generally, the loss can be formulated for arbitrary snapshot spacing and for multiple snapshots forward in time. To compute the gradient of J with respect to the NN parameters $\boldsymbol{\theta}_g$, automatic differentiation can be used to backpropagate through the ODE solver that is used to compute the time integral in (2.9), or an adjoint problem can be solved backwards in time (Chen *et al.* 2019). The adjoint method uses less memory than backpropagation, but $\boldsymbol{h}$ is low-dimensional and we will select a short time window for training, so we choose to backpropagate through the solver.

So far this approach to approximating $\boldsymbol{\chi}$, $\check{\boldsymbol{\chi}}$ and $\boldsymbol{g}$ is general and does not directly account for the fact that the underlying equations are often invariant to certain symmetry operations. For example, one of the symmetries in PCF is a continuous translation symmetry in x and z (i.e. any solution shifted to another location in the domain gives another solution). This poses an issue for training, because, in principle, the training data must include *all* these translations to accurately model the dynamics under any translation. We discuss these and other symmetries of PCF in § 3.1.

In practice, accounting for continuous symmetries is most important along directions that sample different phases very slowly. For PCF, the mean flow is in the x direction, leading to good phase sampling along this direction. However, there is no mean flow in the z direction, so sampling all phases relies on the slow phase diffusion in that direction. Therefore, we only explicitly account for the z-phase in § 3, but in the current discussion we present the general framework accounting for all continuous symmetries.

One intuitive way to account for the phase would be to augment the data with phase-shifted versions of the data, however, this suffers from two drawbacks: (1) this can drastically slow down the training procedure by including all of these phases and (2) data augmentation does not guarantee that the resulting dynamics are equivariant to phase shifts. Instead, to address the issue of continuous translations, we add an additional preprocessing step to the data, using the method of slices (Budanur, Borrero-Echeverry & Cvitanović 2015*a*; Budanur *et al.* 2015*b*) to split the state $\boldsymbol{u}$ into a pattern $\boldsymbol{u}_p \in \mathbb{R}^d$ and a phase $\boldsymbol{\phi} \in \mathbb{R}^c$. Pérez De Jesús & Graham (2023) found for Kolmogorov flow that this step leads to an order of magnitude reduction in mean squared reconstruction error. The number of continuous translation symmetries for which we explicitly account determines c. We discuss the details of computing the pattern and the phase in § 3.1.

In transforming the data to pattern and phase dynamics no information about the system is lost, and the autoencoder and neural ODE become equivariant to translations. As shown in Linot & Graham (2020) and Pérez De Jesús & Graham (2023), autoencoders perform better on the pattern dynamics than the state without phase alignment. In addition, separating the pattern and phase is useful because the evolution of both the pattern and the phase only depend on the pattern. Thus, we simply replace $\boldsymbol{u}$ with $\boldsymbol{u}_p$ in all the above equations and then write one additional ODE for the phase

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \boldsymbol{g}_\phi(\boldsymbol{h}; \boldsymbol{\theta}_\phi). \tag{2.12}$$

We then fix the parameters of $\boldsymbol{g}$ to evolve $\boldsymbol{h}$ (from $\boldsymbol{u}_p$) forward in time and use that to make a phase prediction

$$\tilde{\boldsymbol{\phi}}(t_i + \tau) = \boldsymbol{\phi}(t_i) + \int_{t_i}^{t_i+\tau} \boldsymbol{g}_\phi(\boldsymbol{h}(t_i); \boldsymbol{\theta}_\phi)\, \mathrm{d}t. \tag{2.13}$$

Finally, we determine the parameters $\boldsymbol{\theta}_\phi$ to minimise the difference between the predicted phase $\tilde{\boldsymbol{\phi}}(t_i + \tau)$ and the true phase $\boldsymbol{\phi}(t_i + \tau)$

$$J_\phi = \frac{1}{cK} \sum_{i=1}^{K} (\|\boldsymbol{\phi}(t_i + \tau) - \tilde{\boldsymbol{\phi}}(t_i + \tau)\|^2), \tag{2.14}$$

using the method described previously to compute the gradient of J_ϕ.

3. Results

3.1. *Description of PCF data*

In the following sections we apply DManD to DNS of turbulent PCF in a MFU domain. Specifically, we consider the well-studied Hamilton, Kim and Waleffe (HKW) domain (Hamilton *et al.* 1995). We made this selection to compare our DManD results with the analysis of the SSP in this domain, to compare our DManD results with other Galerkin-based ROMs and to compare our DManD results with known unstable periodic solutions in this domain.

For PCF we solve the Navier–Stokes equations (NSEs)

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{v} = -\boldsymbol{\nabla} p + Re^{-1} \nabla^2 \boldsymbol{v}, \quad \boldsymbol{\nabla} \cdot \boldsymbol{v} = 0 \tag{3.1}$$

for a fluid confined between two plates moving in opposite directions with the same speed. Equation (3.1) is the non-dimensionalised form of the equations with velocities in

the streamwise $x \in [0, L_x]$, wall-normal $y \in [-1, 1]$ and spanwise $z \in [0, L_z]$ directions defined as $\boldsymbol{v} = [v_x, v_y, v_z]$ and pressure p. We solve this equation for a domain with periodic boundary conditions in x and z ($\boldsymbol{v}(0, y, z) = \boldsymbol{v}(L_x, y, z)$, $\boldsymbol{v}(x, y, 0) = \boldsymbol{v}(x, y, L_z)$) and no-slip, no-penetration boundary conditions in y ($v_x(x, \pm 1, z) = \pm 1$, $v_y(x, \pm 1, z) = v_z(x, \pm 1, z) = 0$). The complexity of the flow increases as the Reynolds number Re increases and the domain size L_x and L_z increase. Here we use the HKW cell, which is at $Re = 400$ with a domain size $[L_x, L_y, L_z] = [1.75\pi, 2, 1.2\pi]$ (Hamilton *et al.* 1995). The HKW cell is one of the simplest flows that sustains turbulence for extended periods of time before relaminarising. We chose to use this flow because it is well studied (refer to § 1), it isolates the SSP (Hamilton *et al.* 1995) and a library of ECS exists for this domain (Gibson *et al.* 2008*b*). Here we are interested in modelling the turbulent dynamics, so we will remove data upon relaminarisation as detailed in the following.

Equation (3.1), under the boundary conditions described, is invariant (and its solutions equivariant) under the discrete symmetries of point reflections about $[x, y, z] = [0, 0, 0]$

$$\mathcal{P}[(v_x, v_y, v_z, p)(x, y, z, t)] = (-v_x, -v_y, -v_z, p)(-x, -y, -z, t) \tag{3.2}$$

reflection about the $z = 0$ plane

$$\mathcal{R}[(v_x, v_y, v_z, p)(x, y, z, t)] = (v_x, v_y, -v_z, p)(x, y, -z, t) \tag{3.3}$$

and rotation by π about the z-axis

$$\mathcal{RP}[(v_x, v_y, v_z, p)(x, y, z, t)] = (-v_x, -v_y, v_z, p)(-x, -y, z, t). \tag{3.4}$$

In addition to the discrete symmetries, there are also continuous translation symmetries in x and z

$$\mathcal{T}_{\sigma_x,\sigma_z}[(v_x, v_y, v_z, p)(x, y, z, t)] = (v_x, v_y, v_z, p)(x + \sigma_x, y, z + \sigma_z, t). \tag{3.5}$$

We incorporate all these symmetries in the POD representation (Smith *et al.* 2005), as we discuss further in § 3.2. Then, we use the method of slices (Budanur *et al.* 2015*a*) to phase align in the z direction. By phase aligning in z we fix the location of the low-speed streak. Without the alignment in z, models performed poorly because the models needed to learn how to represent every spatial shift of every snapshot. In what follows, we only consider phase alignment in z, but we note that extending this work to phase alignment in x is straightforward. To phase align the data, we use the first Fourier mode method of slices (Budanur *et al.* 2015*a*). First, we compute a phase by taking the Fourier transform of the streamwise velocity in x and z ($\hat{v}_x(k_x, y, k_z) = \mathcal{F}_{x,z}(v_x)$) at a specific y location (y_1) to compute the phase

$$\phi = \text{atan2}(\text{imag}(\hat{v}_x(0, y_1, 1)), \text{real}(\hat{v}_x(0, y_1, 1))). \tag{3.6}$$

The variables k_x and k_z are the streamwise and spanwise wavenumbers. We select y_1 to be one grid point off the bottom wall. Any y location should work for the phase alignment, but we found choosing the point in the viscous sublayer resulted in a more gradual change of the phase. Rapid changes in the phase make prediction difficult and can require rescaling time, as in Budanur *et al.* (2015*b*), which we want to avoid. Then we compute the pattern dynamics by using the Fourier shift theorem to set the phase to 0 (i.e. move the low-speed streak to the centre of the channel)

$$\boldsymbol{v}_p = \mathcal{F}^{-1}_{x,z}(\hat{\boldsymbol{v}} \exp(-\mathrm{i}k_z\phi)). \tag{3.7}$$

We generate turbulent PCF trajectories using the pseudo-spectral Channelflow code developed by Gibson (2012) and Gibson *et al.* (2021). In this code, the velocity and

pressure fields are projected onto Fourier modes in x and z and Chebyshev polynomials of the first kind in y. These coefficients are evolved forward in time first using the multistage SMRK2 scheme (Spalart, Moser & Rogers 1991), then, after taking multiple timesteps, using the multistep Adams–Bashforth backward-differentiation 3 scheme (Peyret 2002). At each timestep, a pressure boundary condition is found such that incompressibility is satisfied at the wall ($\mathrm{d}v_y/\mathrm{d}y = 0$) using the influence matrix method and tau correction developed by Kleiser & Schumann (1980).

Data were generated with $\Delta t = 0.0325$ on a grid of $[N_x, N_y, N_z] = [32, 35, 32]$ in x, y and z for the HKW cell. Starting from random divergence-free initial conditions, we ran simulations forward for either 10 000 time units or until relaminarisation. Then we dropped the first 1000 time units as transient data and the last 1000 time units to avoid laminar data, and repeated with a new initial condition until we had 91 562 time units of data stored at intervals of one time unit. We split these data into 80 % for training and 20 % for testing. Finally, we preprocess the data by computing the mean over snapshots and the x and z directions $\bar{\boldsymbol{v}}(y)$ from the training data and subtracting it from all data $\boldsymbol{v}' = \boldsymbol{v} - \bar{\boldsymbol{v}}$, and then we compute the pattern $\boldsymbol{v}_p'$ and the phase ϕ as described previously. The pattern $\boldsymbol{u}_p$ described in § 2 is $\boldsymbol{v}_p'$ flattened into a vector (i.e. $d = 3N_xN_yN_z$). The POD and NN training use only the training data, and all comparisons use test data unless otherwise specified.

3.2. *Dimension reduction and dynamic model construction*

3.2.1. *Linear dimension reduction with POD: from $O(10^5)$ to $O(10^3)$*

The first task in DManD for this Couette flow data is finding a low-dimensional parameterisation of the manifold on which the long-time dynamics lie. We parameterise this manifold in two steps. First, we reduce the dimension down from $d = O(10^5)$ to $d_{POD} = O(10^3)$ with the POD and, second, we use an autoencoder to reduce the dimension down to d_h, which ideally is equal to $d_{\mathcal{M}}$. The first step is simply a preprocessing step to reduce the size of the data, which reduces the number of parameters in the autoencoder. Due to Whitney's embedding theorem (Whitney 1936, 1944), we know that as long as $d_{\mathcal{M}} < d_{POD}/2$, then this POD representation is diffeomorphic to the full state. As we show later, the manifold dimension $d_{\mathcal{M}}$ appears to be far lower than $d_{POD}/2$, so no information of the full state should be lost with this first step.

In other words, there are four different spaces for representing the state in this process. There is the (infinite-dimensional) solution space of the NSEs, the d-dimensional data space generated by the DNS, which we will refer to as the 'full state', the d_{POD}-dimensional POD representation of the full state and the d_h-dimensional representation in the manifold coordinate system. We assume that our high-resolution DNS solution accurately represents a true solution of the NSEs, and further assume that this solution lies on a low-dimensional manifold of dimension $d_{\mathcal{M}}$. Then, given a sufficient number of dimensions, the POD, followed by the autoencoder, should provide a representation diffeomorphic to the full state. However, we cannot know that a sufficient number of dimensions has been chosen until after training the models.

The POD originates with the question of what function $\boldsymbol{\Phi}$ maximises

$$\frac{\langle |(\boldsymbol{v}', \boldsymbol{\Phi})_E|^2 \rangle}{\|\boldsymbol{\Phi}\|_E^2}, \tag{3.8}$$

where $\langle \cdot \rangle$ is the ensemble average and the inner product is defined to be

$$(\boldsymbol{q}_1, \boldsymbol{q}_2)_E = \frac{1}{2L_xL_z} \iiint_V \boldsymbol{q}_1 \cdot \boldsymbol{q}_2 \, \mathrm{d}\boldsymbol{x}, \tag{3.9}$$

with the corresponding energy norm $\|\boldsymbol{q}\|_E^2 = (\boldsymbol{q}, \boldsymbol{q})_E$. Solutions $\boldsymbol{\Phi}^{(n)}$ to this problem satisfy the eigenvalue problem

$$\sum_{j=1}^{3} \int_0^{L_x} \int_{-1}^{1} \int_0^{L_z} \langle v_i'(\boldsymbol{x}, t) v_j'^*(\boldsymbol{x}', t) \rangle \Phi_j^{(n)}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x}' = \lambda_i \Phi_i^{(n)}(\boldsymbol{x}) \tag{3.10}$$

(Smith *et al.* 2005; Holmes *et al.* 2012). Unfortunately, upon approximating these integrals, with the trapezoidal rule for example, this becomes a $d \times d$ matrix, making computation intractable. Furthermore, computing the average in (3.10), without any modifications, results in POD modes that fail to preserve the underlying symmetries described previously.

In order to make this problem computationally tractable, and preserve symmetries, we apply the POD method used in Smith *et al.* (2005), with the slight difference that we first subtract off the mean of the state before performing the analysis. The first step in this procedure is to treat the POD modes as Fourier modes in both the x and z directions. Holmes *et al.* (2012) showed that for translation-invariant directions, Fourier modes are the optimal POD modes. This step transforms the eigenvalue problem into

$$L_x L_z \sum_{j=1}^{3} \int_{-1}^{1} \langle \hat{v}_i'(k_x, y', k_z, t) \hat{v}_j'^*(k_x, y', k_z, t) \rangle \varphi_{jk_xk_z}^{(n)}(y') \, \mathrm{d}y' = \lambda_{k_xk_z}^{(n)} \varphi_{ik_xk_z}^{(n)}(y), \tag{3.11}$$

which reduces the $d \times d$ eigenvalue problem down to a $3N_y \times 3N_y$ eigenvalue problem for every wavenumber pair (k_x, k_z) of Fourier coefficients. We used 5000 snapshots evenly sampled over the training data to compute the POD modes. Then, to account for the discrete symmetries, the data are augmented such that the mean in (3.11) is computed by adding all the discrete symmetries of each snapshot, i.e. we compute $\langle \hat{v}_i' \hat{v}_j'^* \rangle$ for $\boldsymbol{v}$, $\mathcal{P}\boldsymbol{v}$, $\mathcal{R}\boldsymbol{v}$ and $\mathcal{R}\mathcal{P}\boldsymbol{v}$ and average the results.

This analysis gives us POD modes

$$\boldsymbol{\Phi}_{k_xk_z}^{(n)}(\boldsymbol{x}) = \frac{1}{\sqrt{L_x L_z}} \exp\left(2\pi \mathrm{i} \left(\frac{k_x x}{L_x} + \frac{k_z z}{L_z}\right)\right) \boldsymbol{\varphi}_{k_xk_z}^{(n)}(y), \tag{3.12}$$

and eigenvalues $\lambda_{k_xk_z}^{(n)}$. The projection onto these modes results in complex values unless $k_x = k_z = 0$. We sort the modes from largest eigenvalue to smallest eigenvalue (λ_i) and project the phase-aligned data (i.e. $\boldsymbol{v}_p'$, as described in § 3.1) onto the leading 256 modes, giving us a vector of POD coefficients $\boldsymbol{a}(t)$. A majority of these modes are complex (i.e. 2 degrees of freedom), so projecting onto these modes results in a 502-dimensional system, i.e. $d_{POD} = 502$. In figure 1(*a*) we plot the eigenvalues, which show a rapid drop and then a long tail that contributes little to the energy content. By dividing the eigenvalues of the leading 256 modes by the total, we find these modes contain 99.8 % of the energy. To further illustrate that 256 modes is sufficient to represent the state in this case, we consider the reconstruction of statistics after projecting onto the POD modes. In figure 1(*b*) we show the reconstruction of four components of the Reynolds stress, $\langle v_x'^2 \rangle$, $\langle v_z'^2 \rangle$, $\langle v_y'^2 \rangle$ and $\langle v_x' v_y' \rangle$. The projection onto POD modes matches all of these quantities extremely well.

3.2.2. *Nonlinear dimension reduction with autoencoders: from $O(10^3)$ to $O(10^1)$*

Now that we have converted the data to POD coefficients and filtered out the low-energy modes, we next train an autoencoder to perform *nonlinear* dimension reduction. However, we first 'preprocess' the POD coefficients by normalising the coefficients before using

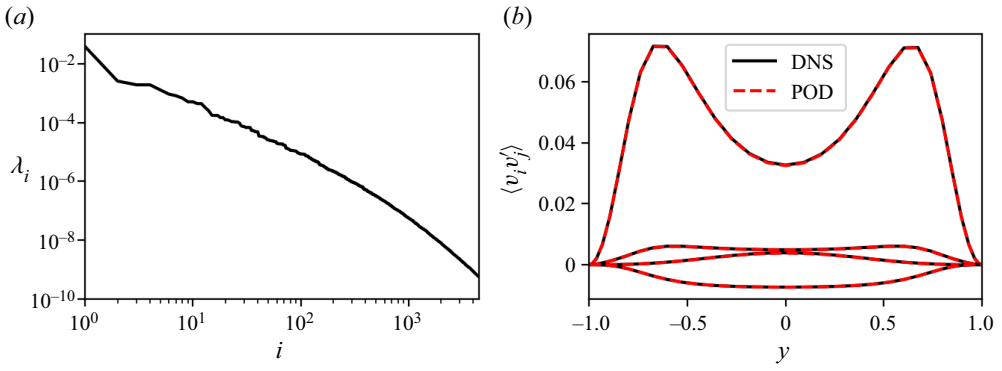


Figure 1. (*a*) Eigenvalues of POD modes sorted in descending order. (*b*) Components of the Reynolds stress for data generated by the DNS and this data projected onto 256 POD modes. In (*b*) the curves are, from top to bottom, $\langle v_x'^2 \rangle$, $\langle v_z'^2 \rangle$, $\langle v_y'^2 \rangle$ and $\langle v_x' v_y' \rangle$.

them in the autoencoder. It is common practice before NN training to subtract the mean and divide by the standard deviation of each component. We do not take this approach here because the standard deviation of the higher POD coefficients, which contribute less to the reconstruction, is much smaller than the lower POD coefficients. In order to retain the important information in the magnitudes, but put the values in a more amenable form for NN training, we instead normalise the POD coefficients by subtracting the mean and dividing by the maximum standard deviation.

With the above preprocessing step completed, we now turn to the reduction of dimension with the nonlinear approach enabled by the autoencoder structure. We consider three approaches to reducing the dimension of the normalised POD coefficient vector $\boldsymbol{a}$: (1) training a hybrid autoencoder; (2) training a standard autoencoder; and (3) linear projection onto a small set of POD modes. We describe the first two approaches in § 2, and note that the POD projection onto 256 (complex) modes can be written as $\boldsymbol{a} = \boldsymbol{U}_r^{\mathrm{T}} \boldsymbol{u}$. The third approach corresponds to setting $\boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{D}}$ to zero in 2.5 and 2.6.

To optimise the parameters of the hybrid and standard autoencoders, we use an Adam optimiser (Kingma & Ba 2015) in Keras (Chollet 2015) to minimise the loss in (2.7) with the normalised POD coefficients as inputs. We train for 500 epochs with a learning rate scheduler that drops the learning rate from 10^{-3} to 10^{-4} after 400 epochs. At this point, we see no improvement in the reconstruction error. For the hybrid autoencoder approach, we set $\alpha = 0.01$, and for the standard autoencoder $\alpha = 0$ (in practice, this term is not included). Table 1 includes additional NN architecture details. We determined these network parameters through a manual parameter sweep by varying the shape and activation functions of the NNs to achieve the lowest error without excessive computational cost.

In figure 2(*a*) we show the relative reconstruction error for the three approaches over a range of dimensions d_h. We compute this error by computing the energy norm of the difference between the state in the ambient space and the predicted state. Then, we divide that quantity by the energy norm of the state in the ambient space and average over the test data. We use NNs with the same architectures for both the standard and the hybrid autoencoder approaches. Due to the variability introduced into autoencoder training by randomly initialised weights and stochasticity in the optimisation, we show the error for four separately trained autoencoders, at each d_h. We see that the autoencoders perform much better than POD in the range of dimensions considered here.

Function	Shape	Activation	Learning rate
$\mathcal{E}$	$502/1000/d_h$	sig/linear	$[10^{-3}, 10^{-4}]$
$\mathcal{D}$	$d_h/1000/502$	sig/linear	$[10^{-3}, 10^{-4}]$
g_{NN}	$d_h/200/200/d_h$	sig/sig/linear	$[10^{-2}, 10^{-3}, 10^{-4}]$
g_ϕ	$d_h/200/200/1$	sig/sig/linear	$[10^{-2}, 10^{-3}, 10^{-4}]$

Table 1. Architectures of NNs. 'Shape' indicates the dimension of each layer, 'Activation' the corresponding activation functions and 'sig' is the sigmoid activation. 'Learning rate' gives the learning rate at multiple times during training. We dropped the learning rate at even intervals.

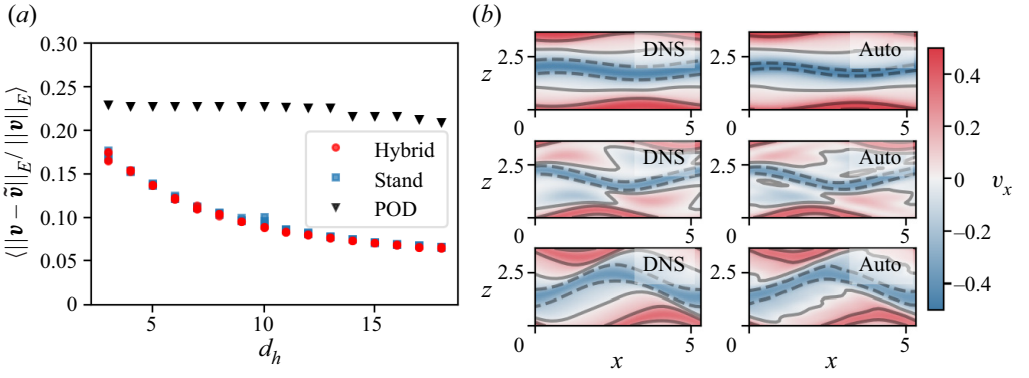


Figure 2. (*a*) Relative error in the energy norm on test data for POD, standard autoencoders and hybrid autoencoders at various dimensions d_h. At each dimension there are four standard and four hybrid autoencoders. (*b*) The centreline streamwise velocity for three examples in the DNS and reconstructed using the hybrid autoencoder with $d_h = 18$. The relative error of reconstruction from top to bottom is 0.021, 0.071 and 0.048.

The performance of the standard and hybrid autoencoder approaches is very similar, and the MSE begins to level off for $d_h \gtrsim 18$. Given a perfect autoencoder (i.e. one that has latent dimension $d_h \geq d_{\mathcal{M}}$ and sufficient capacity to determine an exact mapping between the ambient space and the manifold), the error in this figure ideally should vanish once d_h becomes high enough; however, there is always some error in the autoencoder, either due to finite capacity or limitations in the training process, that prevents this from happening. Figure 2(*b*) shows three examples of the flowfields and their reconstruction for a hybrid autoencoder at $d_h = 18$. This autoencoder captures the key structures of these snapshots. The results in the following use the hybrid approach. We took this approach because the low-dimensional representations of the hybrid autoencoders displayed less variability in predictions from trial to trial than those of the standard autoencoders. This led, in turn, to less variability when training neural ODEs to predict the time dynamics of the different low-dimensional representations of these autoencoders.

Before discussing how to use the low-dimensional representations from these autoencoders to train stabilised neural ODEs, we briefly mention convolutional neural networks (CNNs). These could be a reasonable choice for the autoencoder in the framework outlined previously. We chose not to use them because empirically we found lower errors using dense NNs in the POD basis for simple chaotic systems such as the KSE. Once in the POD basis, a CNN holds no advantage over a densely connected network because there are no longer spatial connections between the POD coefficients.

In addition, although standard CNNs are invariant to small translations of the input they are not equivariant to inputs, which is what we desire. Separating out pattern and phase variables ensures equivariance for any function approximation of $\boldsymbol{\chi}$ and $\check{\boldsymbol{\chi}}$, making it an appealing choice over trying to constrain the structure of the NN to ensure equivariance.

3.2.3. *Neural ODE training*

After training four autoencoders at each dimension d_h, we chose a set of damping parameters, β, and for each, then trained four stabilised neural ODEs for all four autoencoders at every dimension d_h. This results in 16 models at every d_h and β. The final β value of 0.1 was selected so that long-time trajectories neither blew up nor decayed too strongly. We trained each of these models to predict one time unit ahead (i.e. $\tau = 1$) because predicting further ahead took longer to train, and did not improve results. In Linot & Graham (2022) we ran a more extensive study on the KSE of varying this parameter, where we found the predictive capabilities do not start to deteriorate until the time window approached a Lyapunov time.

Before training the ODEs, we preprocess each autoencoder's latent space data set $\boldsymbol{h}$ by subtracting the mean. It is important to centre the data because the linear damping (2.10) pushes trajectories towards the origin. We train the stabilised neural ODEs to predict the evolution of the centred data by using an Adam optimiser in PyTorch (Chen *et al.* 2019; Paszke *et al.* 2019) to minimise the loss in (2.11). We train using a learning rate scheduler that drops at three even intervals during training and we train until the learning curve stops improving. Table 1 presents the details of this NN. Unless otherwise stated, we show results for the 1 model out of those 16 at each dimension with the lowest relative error averaged over all the statistics we consider. Here we emphasise that for all comparisons between the DNS and the DManD model we will use the same initial conditions for the two simulations. In the DManD model, this consists of encoding the initial condition for the DNS and then evolving it forward with the neural ODE in the latent space to obtain a time series of $\boldsymbol{h}$. Then we decode this time series for comparison. Finally, to provide a rough estimate of the computational cost of training these NNs we note that training the autoencoder ($\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}$), the stabilised neural ODE ($\boldsymbol{g}_{NN}$) and the phase neural ODE ($\boldsymbol{g}_{\phi}$) to convergence took $\sim$4.4, 1.0 and 1.7 h, respectively, for the $d_h = 18$ hybrid autoencoder model on a 2.40 GHz Intel Xeon CPU E5-2640 v4.

3.3. *Short-time tracking*

In the following two sections, we evaluate the performance of the DManD models at reconstructing short-time trajectories and long-time statistics. Figure 3 shows snapshots of the streamwise velocity at the centre plane of the channel, $y = 0$, for the DNS and DManD at $d_h = 18$. We choose to show results for $d_h = 18$ because the autoencoder error begins to level off around this dimension and, as we will show, the error in statistics levels off before this dimension. The value $d_h = 18$ is not necessarily the minimal dimension required to model this system. In figure 3, both the DNS and the DManD model show key characteristics of the SSP: (1) low-speed streaks become wavy; (2) the wavy low-speed streaks break down generating rolls; (3) the rolls lift fluid from the walls, regenerating streaks.

Not only does DManD capture the qualitative behaviour of the SSP, but figure 3 also shows good quantitative agreement as well. To further illustrate this, in figure 4 we show

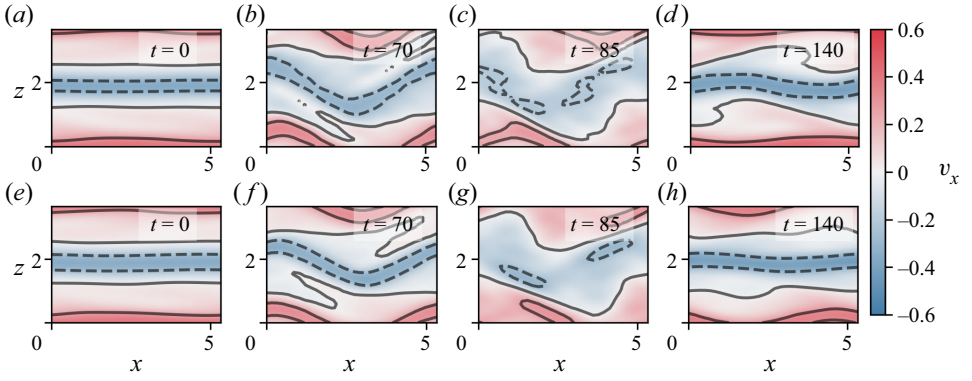


Figure 3. Snapshots of the streamwise velocity at $y = 0$ from the DNS and from the DManD model at $d_h = 18$.

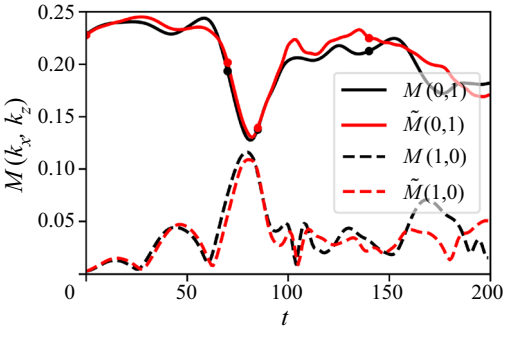


Figure 4. Modal RMS velocity from the DNS (M) and from the DManD model at $d_h = 18$ ($\tilde{M}$). The markers correspond to the times in figure 3.

the modal root-mean-squared (RMS) velocity

$$M(k_x, k_z) = \left(\int_{-1}^{1} (\hat{v}_x^2(k_x, y, k_z) + \hat{v}_y^2(k_x, y, k_z) + \hat{v}_z^2(k_x, y, k_z)) \, \mathrm{d}y \right)^{1/2}, \tag{3.13}$$

which Hamilton *et al.* (1995) used to identify the different parts of the SSP. Specifically, we consider the $M(0, 1)$ mode, which corresponds to the low-speed streak, and the $M(1, 0)$ mode, which corresponds to the x-dependence that appears when the streak becomes wavy and breaks up. In this example, the two curves match well over a cycle of the SSP and only start to move away after ∼150 time units, which is about three Lyapunov times.

While the previous result shows a single example, we also consider ensembles of initial conditions. Figure 5 shows the tracking error $\|\boldsymbol{a}(t_i + t) - \tilde{\boldsymbol{a}}(t_i + t)\|$ of 10 trajectories, starting at t_i, for a model at $d_h = 18$. Here we normalise the tracking error by the error between solutions at random times t_i and t_j:$\mathcal{N} = \langle \|\boldsymbol{a}(t_i) - \boldsymbol{a}(t_j)\| \rangle$. Here we note that, because the 256 complex modes we use capture 99.8 % of the energy in the flow, this normalised tracking error will be approximately equal to the normalised tracking error in the full space. In this case, the darkest line corresponds to the flow field in figures 3 and 4. When considering the other initial conditions in figure 5, there tends to be a relatively slow rise in the error over ∼50 time units and then a more rapid increase after this point.

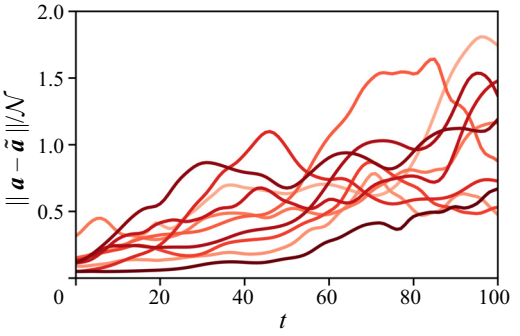


Figure 5. Normalised tracking error for 10 random initial conditions (different shades) using DManD with $d_h = 18$.

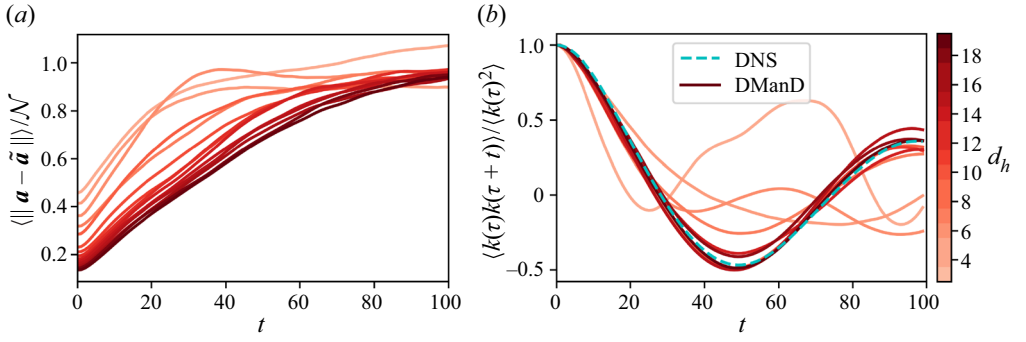


Figure 6. (*a*) Ensemble-averaged short-time tracking and (*b*) temporal autocorrelation of the kinetic energy for DManD models of increasing dimension. In (*b*), odd numbers above $d_h = 5$ are omitted for clarity.

To better understand how this tracking varies with the dimension of the model we next consider the ensemble-averaged tracking error.

In figure 6(*a*) we show the normalised ensemble-averaged tracking error for model dimensions between $d_h = 3$ and 18. For $d_h = 3$–5 there is a rapid rise in the error until ∼40 time units after which the error levels off. This behaviour often happens due to trajectories quickly diverging and landing on stable fixed points or POs that do not exist in the true system. For $d_h = 6$–10 there is an intermediate behaviour where lines diverge more quickly than on the higher-dimensional models, but tend to approach the same tracking error at ∼100 time units. Then, for the remaining models $d_h = 11$–18, there is a smooth improvement in the tracking error over this time interval. As the dimension increases in this range the trends stay the same, but the error continues to decrease, which is partially due to improvement in the autoencoder performance.

The instantaneous kinetic energy of the flow is given by $E(t) = 1/2\|\boldsymbol{v}\|_E^2$ (3.9) and we denote its fluctuating part as $k(t) = E(t) - \langle E\rangle$. In figure 6(*b*) we show the temporal autocorrelation of k. Again, for $d_h = 3$–5 we see clear disagreement between the true autocorrelation and the prediction. Above $d_h > 5$ all of the models match the temporal autocorrelation well, without a clear trend in the error as dimension changes. All these models match well for ∼40 time units, with $d_h = 18$ (the darkest line) matching the data extremely well for two Lyapunov times.

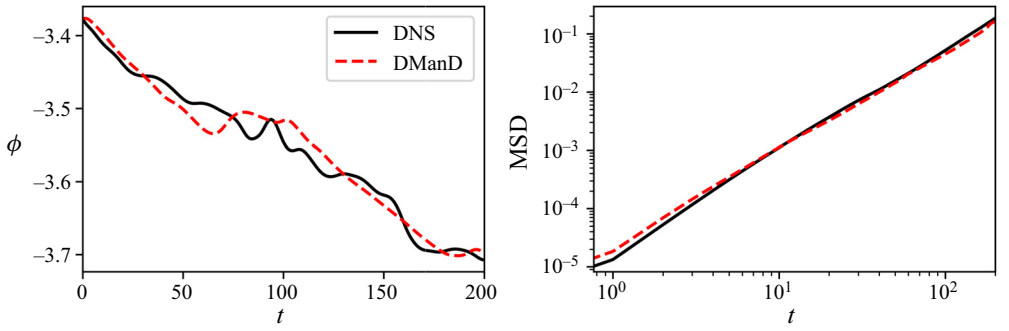


Figure 7. (*a*) Example of the phase evolution and (*b*) the MSD of the phase evolution for the DNS and the DManD model at $d_h = 18$.

Finally, before investigating the long-time predictive capabilities of the model, we show the tracking of phase dynamics for $d_h = 18$. As mentioned in § 2, we decouple the phase and pattern dynamics such that the time evolution of the phase only depends upon the pattern dynamics. Here we take the $d_h = 18$ model and used it to train an ODE for the phase dynamics. For training we repeat the process used for training $\boldsymbol{g}_{NN}$ to train $\boldsymbol{g}_\phi$ with the loss in (2.14). Table 1 contains details on the NN architecture.

In figure 7(*a*) we show an example of the model phase evolution over 200 time units. In this example, the model follows the same downward drift in phase despite not matching exactly. Then, to show the statistical agreement between the DNS and the model, we show the mean squared phase displacement $\text{MSD} = \langle(\phi(t_i) - \phi(t_i + t))^2\rangle$ for both the DNS and the model in figure 7(*b*), as was done for Kolmogorov flow by Pérez De Jesús & Graham (2023). The curves are in good agreement. All of the remaining long-time statistics we report are phase invariant, so the remaining results use only models for the pattern dynamics.

3.4. *Long-time statistics*

Next, we investigate the ability of the DManD model to capture the long-time dynamics of PCF. An obvious prerequisite for models to capture long-time dynamics is the long-time stability of the models. As mentioned in § 2, the long-time trajectories of standard neural ODEs often become unstable, which led us to use stabilised neural ODEs with an explicit damping term. We quantify this observation by counting, of the 16 models trained at each dimension d_h, how many become unstable with and without the presence of an explicit damping term. From our training data, we know where $\boldsymbol{h}$ should lie, so if it falls far outside this range after some time we can classify the model as unstable. In particular, we classify models as unstable if the norm of the final state is two times that of the maximum in our data ($\|\tilde{\boldsymbol{h}}(T)\| > 2 \max_t \|\boldsymbol{h}(t)\|$), after $T = 10^4$ time units. In all of the unstable cases $\|\tilde{\boldsymbol{h}}(t)\|$ follows the data over some short time range before eventually growing indefinitely.

In figure 8 we show the number of unstable models with and without damping. With damping, all of the models are stable, whereas without damping almost all models become unstable for $d_h = 5$–16, and around half become unstable in the other cases. In addition, with longer runs or with different initial conditions, many of the models without damping labelled as stable here also eventually become unstable. This lack of stability happens when inaccuracies in the neural ODE model push trajectories off the attractor. Once off the attractor, the model is presented with states unlike the training data leading to further growth in this error. We show more results highlighting this behaviour in

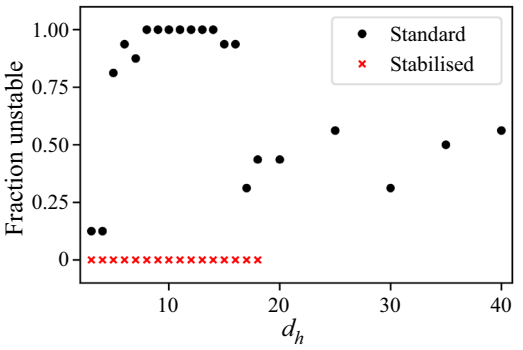


Figure 8. Fraction of unstable DManD models with standard neural ODEs and with stabilised neural ODEs at various dimensions.

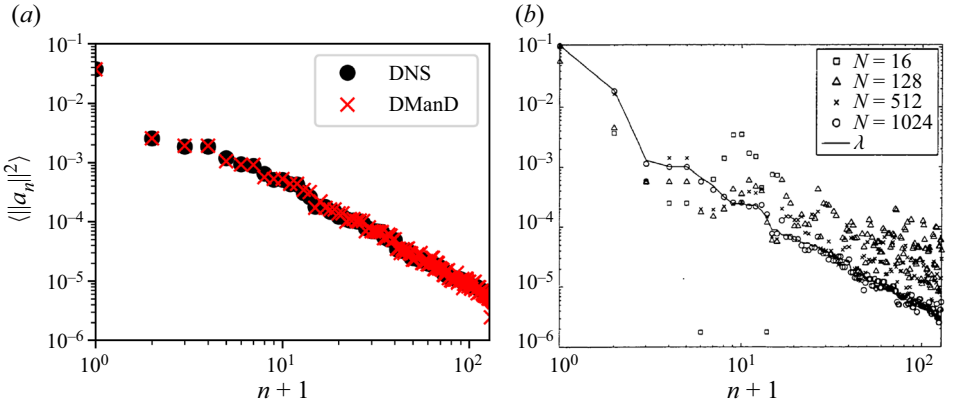


Figure 9. Comparison of $\langle \|a_n\|^2 \rangle$ (mean-squared POD coefficient amplitudes) from the DNS to (*a*) $\langle \|a_n\|^2 \rangle$ from the DManD model at $d_h = 18$ and (*b*) $\langle \|a_n\|^2 \rangle$ from POD-Galerkin with N POD modes (reproduced with permission from Gibson 2002). In (*b*), the quantity λ is equivalent to $\langle \|a_n\|^2 \rangle$ from the DNS.

Linot & Graham (2022) and Linot *et al.* (2023*a*). Thus, although some standard neural ODE models do provide reasonable statistics, using these models presents challenges due to this lack of robustness. As such, all other results we show use stabilised neural ODEs.

While figure 8 indicates that stabilised neural ODEs predict $\tilde{\boldsymbol{h}}$ in a similar range to that of the data, it does not quantify the accuracy of these predictions. In fact, with few dimensions many of these models do not remain chaotic, landing on fixed points or POs. The first metric we use to quantify the long-time performance of the DManD method is the mean-squared POD coefficient amplitudes ($\langle \|a_n\|^2 \rangle$). We consider this quantity because Gibson reports it for POD-Galerkin in Gibson (2002) at various levels of truncation. In figure 9 we show how well the DManD model, with $d_h = 18$, captures this quantity, in comparison to the POD-Galerkin model in Gibson (2002). The two data sets slightly differ because we subtract the mean before applying POD and Gibson did not. The DManD method, with only 18 degrees of freedom, matches the mean-squared amplitudes to high accuracy, far better than all of the POD-Galerkin models. It is not until POD-Galerkin keeps 1024 modes that the results become comparable, which corresponds to $\sim$2000 degrees of freedom because most coefficients are complex. In addition, our method requires only data, whereas the POD-Galerkin approach requires both data for

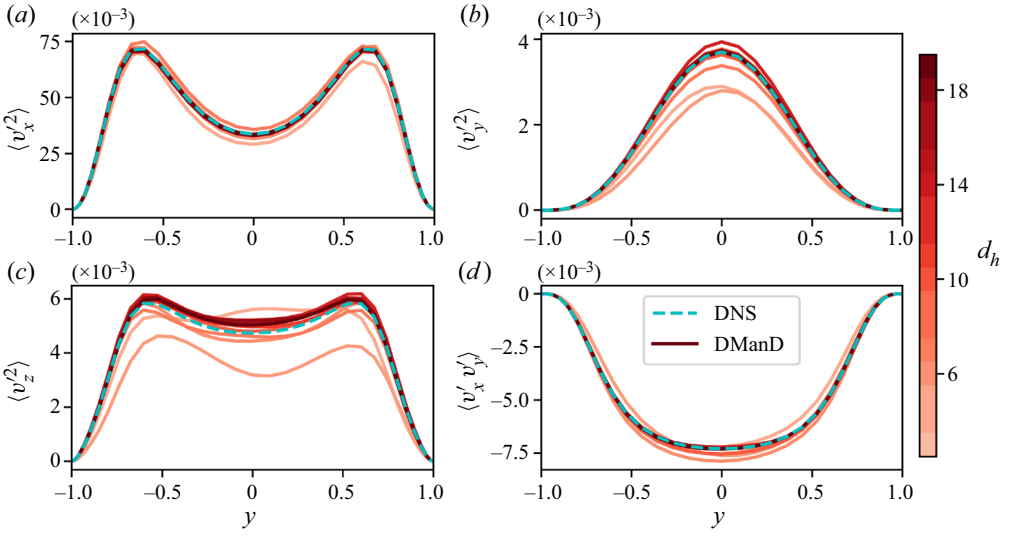


Figure 10. Components of the Reynolds stress with increasing dimension for DManD models at various dimensions. Odd numbers above $d_h = 5$ are omitted for clarity.

computing the POD and knowledge of the equations of motion for projecting the equations onto these modes.

We now investigate how the Reynolds stress and the power input versus dissipation vary with dimension. Figure 10 shows four components of the Reynolds stress at various dimensions. For $\langle v_x'^2 \rangle$ and $\langle v_x' v_y' \rangle$, nearly all the models match the data, with relatively small deviations only appearing for $d_h \sim 3$–6. For $\langle v_y'^2 \rangle$ and $\langle v_z'^2 \rangle$, this deviation becomes more obvious, and the lines do not converge until around $d_h > 10$, with all models above this dimension exhibiting a minor overprediction in $\langle v_z'^2 \rangle$.

To evaluate how accurate the models are at reconstructing the energy balance, we look at joint PDFs of power input and dissipation. The power input is the amount of energy required to move the walls:

$$I = \frac{1}{2L_x L_z} \int_0^{L_x} \int_0^{L_z} \left. \frac{\partial v_x}{\partial y} \right|_{y=-1} + \left. \frac{\partial v_x}{\partial y} \right|_{y=1} \mathrm{d}x\, \mathrm{d}z, \tag{3.14}$$

and the dissipation is the energy lost to heat due to viscosity:

$$D = \frac{1}{2L_x L_z} \int_0^{L_x} \int_{-1}^{1} \int_0^{L_z} |\nabla \times \boldsymbol{v}|^2 \, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z. \tag{3.15}$$

These two terms define the rate of change of energy in the system $\dot{E} = (I - D)/Re$ (Kaszás, Cenedese & Haller 2022), which must average to zero over long times. Checking this statistic is important to show the DManD models correctly balance the energy.

Figure 11(*a–c*) show the PDF from the DNS, the PDF for $d_h = 6$ and the PDF for $d_h = 18$, generated from a single trajectory evolved for 5000 time units, and figures 11(*e,f*) show the absolute difference between the true and model PDFs. With $d_h = 6$ the model overestimates the number of low dissipation states, while $d_h = 18$ matches the density well. In figure 11(*d*) we compare the joint PDFs at all dimensions with the true PDF using the earth movers distance (EMD) (Rubner, Tomasi & Guibas 1998). The EMD determines the distance between two PDFs as a solution to the *transportation problem* by treating

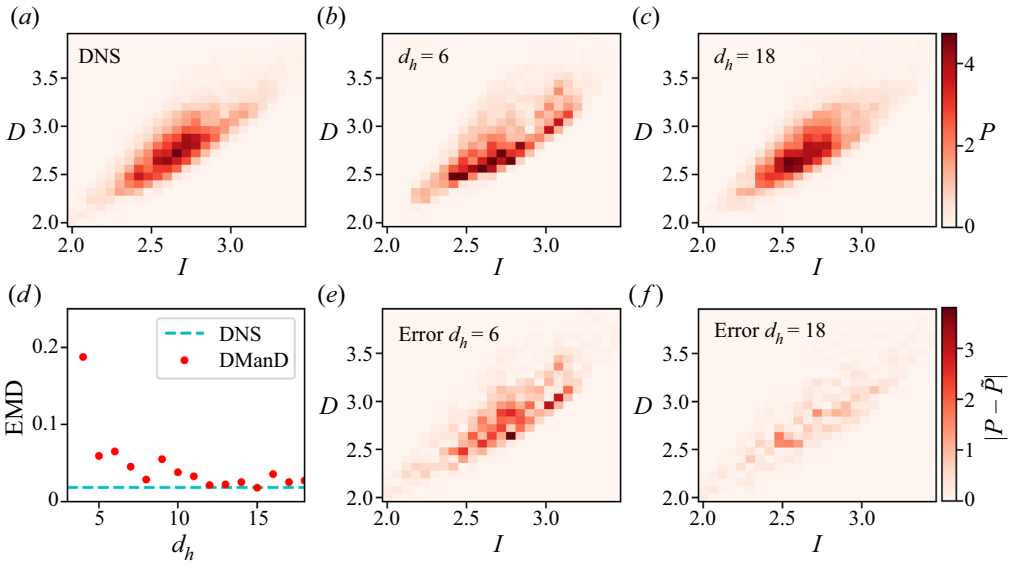


Figure 11. (*a–c*) Examples of joint PDFs for the true system, the DManD model at $d_h = 6$ and the DManD model at $d_h = 18$. (*d*) EMD between the PDF from the DNS and the PDFs predicted by the DManD model at various dimensions. 'DNS' is the error between two PDFs generated from DNS trajectories of the same length with different initial conditions. (*e*,*f*) The error associated with the DManD model PDFs at $d_h = 6$ and $d_h = 18$.

the true PDF as 'supplies' and the model PDF as 'demands' and finding the 'flow' that minimises the work required to move one to the other. Specifically, we find the flow f_{ij} that minimises $\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}d_{ij}$ subject to the constraints:

$$f_{ij} \geq 0, \quad 1 \leq i \leq m, \ 1 \leq j \leq n, \tag{3.16}$$

$$\sum_{j=1}^{n} f_{ij} = p_i, \quad 1 \leq i \leq m, \tag{3.17}$$

$$\sum_{i=1}^{m} f_{ij} = q_j, \quad 1 \leq j \leq n, \tag{3.18}$$

where p_i is the probability density at the ith bin in the model PDF and q_j is the probability density at the jth bin in the true PDF, for PDFs with n and m bins (in this case $n = m$). In addition, d_{ij} is the cost to move between bins, which we take to be the L_2 distance between bins i and j (for $i = j$ $d_{ij} = 0$). After solving this minimisation problem for the optimal flow f_{ij}^* the EMD is defined as

$$\text{EMD} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}^* d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}^*}. \tag{3.19}$$

For a more detailed explanation of the EMD we refer the reader to Levina & Bickel (2001).

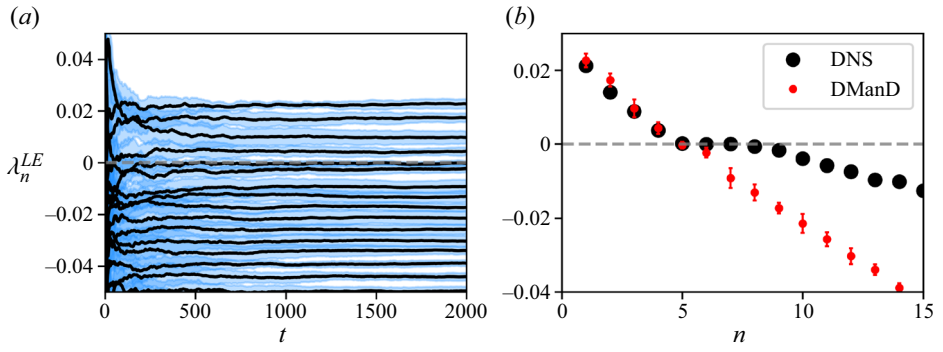


Figure 12. (*a*) The mean (black) and standard deviation (blue) of the Lyapunov exponents computed with the DManD model through time for 10 random initial conditions. The grey dashed line identifies $\lambda = 0$. (*b*) Lyapunov exponents computed from the model and from the DNS (Inubushi *et al.* 2015).

We compute the distance between PDFs using the EMD because it is a *cross-bin* distance, meaning the distance accounts for the density in neighbouring bins. This is in contrast to *bin-to-bin* distances, such as the Kullback–Leibler divergence, which only uses the error at a given bin. Bin-to-bin distances can vary significantly with small shifts in one PDF (misalignment) and when changing the number of bins used to generate the PDF (Ling & Okada 2007). We choose the EMD because it does not suffer from these issues. In figure 11(*d*) we see a steep drop in the EMD at $d_h = 5$ and after $d_h \gtrsim 15$ the joint PDFs are in excellent agreement with the DNS. The dashed line corresponds to the EMD between two different trajectories from the DNS. An interesting observation is that the DManD models appear to accurately capture the statistics of the DNS for $d_h \gtrsim 15$, and Inubushi *et al.* (2015) reported a Lyapunov dimension of 14.8. The close agreement between this fractal dimension and the dimension at which our models perform well further supports that, in theory, for the models with $d_h \gtrsim 15$ we have a sufficient number of dimensions to 'exactly' parameterise the manifold on which the dynamics lie.

Finally, we investigate the Lyapunov exponents of the DManD models. In figure 12, we report the spectrum of Lyapunov exponents λ_n^{LE} for the DManD model with $d_h = 18$, as well as the leading values computed by Inubushi *et al.* (2015) from DNS. Figure 12(*a*) shows the prediction of the Lyapunov exponents through time in the DManD model, averaged over 10 initial conditions using the methods described in Sandri (1996) with the code made available by Rozdeba (2017). At around 2000 time units we see that the Lyapunov exponents have nearly converged, with relatively small discrepancies across trials, as seen by the standard deviation. In figure 12(*b*), we report the Lyapunov spectra for the DManD model and the DNS. The DManD model correctly predicts that there are four positive exponents, and the computed values are in very good agreement with those from the DNS. Along with the positive exponents, Inubushi *et al.* (2015) identified that there were three zero exponents associated with the time translation symmetry and the spatial translational symmetries in x and z. We see that the DManD model reasonably predicts two of these zero Lyapunov exponents, but lacks the third due to the phase alignment of the model. The negative values are not in such good agreement, probably for one or both of two reasons. First, the DManD model considers only the dynamics on the low-dimensional invariant manifold of the long-time dynamics, so cannot capture Lyapunov exponents corresponding to directions off of this manifold. Second, the stabilisation term included in the neural ODE representation may lead to some additional damping. Nevertheless, the quantitative prediction of both the number and magnitude of the unstable and neutrally

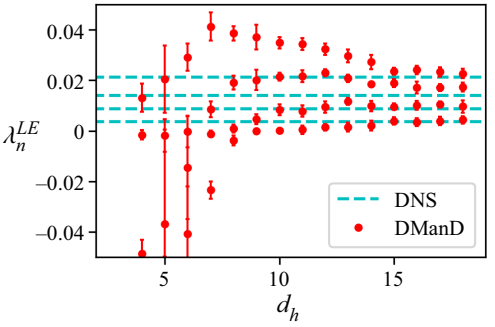


Figure 13. The four leading Lyapunov exponents for the DManD models at various dimensions and for the DNS.

stable Lyapunov exponents indicates the fidelity of the DManD model with regard to the chaotic dynamics of the flow.

As with the other statistics, we may again investigate the exponents as we vary the DManD model dimension. Figure 13 shows the four leading Lyapunov exponents (all positive in the DNS) for the models as we vary the dimension. Again, we compute these Lyapunov exponents from 10 initial conditions evolved forward 2000 time units. The positive Lyapunov exponents may be used to compute the Kolmogorov–Sinai entropy $\mathrm{KS} = \Sigma_{\lambda_n^{LE}>0}\lambda_n^{LE}$. This is a measure of the information created by a dynamical system (Eckmann & Ruelle 1985). As we increase the model dimension, the estimate of these Lyapunov exponents improves in two different ways. First, from $d_h = 4$–8 the number of positive Lyapunov exponents increases from one to three, recovering the correct number of four positive Lyapunov exponents at $d_h = 9$. Second, from $d_h = 9$–15 we see a gradual decrease in the two highest Lyapunov exponents and a gradual increase in the other two positive Lyapunov exponents. After $d_h \gtrsim 15$ the accuracy of the Lyapunov exponents only shows mild improvement, further supporting the conclusion that near $d_h = 15$ we have a sufficient number of dimensions. At $d_h = 18$ (the most accurate model) the Kolmogorov–Sinai entropy of the DManD model is $\mathrm{KS} = 0.054$ which is close to the DNS value of $\mathrm{KS} = 0.048$ (Inubushi *et al.* 2015).

3.5. *Finding ECS in the model*

Now that we know that the DManD model quantitatively captures many of the key characteristics of MFU PCF, we now want to explore using the model to discover ECS. In particular, we first investigate whether known POs of the DNS exist in the DManD model, and then we use the DManD model to search for new POs. Finding good initial conditions for use in a PO search is a challenging task, for example, Page & Kerswell (2020) compared using DManD with recurrent flow analysis for this task. Here we show that the DManD model offers a rapid method for finding useful initial conditions to input into the full DNS ECS solver. As our model predicts phase-aligned dynamics, the POs of the DManD model are either POs or RPOs, depending on the phase evolution, which we have not tracked. In the following, we omit all $\tilde{\cdot}$ for clarity, so all functions should be assumed to come from a DManD model.

Here we outline the approach we take to find POs, which follows Cvitanović *et al.* (2016). When searching for POs we seek an initial condition to a trajectory that repeats

after some time period. This is equivalent to finding the zeros of

$$\boldsymbol{H}(\boldsymbol{h}, T) = \boldsymbol{G}_T(\boldsymbol{h}) - \boldsymbol{h}, \tag{3.20}$$

where $\boldsymbol{G}_T(\boldsymbol{h})$ is the flow map forward T time units from $\boldsymbol{h}$, i.e. $\boldsymbol{G}_T(\boldsymbol{h}(t)) = \boldsymbol{h}(t+T)$. We compute $\boldsymbol{G}_T(\boldsymbol{h})$ from (2.9). Finding zeros to (3.20) requires that we find both a point $\boldsymbol{h}^*$ on the PO and a period T^* such that $\boldsymbol{H}(\boldsymbol{h}^*, T^*) = 0$. One way to find $\boldsymbol{h}^*$ and T^* is by using the Newton–Raphson method.

By performing a Taylor series expansion of $\boldsymbol{H}$ we find near the fixed point $\boldsymbol{h}^*, T^*$ of $\boldsymbol{H}$ that

$$\left.\begin{aligned} \boldsymbol{H}(\boldsymbol{h}^*, T^*) - \boldsymbol{H}(\boldsymbol{h}, T) &\approx \boldsymbol{D}_h\boldsymbol{H}(\boldsymbol{h}, T)\Delta\boldsymbol{h} + \boldsymbol{D}_T\boldsymbol{H}(\boldsymbol{h}, T)\Delta T, \\ -\boldsymbol{H}(\boldsymbol{h}, T) &\approx \boldsymbol{D}_h\boldsymbol{H}(\boldsymbol{h}, T)\Delta\boldsymbol{h} + \boldsymbol{g}(\boldsymbol{G}_T(\boldsymbol{h}))\Delta T, \end{aligned}\right\} \tag{3.21}$$

where $\boldsymbol{D}_h$ is the Jacobian of $\boldsymbol{H}$ with respect to $\boldsymbol{h}$, $\boldsymbol{D}_T$ is the Jacobian of $\boldsymbol{H}$ with respect to the period T, $\Delta\boldsymbol{h} = \boldsymbol{h}^* - \boldsymbol{h}$ and $\Delta T = T^* - T$. To have a complete set of equations for $\Delta\boldsymbol{h}$ and ΔT, we supplement (3.21) with the constraint that the updates $\Delta\boldsymbol{h}$ are orthogonal to the vector field at $\boldsymbol{h}$: i.e.

$$\boldsymbol{g}(\boldsymbol{h})^{\mathrm{T}}\Delta\boldsymbol{h} = 0. \tag{3.22}$$

With this constraint, at Newton step (i), the system of equations becomes

$$\begin{bmatrix} \boldsymbol{D}_{h^{(i)}}\boldsymbol{H}(\boldsymbol{h}^{(i)}, T^{(i)}) & \boldsymbol{g}(\boldsymbol{G}_{T^{(i)}}(\boldsymbol{h}^{(i)})) \\ \boldsymbol{g}(\boldsymbol{h}^{(i)})^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{h}^{(i)} \\ \Delta T^{(i)} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{H}(\boldsymbol{h}^{(i)}, T^{(i)}) \\ 0 \end{bmatrix}, \tag{3.23}$$

which, in the standard Newton–Raphson method, is used to update the guesses $\boldsymbol{h}^{(i+1)} = \boldsymbol{h}^{(i)} + \Delta\boldsymbol{h}^{(i)}$ and $T^{(i+1)} = T^{(i)} + \Delta T^{(i)}$.

Typically, a Newton–Krylov method is used to avoid explicitly constructing the Jacobian (Viswanath 2007). However, with DManD, computing the Jacobian is simple, fast and requires little memory because the state representation is dramatically smaller in the DManD model than in the DNS. We compute the Jacobian $\boldsymbol{D}_h\boldsymbol{H}(\boldsymbol{h}, T)$ directly, with the same automatic differentiation tools used for training the neural ODE. Furthermore, if we had chosen to represent the dynamics in discrete, rather than continuous time, computation of general POs would not be possible, as the period T can take on arbitrary values and a discrete-time representation would limit T to multiples of the time step. When finding POs of the DManD model we used the SciPy 'hybr' method, which uses a modification of the Powell hybrid method (Virtanen *et al.* 2020), and for finding POs of the DNS we used the Newton GMRES-Hookstep method built into Channelflow (Gibson *et al.* 2021). In the following trials we only consider DManD models with $d_h = 18$.

For the HKW cell there exists a library of POs made available by Gibson *et al.* (2008*b*). To investigate whether the DManD model finds POs similar to existing solutions, we took states from the known POs, encoded them and used this as an initial condition in the DManD Newton solver to find POs in the model. In figure 14 we show projections of 12 known POs, which we identify by the period T, and compare them with projections of POs found using the DManD model. This makes up a majority of the POs made available by Gibson *et al.* (2008*b*). Of the other known solutions, three are RPOs with phase shifts in the streamwise direction that our model, with the current set-up, cannot capture. The other two have short periods of $T = 19.02$ and $T = 19.06$. Most of the POs found with DManD land on initial conditions near that of the DNS and follow similar trajectories to the DNS.

How close many of these trajectories are to the true PO is surprising and encouraging for many reasons. First, the data used for training the DManD model does not explicitly

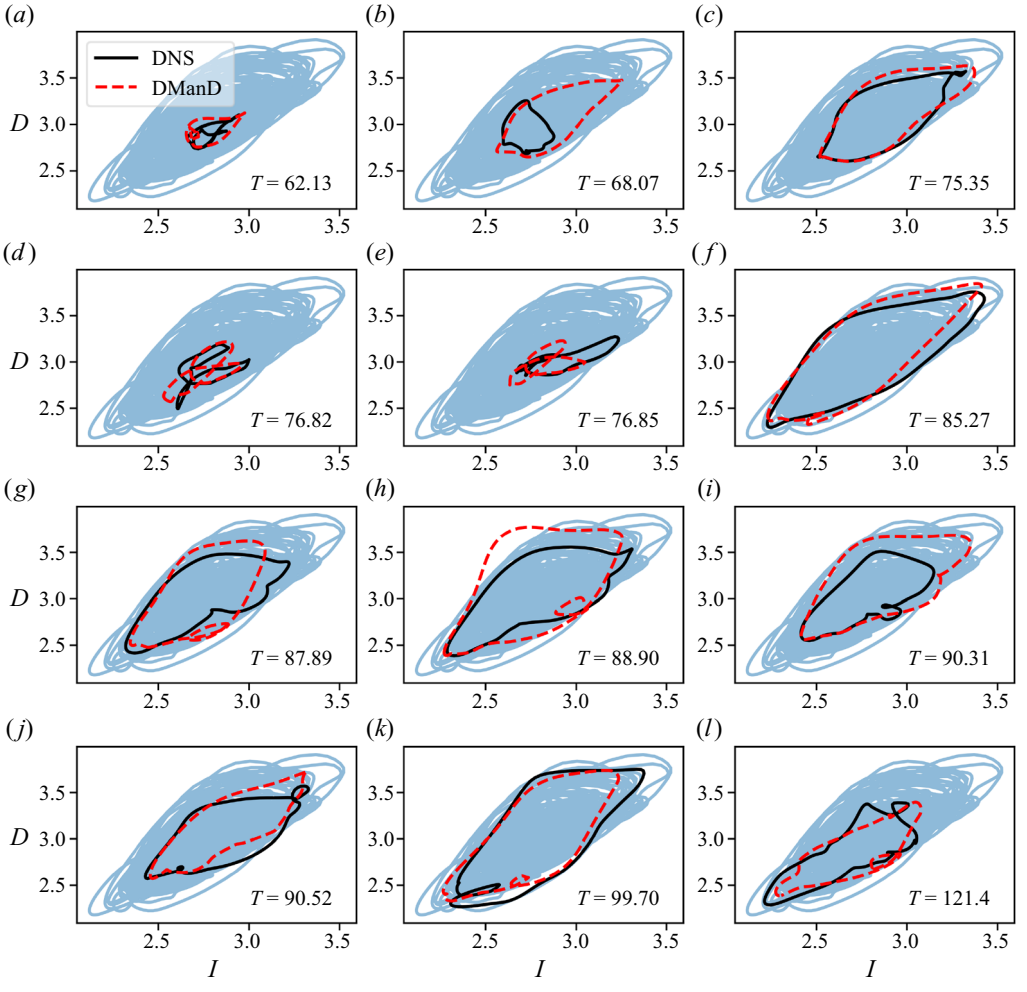


Figure 14. Power input versus dissipation of known POs (period reported in bottom right) from the DNS and POs found in the DManD model at $d_h = 18$. The blue line is a long trajectory of the DNS for comparison.

contain any POs. Second, this approach by no means guarantees convergence on a PO in the DManD model. Third, starting with an initial condition from a PO does not necessarily mean that the solution the Newton solver lands on will be the closest PO to that initial condition, so there may exist POs in the DManD model closer to the DNS solutions than what we present here.

Now that we know the DManD model can find POs similar to those known to exist for the DNS, we now use it to search for new POs. First, we searched for POs in three of the $d_h = 18$ models by randomly selecting 20 initial conditions and selecting 4 different periods $T = [20, 40, 60, 80]$. We then took the initial conditions and periods for converged POs and decoded and upsampled them onto a $48 \times 49 \times 48$ grid. We performed this upsampling because Viswanath (2007) reported that solutions on the coarser grid can be computational artifacts. Finally, we put these new initial conditions into Channelflow and ran another Newton search for 100 iterations. This procedure resulted in us finding nine new RPOs and three existing POs, the details of which we include in table 2.

https://doi.org/10.1017/jfm.2023.720 Published online by Cambridge University Press

Label	1	2	3	4	5	6	7	8	9	10	11	12
σ_z	1.91×10^{-1}	-9.66×10^{-2}	-1.77×10^{-3}	1.15×10^{-1}	-9.21×10^{-3}	-1.90×10^{-1}	-1.28×10^{-2}	-1.19×10^{-1}	-5.63×10^{-5}	4.64×10^{-14}	2.17×10^{-14}	2.73×10^{-13}
T	37.94	84.25	91.29	82.07	74.14	41.24	110.67	83.31	64.64	19.06	68.07	75.35
Error	2.23×10^{-3}	1.01×10^{-3}	3.92×10^{-3}	2.84×10^{-3}	1.87×10^{-3}	5.26×10^{-4}	1.25×10^{-3}	1.13×10^{-3}	2.25×10^{-3}	1.57×10^{-4}	2.55×10^{-4}	1.07×10^{-4}

Table 2. Details on the RPOs and POs found using initial conditions from the DManD model. The first nine solutions are new and the last three had previously been found. 'Label' indicates the label in figure 15, σ_z corresponds to the phase-shift in z, T is the period of the orbit and 'Error' is $\|\text{shifted final state} - \text{initial state}\|/\|\text{initial state}\|$, which is the same error as in Viswanath (2007).

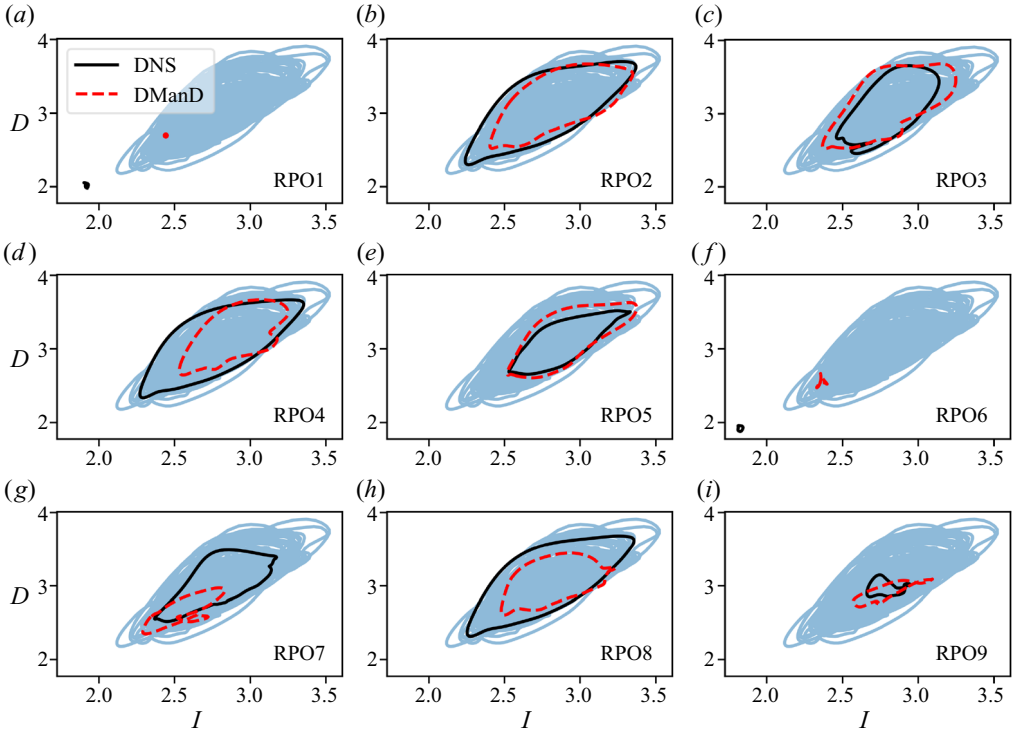


Figure 15. Power input versus dissipation of new POs found in the DManD model at $d_h = 18$ and used to find solutions in the DNS (periods reported in table 2). The blue line is a long trajectory of the DNS for comparison.

In figure 15 we show the new RPOs in the DManD model and the RPOs they converged to after putting them into the Channelflow Newton solver as initial guesses. Again, many of the RPOs end up following a similar path through this state space, with the biggest exceptions being RPO1 and RPO6, which converged to low-power input solutions. It is worth noting that this worked well, considering that the DManD initial conditions are POD coefficients from a model trained using data on a coarser grid than used to search for these solutions and considering Newton–Raphson methods can converge to solutions far from the initial condition (as in RPO1 and RPO6). We have described a new method to rapidly find new ECS, wherein an accurate low-dimensional model, such as the DManD model presented here, is used to quickly perform a large number of ECS searches in the model, and then these solutions can be fine-tuned in the full simulation to land on new solutions.

4. Conclusion

In the present work we have described DManD and applied it for accurate modelling of MFU PCF with far fewer degrees of freedom ($O(10)$) than required for the DNS ($O(10^5)$). The DManD method consists of first finding a low-dimensional parameterisation of the manifold on which data lies, and then discovering an ODE to evolve this low-dimensional state representation forward in time. In both cases, we use NNs to approximate these functions from data. We find that an extremely low-dimensional parameterisation of this manifold can be found using an autoencoder. Then, we use stabilised neural ODEs to accurately evolve the low-dimensional state forward in time. Although we used NNs for

the function approximations and applied the method to MFU turbulence, the approach is general in that any method can be used for the function approximation and it applies to any system with long-time dynamics that lie on a manifold. Applying the method to another parameter regime, for example a higher Reynolds number, involves acquiring new data and varying the dimension d_h such that the dynamics are appropriately captured, along with the hyperparameters of the NNs. In addition, parameter dependency could be incorporated directly by inputting the parameters into the autoencoder (e.g. $\boldsymbol{h} = \boldsymbol{\chi}(\boldsymbol{u}, Re; \boldsymbol{\theta}_E)$).

The DManD model captures the SSP and accurately tracks trajectories and the temporal autocorrelation over short time horizons. For DManD models with $d_h \gtrsim 15$ we found excellent agreement between the model and the DNS in computing the mean-squared POD coefficient amplitude, the Reynolds stress and the joint PDF of power input versus dissipation. For comparison, we showed that a POD-Galerkin model requires $\sim$2000 degrees of freedom to get similar performance in matching the mean-squared POD coefficient amplitudes. Finally, we used the DManD model at $d_h = 18$ for PO searches. Using a set of existing POs, we successfully landed on nearby POs in the model. Finally, we found nine previously undiscovered RPOs by first finding solutions in the DManD model and then using those as initial guesses to search in the full DNS. Future work will be necessary to examine how well the present approach will extend to higher Reynolds numbers where a spectrum of turbulent length scales arises.

The results reported here have both fundamental and technological importance. At the fundamental level, they indicate that the true dimension of the dynamics of a turbulent flow can be orders of magnitude smaller than the number of degrees of freedom required for a fully resolved simulation. Technologically this point is important because it may enable, for example, highly sophisticated model-based nonlinear control algorithms to be used: determining the control strategy from the low-dimensional DManD model rather than a full-scale DNS, and applying it to the full flow will speed up both learning and implementing a control policy (Zeng, Linot & Graham 2022*a*, ; Linot, Zeng & Graham 2023*b*).

Funding. This work was supported by the Air Force Office of Scientific Research (grant no. FA9550-18-1-0174) and the Office of Naval Research (grant no. N00014-18-1-2865 (Vannevar Bush Faculty Fellowship)).

Declaration of interests. The authors report no conflict of interest.

Data availability. The code that supports the findings of this study is openly available at https://github.com/alinot5/DManDCouette.git

Author ORCIDs.
Alec J. Linot https://orcid.org/0000-0002-4591-7133;
Michael D. Graham https://orcid.org/0000-0003-4983-4949.

References

Arndt, R.E.A., Long, D.F. & Glauser, M.N. 1997 The proper orthogonal decomposition of pressure fluctuations surrounding a turbulent jet. *J. Fluid Mech.* **340**, 1–33.

Ball, K.S., Sirovich, L. & Keefe, L.R. 1991 Dynamical eigenfunction decomposition of turbulent channel flow. *Intl J. Numer. Meth. Fluids* **12** (6), 585–604.

Borrelli, G., Guastoni, L., Eivazi, H., Schlatter, P. & Vinuesa, R. 2022 Predicting the temporal dynamics of turbulent channels through deep learning. *Intl J. Heat Fluid Flow* **96**, 109010.

Budanur, N.B., Borrero-Echeverry, D. & Cvitanović, P. 2015*a* Periodic orbit analysis of a system with continuous symmetry – a tutorial. *Chaos* **25** (7), 073112.

Budanur, N.B., Cvitanović, P., Davidchack, R.L. & Siminos, E. 2015*b* Reduction of SO(2) symmetry for spatially extended dynamical systems. *Phys. Rev. Lett.* **114**, 084102.

Chen, R.T.Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. 2019 Neural ordinary differential equations. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, QC.

Chollet, F., *et al.* 2015 Keras. Available at: https://keras.io.

Coifman, R.R., Lafon, S., Lee, A.B., , Maggioni, M., Nadler, B., Warner, F. & Zucker, S.W. 2005 Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps. *Proc. Natl Acad. Sci. USA* **102** (21), 7426–7431.

Cvitanović, P., Artuso, R., Mainieri, R., Tanner, G. & Vattay, G. 2016 *Chaos: Classical and Quantum*. Niels Bohr Institute.

Dormand, J.R. & Prince, P.J. 1980 A family of embedded Runge–Kutta formulae. *J. Comput. Appl. Maths* **6** (1), 19–26.

Eckmann, J.P. & Ruelle, D. 1985 Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.* **57**, 617–656.

Eivazi, H., Guastoni, L., Schlatter, P., Azizpour, H. & Vinuesa, R. 2021 Recurrent neural networks and Koopman-based frameworks for temporal predictions in a low-order model of turbulence. *Intl J. Heat Fluid Flow* **90**, 108816.

Eivazi, H., Veisi, H., Naderi, M.H. & Esfahanian, V. 2020 Deep neural networks for nonlinear model order reduction of unsteady flows. *Phys. Fluids* **32** (10), 105104.

Floryan, D. & Graham, M.D. 2022 Data-driven discovery of intrinsic dynamics. *Nat. Mach. Intell.* **4** (12), 1113–1120.

Foias, C., Jolly, M.S., Kevrekidis, I.G., Sell, G.R. & Titi, E.S. 1988 On the computation of inertial manifolds. *Phys. Lett. A* **131** (7–8), 433–436.

García-Archilla, B., Novo, J. & Titi, E.S. 1998 Postprocessing the Galerkin method: a novel approach to approximate inertial manifolds. *SIAM J. Numer. Anal.* **35** (3), 941–972.

Gibson, J.F. 2002 Dynamical systems models of wall-bounded, shear-flow turbulence. PhD thesis, Cornell University, New York.

Gibson, J.F. 2012 Channelflow: a spectral Navier–Stokes simulator in C++. pp. 1–41. University of New Hampshire.

Gibson, J.F., *et al.* 2021 Channelflow 2.0. arXiv:channelflow.ch

Gibson, J.F., Halcrow, J. & Cvitanović, P. 2008*a* Visualizing the geometry of state space in plane Couette flow. *J. Fluid Mech.* **611** (1987), 107–130.

Gibson, J.F., Halcrow, J., Cvitanović, P. & Viswanath, D. 2008*b* Heteroclinic connections in plane Couette flow. *J. Fluid Mech.* **621**, 365–376.

Goodfellow, I., Bengio, Y. & Courville, A. 2016 *Deep Learning*. MIT. Available at: http://www.deeplearningbook.org.

Graham, M.D. & Floryan, D. 2021 Exact coherent states and the nonlinear dynamics of wall-bounded turbulent flows. *Annu. Rev. Fluid Mech.* **53** (1), 227–253.

Graham, M.D., Steen, P.H. & Titi, E.S. 1993 Computational efficiency and approximate inertial manifolds for a Bénard convection system. *J. Nonlinear Sci.* **3** (1), 153–167.

Hamilton, J., Kim, J. & Waleffe, F. 1995 Regeneration mechanisms of near-wall turbulence structures. *J. Fluid Mech.* **287**, 317–348.

Hasegawa, K., Fukami, K., Murata, T. & Fukagata, K. 2020*a* CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers. *Fluid Dyn. Res.* **52** (6), 065501.

Hasegawa, K., Fukami, K., Murata, T. & Fukagata, K. 2020*b* Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theor. Comput. Fluid Dyn.* **34** (4), 367–383.

Hinton, G. & Roweis, S. 2003 Stochastic neighbor embedding. *Adv. Neural Inform. Proc. Syst.* **15**, 833–840.

Hinton, G.E. & Salakhutdinov, R.R. 2006 Reducing the dimensionality of data with neural networks. *Science* **313** (5786), 504–507.

Holmes, P., Lumley, J.L., Berkooz, G. & Rowley, C.W. 2012 *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press.

Hopf, E. 1948 A mathematical example displaying features of turbulence. *Commun. Pure Appl. Maths* **1** (4), 303–322.

Inubushi, M., Takehiro, S.-i. & Yamada, M. 2015 Regeneration cycle and the covariant Lyapunov vectors in a minimal wall turbulence. *Phys. Rev.* E **92**, 023022.

Jiménez, J. & Moin, P. 1991 The minimal flow unit in near-wall turbulence. *J. Fluid Mech.* **225**, 213–240.

Kaszás, B., Cenedese, M. & Haller, G. 2022 Dynamics-based machine learning of transitions in Couette flow. *Phys. Rev. Fluids* **7**, L082402.

KAWAHARA, G. & KIDA, S. 2001 Periodic motion embedded in plane Couette turbulence: regeneration cycle and burst. *J. Fluid Mech.* **449**, 291–300.

KAWAHARA, G., UHLMANN, M. & VAN VEEN, L. 2012 The significance of simple invariant solutions in turbulent flows. *Annu. Rev. Fluid Mech.* **44** (1), 203–225.

KINGMA, D.P. & BA, J.L. 2015 Adam: a method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15.

KLEISER, L. & SCHUMANN, U. 1980 *Treatment of Incompressibility and Boundary Conditions in 3-D Numerical Spectral Simulations of Plane Channel Flows*, pp. 165–173. Vieweg+Teubner.

LEE, K. & CARLBERG, K.T. 2020 Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.* **404**, 108973.

LEVINA, E. & BICKEL, P. 2001 The earth mover's distance is the Mallows distance: some insights from statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 251–256. IEEE.

LING, H. & OKADA, K. 2007 An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Trans. Pattern Anal. Mach. Intell.* **29** (5), 840–853.

LINOT, A.J., BURBY, J.W., TANG, Q., BALAPRAKASH, P., GRAHAM, M.D. & MAULIK, R. 2023*a* Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems. *J. Comput. Phys.* **474**, 111838.

LINOT, A.J. & GRAHAM, M.D. 2020 Deep learning to discover and predict dynamics on an inertial manifold. *Phys. Rev.* E **101**, 062209.

LINOT, A.J. & GRAHAM, M.D. 2022 Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations. *Chaos* **32** (7), 073110.

LINOT, A.J., ZENG, K. & GRAHAM, M.D. 2023*b* Turbulence control in plane Couette flow using low-dimensional neural ODE-based models and deep reinforcement learning. *Intl J. Heat Fluid Flow* **101**, 109139.

MILANO, M. & KOUMOUTSAKOS, P. 2002 Neural network modeling for near wall turbulent flow. *J. Comput. Phys.* **182** (1), 1–26.

MOEHLIS, J., FAISST, H. & ECKHARDT, B. 2004 A low-dimensional model for turbulent shear flows. *New J. Phys.* **6** (1), 56.

MOEHLIS, J., SMITH, T.R., HOLMES, P. & FAISST, H. 2002 Models for turbulent plane Couette flow using the proper orthogonal decomposition. *Phys. Fluids* **14** (7), 2493–2507.

MOIN, P. & MOSER, R.D. 1989 Characteristic-eddy decomposition of turbulence in a channel. *J. Fluid Mech.* **200**, 471–509.

MURATA, T., FUKAMI, K. & FUKAGATA, K. 2020 Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *J. Fluid Mech.* **882**, A13.

NAGATA, M. 1990 Three-dimensional finite-amplitude solutions in plane Couette flow: bifurcation from infinity. *J. Fluid Mech.* **217**, 519–527.

NAIR, N.J. & GOZA, A. 2020 Leveraging reduced-order models for state estimation using deep learning. *J. Fluid Mech.* **897**, 1–13.

NAKAMURA, T., FUKAMI, K., HASEGAWA, K., NABAE, Y. & FUKAGATA, K. 2021 Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Phys. Fluids* **33** (2), 025116.

PAGE, J., BRENNER, M.P. & KERSWELL, R.R. 2021 Revealing the state space of turbulence using machine learning. *Phys. Rev. Fluids* **6**, 034402.

PAGE, J. & KERSWELL, R.R. 2020 Searching turbulence for periodic orbits with dynamic mode decomposition. *J. Fluid Mech.* **886**, A28.

PASZKE, A., *et al.* 2019 PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates.

PÉREZ DE JESÚS, C.E. & GRAHAM, M.D. 2023 Data-driven low-dimensional dynamic model of Kolmogorov flow. *Phys. Rev. Fluids* **8** (4), 044402.

PEYRET, R. 2002 *Spectral Methods for Incompressible Viscous Flow*. Springer.

PORTWOOD, G.D., *et al.* 2019 Turbulence forecasting via neural ODE. arXiv:1911.05180

REMPFER, D. & FASEL, H.F. 1994 Evolution of three-dimensional coherent structures in a flat-plate boundary layer. *J. Fluid Mech.* **260**, 351–375.

ROJAS, C.J.G., DENGEL, A. & RIBEIRO, M.D. 2021 Reduced-order model for fluid flows via neural ordinary differential equations. arXiv:2102.02248

ROWEIS, S.T. & SAUL, L.K. 2000 Nonlinear dimensionality reduction by locally linear embedding. *Science* **290** (5500), 2323–2326.

ROZDEBA, P. 2017 pyLyapunov. Available at: https://github.com/paulrozdeba/pyLyapunov

RUBNER, Y., TOMASI, C. & GUIBAS, L.J. 1998 A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 59–66. IEEE.

SANDRI, M. 1996 Numerical calculation of Lyapunov exponents. *Math. J.* **6**, 78–84.

SCHÖLKOPF, B., SMOLA, A. & MÜLLER, K.-R. 1998 Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* **10** (5), 1299–1319.

SMITH, T.R., MOEHLIS, J. & HOLMES, P. 2005 Low-dimensional modelling of turbulence using the proper orthogonal decomposition: a tutorial. *Nonlinear Dyn.* **41** (1), 275–307.

SPALART, P.R., MOSER, R.D. & ROGERS, M.M. 1991 Spectral methods for the Navier–Stokes equations with one infinite and two periodic directions. *J. Comput. Phys.* **96** (2), 297–324.

SRINIVASAN, P.A., GUASTONI, L., AZIZPOUR, H., SCHLATTER, P. & VINUESA, R. 2019 Predictions of turbulent shear flows using deep neural networks. *Phys. Rev. Fluids* **4**, 054603.

TAKENS, F. 1981 Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980* (ed. David Rand & Lai-Sang Young), pp. 366–381. Springer.

TENENBAUM, J.B., DE SILVA, V. & LANGFORD, J.C. 2000 A global geometric framework for nonlinear dimensionality reduction. *Science* **290** (5500), 2319–2323.

TITI, E.S. 1990 On approximate inertial manifolds to the Navier–Stokes equations. *J. Math. Anal. Appl.* **149** (2), 540–557.

VAN DER MAATEN, L.J.P., POSTMA, E.O. & VAN DEN HERIK, H.J. 2009 Dimensionality reduction: a comparative review. *J. Mach. Learn. Res.* **10**, 1–41.

VIRTANEN, P., *et al.* 2020 SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272.

VISWANATH, D. 2007 Recurrent motions within plane Couette turbulence. *J. Fluid Mech.* **580**, 339–358.

VLACHAS, P.R., PATHAK, J., HUNT, B.R., SAPSIS, T.P., GIRVAN, M., OTT, E. & KOUMOUTSAKOS, P. 2020 Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191–217.

WALEFFE, F. 1997 On a self-sustaining process in shear flows. *Phys. Fluids* **9** (4), 883–900.

WALEFFE, F. 1998 Three-dimensional coherent states in plane shear flows. *Phys. Rev. Lett.* **81** (19), 4140–4143.

WAN, Z.Y., VLACHAS, P., KOUMOUTSAKOS, P. & SAPSIS, T. 2018 Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PLoS ONE* **13** (5), e0197704.

WHITNEY, H. 1936 Differentiable manifolds. *Ann. Maths* **37** (3), 645–680.

WHITNEY, H. 1944 The self-intersections of a smooth n-manifold in 2n-space. *Ann. Maths* **45** (2), 220–246.

YOUNG, C.D. & GRAHAM, M.D. 2023 Deep learning delay coordinate dynamics for chaotic attractors from partial observable data. *Phys. Rev.* E **107** (3), 034215.

YU, H., TIAN, X., WEINAN, E & QIANXIAO, L. 2021 Onsagernet: learning stable and interpretable dynamics using a generalized Onsager principle. *Phys. Rev. Fluids* **6**, 114402.

ZENG, K. & GRAHAM, M.D. 2023 Autoencoders for discovering manifold dimension and coordinates in data from complex dynamical systems. arXiv:2305.01090

ZENG, K., LINOT, A. & GRAHAM, M.D. 2022*a* Learning turbulence control strategies with data-driven reduced-order models and deep reinforcement learning. In *12th International Symposium on Turbulence and Shear Flow Phenomena (TSFP12) Osaka, Japan (Online)*.

ZENG, K., LINOT, A.J. & GRAHAM, M.D. 2022*b* Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning. *Proc. R. Soc.* A **478** (2267), 20220297.