Smith-Knight & Rayleigh-Knight Essay

# Learning Stochastic Dynamics with Probabilistic Neural Networks to study Zonal Jets

I. J. S. Shokar[1] - Supervised by P. H. Haynes[1] & R. R. Kerswell[1]

[1] *Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, United Kingdom*

January 2023

**Abstract**

Machine learning has proven to be an effective tool for identifying patterns in data, and increasingly has been applied to the field of fluid dynamics for both understanding and emulating fluid flows. In this study, we present a deep learning approach to deriving a reduced-order model of stochastically forced atmospheric zonal jets. The approach provides a four orders of magnitude speed-up in simulating the jets, over numerical integration, together with a lower-degrees-of-freedom latent representation of the system - used to yield insight into the underlying dynamics.

We consider the behaviour of zonal jets on a beta plane as represented by a two-dimensional model driven by stochastic forcing, which parameterises the turbulence due to baroclinic instability. This idealised model gives a useful analogue for week-to-week variations in the large-scale dynamics of the tropospheric midlatitude jet - the driver of European weather. As the time evolution of the jets depends both on the nonlinear two-way interaction between the mean flow and the eddies and, crucially, the time history of the stochastic forcing, the current state or recent history of the system does not predict the forward evolution but instead determines a distribution of possible time evolutions.

To model the flow, we utilise methods in manifold learning to learn a transformation to a latent representation of the system and then use a probabilistic neural network to model the stochastic latent dynamics. We verify the neural network's performance by comparing the statistical and spectral properties of an ensemble from the neural network, obtained via sampling in the latent space, with an ensemble of numerical integrations, with different realisations of the stochastic forcing - with identical initial conditions. To study jet variability, we use ensembles of trajectories in both the latent and observation space to quantify to what extent different system states are driven by deterministic or stochastic dynamics.

**Contributions** Section 1 and the appendices A.1-A.3 contain review material, with Appendix A.3, detailing the implementation of the Numerical Model used in this study, referencing appropriate sections from [1]. Section 2 constitutes original research conducted by myself under the invaluable guidance of my supervisors Peter Haynes and Rich Kerwsell, which builds upon the research of others outlined in Section 1. Appendix A.4 also details original research that was conducted as a supplementary study but does not fit with the work outlined in this essay. Section 3 constitutes my concluding remarks and avenues for further research, that are plausible within the scope of this doctoral study.

# 1 Introduction and Existing Literature

## 1.1 Research Motivation

Geophysical fluid dynamics (GFD) is the study of fluid motion in the Earth's atmosphere and oceans, where physical ingredients such as rotation and stratification permit a variety of wave motions. Understanding how these fluids flow, interact and influence each other is of crucial importance to predicting and studying weather and climate. Weather refers to the state of the atmosphere at a specific time and location, including variables such as temperature, precipitation or wind, while climate is the long-term average of weather patterns. The study of GFD provides insights into the physical processes that drive weather and climate and how they change over time.

The field of GFD is also crucial for the development of numerical models that can simulate and forecast weather and climate. These models, known as Global Climate Models (GCMs) [2], are based on the laws of physics, however, they are complex and extremely computationally expensive, resulting in many needing to be run on some of the world's largest supercomputers [3]. Advancements in hardware and the increase in the availability of data have enabled the accuracy and resolution of these models to steadily increase. However, computational constraints still require the integrations to take place on a reasonably coarse grid - current forecasts at ECMWF (the European Centre for Medium-Range Weather Forecasts) are run at a resolution of 10km [4], with many processes needing to be parameterised, or approximated if they take place at spatial or temporal scales smaller than that of the model resolution.

Parameterisations rely on semi-empirical physical principles and typically improve modelling when included in coarse-resolution climate models, however as they are simple approximations of the physics that takes place on sub-grid scales they often induce biases. There are questions over how these small-scale processes influence the behaviour of the large scales and whether current parameterisations are able to correctly model these, raising the question of which physical mechanisms should be retained in order to maintain the fidelity of these models [5].

Currently, it is the parameterisation schemes that constitute the primary source of uncertainty in GCMs and forecast models [6], and one reason for this is that high Reynolds number flows such as dynamics in the atmosphere are non-linear, turbulent and thus chaotic systems, as described by Lorentz [7], leading to a sensitive and complex coupling between the climate and these parameterised processes. In some cases, small incorrect assumptions made during the parameterisation scheme can result in very different representations of the approximated process [6]. With the domain of GFD being a multi-scale system, Lorentz also asserted that there is a finite horizon of predictability, even if our observational errors and associated uncertainties are reduced [8], motivating the requirement to produce ensemble forecasts or parameterisations, rather than single trajectories. Traditionally these parameterisations have been physics-based, however, we look to produce a data-driven modelling approach in the form of Reduced Order Models (ROMs) of atmospheric processes that provide the same information but at a reduced computational expense, allowing the generation of large ensembles through probabilistic modelling, as well as provide scope to include additional information or resolution, without increasing the computational cost beyond that of the original scheme.

The complexity of GCMs makes studying processes in isolation difficult, so here we use an idealised model, retaining only the essential ingredients necessary. Here we consider a barotropic, stochas-

tically forced turbulent flow on a beta plane [9]. This idealised model gives rise to dynamics of interest, including the spontaneous formation and equilibration of persistent zonal jets and cyclones. This idealised model gives a useful analogue for week-to-week variations in the large-scale dynamics of the tropospheric midlatitude jet - the driver of European weather.

Atmospheric jets are strong and persistent, longitudinally-aligned, alternating currents in the upper troposphere, known for their strong winds and intense weather patterns. Acting like fast-moving rivers, they play a fundamental role in the climate system, transporting geophysically-important quantities, such as momentum, heat and tracers, including chemically and thermodynamically important quantities such as ozone and water vapour [1]. Changes in jet streams, caused by increasing greenhouse gas concentrations, may have significant impacts on regional weather patterns. Projections from the CMIP6 (Coupled Model Intercomparison Project: Phase 6) GCM suggests poleward shift of the eddy-driven jets and storm tracks in both hemispheres [10, 11] while other studies predict a weakening of jet streams due to a reduction in the temperature gradient between the poles and equator [12, 13], leading to increased variability. It was also shown that within the CMIP6 projections, there are still biases in the representation of jets [14, 15], highlighting the requirement for improved modelling within GCMs, given their use in projections and informing mitigation strategies.

In simplifying the system, the effects of turbulence are parameterised by a stochastic forcing. More widely, stochastic parameterisations have been developed to represent the probability distribution of possible sub-grid scale tendencies conditioned on the large scale [16]. Originally proposed to provide better estimates of uncertainty than that of multi-simulator ensembles, stochastic parameterisations have also been shown to be more consistent with the underlying equations of motion and even reduce the systematic error [17]. While some research has been conducted into producing machine-learnt stochastic parameterisations, as will be discussed in Section 1.3 [18, 19, 20], these all attempt to characterise uncertainty in the observation fields using probabilistic methods, our work looks at finding a latent representation of a fundamentally stochastic system in the form of a ROM, on which the underlying dynamics evolve, which characterises the interaction between the forcing and the mean-flow. This enables us to quantify to what extent different system states are driven by deterministic or stochastic dynamics.

## 1.2 Beta Plane System

Large-scale geophysical flows are primarily influenced by rotation and stratification. In the horizontal direction, the pressure gradient and the Coriolis force are in balance, known as geostrophic balance, while in the vertical the pressure gradient and gravity are in hydrostatic balance. The circulation around the planet is primarily driven by heat radiation as a result of the temperature gradient between the equatorial regions that receive a greater proportion of radiation from the sun, compared to higher latitudes. As previously mentioned, we are looking to use idealised models, obtained by using a series of model reductions to the underlying equations that allow one to focus on essential physics with regard to certain phenomena- retaining the dominant forces acting on the fluids. One such reduction is the choice to disregard unimportant modes such as gravity waves and sound waves when studying large-scale jet dynamics.

When viewed from a distance, planetary atmospheres and oceans are shallow fluid layers on a rotating sphere, a consequence of their large horizontal extent compared with their depth. By neglecting stratification to produce a single-layer model and we incorporate the effects of planetary

rotation by adopting a beta plane approximation, which is a simple device used to represent the latitudinal variation in the vertical component of the planetary rotation [21, 22, 23]. Here, the Coriolis parameter is given by $f = (2\sigma/a)sin\theta$ at latitude $\theta$ for a planet with radius $a$ rotating with angular velocity $\Omega$. Under these assumptions, we obtain the shallow water equations, considering the simplest barotropic formulation - a single-fluid layer of constant density:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{f} \times \mathbf{u} = -g\nabla h, \qquad \frac{\partial h}{\partial t} + \nabla \cdot (h\mathbf{u}) = 0 \tag{1.1}$$

Within this unstratified model, coordinates $x$ and $y$ correspond to the zonal and latitudinal directions respectively, $\mathbf{u}(x, y, t) = (u, v)$ is the horizontal velocity, $h(x, y, t)$ denotes the total fluid depth, $\mathbf{f} = \mathbf{f}\hat{z}$ is the local vertical component of the planetary rotation vector and $\nabla = (\partial_x, \partial_y)$ is the two-dimensional gradient operator. This is known as the beta plane approximation, first introduced by [24].

The ratio of the magnitude of the advective term, $\mathbf{u} \cdot \nabla \mathbf{u}$, to the Coriolis term, $\mathbf{f} \times \mathbf{u}$, is a key parameter called the Rossby number. For a characteristic horizontal velocity $U_c$, a characteristic horizontal length scale $L_c$, and a characteristic Coriolis frequency $f_c$, this is given by $R_o = U_c/f_c L_c$ where $R_o \ll 1$ implies the importance of rotation. Typical values for large-scale mid-latitude flows in the Earth's atmosphere are 0.1 and 0.01 for the ocean, confirming that rotational effects are significant.

We can simplify the system by using the smallness of this parameter in the limit in which there is an approximate balance between the pressure gradient and the Coriolis force, known as quasi-geostrophic (QG) motion, this was first formulated by [25]. This leads to the shallow water quasi-geostrophic potential vorticity equation, where the dynamics are determined by a single dynamical variable, the stream function, $\psi$ or alternatively the vorticity, $\zeta = \nabla^2\psi$:

$$\left(\frac{\partial}{\partial t} + u_g \cdot \nabla\right)\left(\nabla^2\psi + y - \frac{1}{L_d^2}\psi\right) = 0 \tag{1.2}$$

where $\mathbf{u}_g$ denotes the geostrophic velocity field and $L_d = \sqrt{gH}/f$ is the Rossby radius of deformation - the length scale at which the geostrophic balance becomes important. We can take the limit in which the characteristic scales of motion are much smaller than $L_d$, resulting in the dynamics becoming two-dimensional.

In the three-dimensional, stratified system, turbulence naturally arises as a result of dynamical instabilities such as baroclinic instability, however, in the absence of stratification in the two-dimensional case, there is a requirement for small-scale eddies, that generate turbulence, to be parameterised by a stochastic forcing, $\xi$. The forcing is defined to be rapidly decorrelating, spatially homogeneous and random in space and time, in which energy is injected homogeneously and isotropically at a constant rate $\epsilon$ into a narrow band of wavenumbers in Fourier space isotropically with an associated wavenumber $k_f$. In doing so, it is assumed that the barotropic and baroclinic interactions are decoupled and that the baroclinic eddies can be represented by an external force added to the barotropic model.

A state of equilibration can then be achieved by the inclusion of dissipation terms, representing physical processes such as radiative cooling [26] or Ekman friction [27]. A linear damping rate $\mu$ is used to dissipate energy at the largest spatial scales, while the build-up of enstrophy at the smallest scales is removed using hyperviscosity of order n and rate $\nu_n$, $n \in \mathbb{N}$. For simplicity, we

approximate the Coriolis parameter, using a first-order Taylor series, $f = f_0 + \beta y$, with a constant latitudinal gradient $\beta = (2\Omega/a)cos\theta_0$ close to latitude $\theta_0$- the $\beta$ plane [21, 22, 23]. Incorporating these choices the beta plane vorticity equation becomes:

$$\frac{\partial \zeta}{\partial t} + J(\psi, \zeta) + \beta \frac{\partial \psi}{\partial x} = \xi - \mu\zeta + \nu_n \nabla^{2n} \zeta \tag{1.3}$$

for relative vorticity $\zeta = \nabla^2 \psi$ and the corresponding velocity field $(u, v) = (-\partial_y \psi, \partial_x \psi)$, with J being the Jacobian determinant. This nonlinear model permits all interactions between eddies and as $\xi$ is specified to be spatially homogeneous, admits the latitudinal symmetry $(t, x, y, \psi, \zeta) \mapsto (t, x, -y, -\psi, -\zeta)$. A Doubly-periodic geometry is imposed on a square domain of size $L_D$ such that $(x, y) \in [0, 2\pi L_D)^2$, and information regarding how this model is implemented numerically, using a pseudo-spectral solver, can be found in appendix A.3.

With the study focusing on zonally-orientated flows, an equation governing the evolution of the zonally averaged zonal velocity field is obtained by applying an eddy-eddy mean decomposition to the variables $u(x, y, t) = \overline{u}(y, t) + u'(x, y, t)$ with the mean flow defined as the zonal average, $u$:

$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial y}(\overline{u'v'}) = -\mu U + \nu_n \frac{\partial^{2n}}{\partial y^{2n}} U \tag{1.4}$$

Where for convenience with notation, we represent the zonal mean zonal velocity profile as $U(y, t) = \overline{u}(y, t)$. In addition to the zonal average, the overbar denotes a time average over a short interval. Assuming that $\xi(y, t) = 0$ as $\xi$ is spatially homogeneous, $\xi(y, t)$ does not appear on the right-hand side of (1.4), and forces $u'$.

Due to the tendency of energy in two-dimensional space to continuously transfer into larger scales, the eddy-driven zonal jets are forced and maintained by the eddy momentum flux convergence, or Reynolds stress divergence, $\partial_y(\overline{u'v'}) = -\overline{v'\zeta'}$, whereby the eddy momentum flux convergence equals the meridional flux of eddy potential vorticity, implying that jet formation can be described in terms of the latitudinal transfer of potential vorticity [28] and that that eastward jets form where eddy momentum fluxes converge, and westward jets when they diverge [29]. It was then shown that regions of strong alternating currents suppress eddy diffusivities and act as effective barriers to north-south, or meridional, transport [30, 31].

The jets are not directly driven but arise spontaneously from the forcing, while the equilibrated jet streams are inherently unstable, with questions arising concerning their time variability. They exhibit a multitude of types of variability, including latitudinal shifts, strength changes and jet mergers [32, 33]. Figure 1 shows a latitude-time plot of $U(y, t)$, where between $t = 250$ and $t = 300$ the quasi-stable two-jet state can be seen to drift northwards, and this example of a random wandering of the jets is very likely driven by the stochastic forcing analogous to a random walk. An example of a jet nucleation can be seen at $t \approx 370$ at the bottom of the figure, before the merging event seen at $t \approx 500$. It cannot be determined how influential the forcing is on these events, as the change in state could be a consequence of excitation from one basin of the attractor to another caused by the forcing, however, this behaviour also shows similarity to deterministic chaos, with the trajectories in phase space transitioning between unstable solutions [34].

Anticipating that the stochastic forcing plays a large role in the system variability, by looking at various states of the system we can observe the potential impact the forcing has on the system. This motivates producing a reduced-order model that can produce a large ensemble of future time

evolutions from a particular system state, much faster than with numerical integration, alongside a latent space, in which state-space trajectories (parameterised in terms of the latent dimensions) can be explored to yield insight into the underlying system dynamics to quantify to what extent different system states are driven by deterministic or stochastic dynamics.

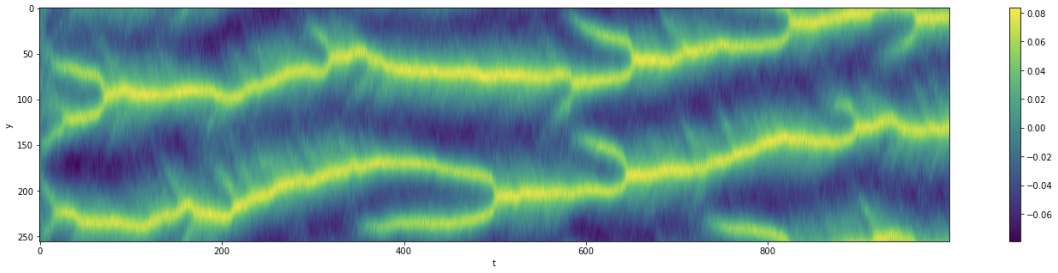


**Figure 1:** *An example latitude-time plot of the beta plane system displaying $\bar{u}(y,t)$ over 1,000 observable time-steps (equivalent to 12.5 damping time units) from a numerical integration.*

## 1.3 Data-Driven Modelling

Data-driven methods fit a model via training on data as opposed to numerical modelling based on physical equations. In this research we implement Deep Learning [35], a subset of Machine Learning, that uses layers of artificial neurons and representation learning to obtain the correct model fit by hierarchically extracting informative features from data. Neural networks can be thought of as parametric, non-linear function approximates that are fit from training data, to learn functions from the input space to the output space, more detail is given in Section 1.5. Work has been conducted on applying Deep Learning to Geophysical Fluid Dynamics in various contexts, and here I will give a brief overview of a few of the most relevant studies.

In recent years, a large volume of research has been conducted in using Deep Learning to replace model-based parameterisations- utilising the computational speed-up that Deep Learning offers [36]. These have looked at emulating various different atmospheric processes with Deep Learning such as radiation schemes [37, 38], convection [39] and cloud resolving models [40]. In our case however, these models are not appropriate as they are deterministic, not able to capture the stochasticity that drives dynamics in beta plane system.

The SINDy algorithm (Sparse identification of nonlinear dynamics) [41] was developed to determine equations of motions from data, using a library of basis functions. This has proved very successful for both full-flows as well as to determine time-evolution of a reduced order basis- a Galerkin Method. However, like previous methods, SINDy assumes a deterministic system, with noise only added to account for uncertainties in precision. One research direction would be to update SINDy to also model stochastic differential equations, however we decided that the use of Deep Learning provided greater flexibility, and potentially greater model performance, albeit at the cost of extra exploration required to fine-tune the model.

While Physics Informed Neural Networks (PINNs) is an umbrella term to describe a set of methods that use constraints when training Deep Neural Networks, such as physical laws, as regularisation agent that limits the space of solutions [42, 43]. PINNs can also be used to describe methods that use the automatic differentiation used in the back-propagation algorithm to train Neural Networks to calculate partial derivatives and form the equations, which are enforced through a loss function

[44, 45].

Generative Adversarial Networks (GANs) [46], a Deep Learning model architecture, have been used to produce stochastic parameterisations [18]. They use an adversarial network that looks to determine whether an output from the model is generated or from the training dataset. The network is given the goal of generating samples that cannot be correctly classified by the adversarial network and thus produces samples that are statistically similar to the training dataset. They have been very effective for super-resolution tasks and resultingly for statistical downscaling [47, 48] as well as now-casting realistic precipitation maps [20]. We will use an aspect of GANs to aid the training of our reduced order model, outlined in Section 1.7.

Most recently diffusion based generative models were used to predict local-scale precipitation maps [49]. Diffusion models have, over the past year, become the state-of-the-art in image generation tasks, however as the number of matrix multiplications required to make a prediction is of an order of $10^3$ times greater than the above methods, for very marginal gains in performance, we will not be exploring their potential here.

## 1.4 Reduced Order Modelling

Here we use ROMs to refer to models with fewer degrees-of-freedom than the original system. We will focus our study on one approach to obtaining ROMs - Manifold Learning [50]. Here we assume that patterns underlying in our data, in the case of systems of fluids these are dynamics that evolve our fields, lie on a lower-dimensional manifold, $\mathcal{M}$, with dimension $D_{\mathcal{M}}$, embedded in the higher-dimensional observation space, with dimension $D$, where $D_{\mathcal{M}} \ll D$. Manifold Learning differs from the standard class of dimensionality-reduction techniques used in fluid dynamics currently, with these being Singular Value Decomposition (SVD), which can be extended to time-varying physical systems and are so-called Proper Orthogonal Decomposition (POD) [51]. POD and SVD only capture linear manifolds where as Manifold Learning generally describes a class of algorithms that use nonlinear dimensionality reduction to obtain the form of the internal manifold.

POD decomposes a given field $\mathbf{u}(\mathbf{x}, t)$, with corresponding dynamical system $\partial_t \mathbf{u} = f(\mathbf{u}, t)$, into a set of spatial functions $\Phi_k(\mathbf{x})$, or modes, with time coefficients $a_k(t)$:

$$\mathbf{u}(\mathbf{x}, t) = \sum_{k=1}^{N} a_k(t)\Phi_k(\mathbf{x}), \tag{1.5}$$

where $N$ is the wavenumber of the highest chosen mode. Other methods to decompose the spatial features of $\mathbf{u}$ include, for example a Fourier decomposition, however POD is optimal in as much as it maximises the Kinetic energy captured by the first N modes. The orthonormal property of $\Phi$ implies that each time coefficient $a_k(t)$ depends only on the corresponding $k^{th}$ spatial mode $\Phi_k(\mathbf{x})$. To obtain $\Phi$, we produce snapshots of $\mathbf{u}(\mathbf{x}, t)$, defined as:

$$U = \begin{pmatrix} u(x_1, t_1) & ... & u(x_n, t_1) \\ ... & & ... \\ u(x_1, t_\tau) & ... & u(x_n, t_\tau) \end{pmatrix} \tag{1.6}$$

for n spatial dimensions in the observation space of $\mathbf{u}$ and $\tau$ time samples, in order to compute

the covariance matrix, $C = (\tau - 1)^{-1} U^T U$. We can diagonalise $C = \Phi \Lambda \Phi^{-1} = \Phi \Lambda \Phi^T$ using that the $\Phi$ is orthonormal ($\Phi^{-1} = \Phi^T$), where $\Lambda = \lambda \mathbb{I}$, to obtain $\Phi$. By convention, eigenvectors $\lambda_k$ are ordered from smallest to largest, to give corresponding $\Phi_k$ that correspond to each spatial mode $k$. To obtain the time coefficients, $a_k(t)$, $\mathbf{u}(\mathbf{x}, t)$ is projected onto $\Phi_k(\mathbf{x})$, or equivalently $A = U\Phi$, where $A$ is our time coefficient matrix.

POD has previously been coupled with time evolution models [52], such as Recurrent Neural Networks (RNN) [53], while linear Deep Learning models can be thought of as equivalent to SVD or POD transformations and have been used to study fluids [**?**]. However, given that we are working with a non-linear system, we choose to follow suit of [54] that uses a non-linear Deep Learning approach to obtain the dimensionality reduction transformation, which has been shown to capture complex patterns more effectively, with a smaller number of modes, corresponding to the size of $D_{\mathcal{M}}$.

Also used in fluid dynamics is the Koopman spectral theory, where the Koopman Operator, which is a linear operator advances measurement functions of the state with the flow of the dynamics in an infinite-dimensional space, which we approximate by the dynamic mode decomposition (DMD) algorithm. While this does not provide the form of a ROM, it is mentioned due to its parallels with the rapidly evolving area within Deep Learning known as Reservoir Computing [55], a form of Echo state networks[56], which may provide a secondary approach to modelling during this PhD. These networks only require the training of a single readout layer that maps from a reservoir network to the desired output, significantly reducing training time. It has also been proposed that these networks are able to make statistical predictions of extreme events from small datasets even if the events lie outside the distribution of the training data [57], however, I still remain sceptical about these claims and conducting research to see if these claims could be supported would be interesting.

It is prudent to remember that the beta plane system is a stochastically forced PDE, and therefore the dynamics will not lie on a stationary manifold in a lower-dimensional space but will correspond to a statistical manifold [58], where each location in the parameter space associates to a probability density function (PDF). The dimension of the manifold, $D_{\mathcal{M}}$ will pertain to the number of parameters, while we will represent the associated PDFs with a mean, $\mu$, and standard deviation $\sigma$, enforcing that the learned distributions take the form of a normal distribution. This is expanded upon in Section 1.6.

Others have looked at finding a manifold for stochastic dynamical systems, by defining an inertial manifold with drift for [59], non-compact manifolds [60] and using a stochastic Galerkin reduced basis [61, 62]. However, we propose that associated manifolds can be found using a variant of the Autoencoder, learned through Variational Bayes [63], the Variational Autoencoder (VAE) one that appropriately samples from a probability distribution for each latent variable. Via sampling, this approach provides an ensemble, where members of the ensemble share latent parameters, with equivalent mean, $\mu$, and standard deviation $\sigma$, which we will use as analogous to numerical integration with identical dynamical parameters, but different realisations of the random forcing, $\xi$. In the case of the beta plane system, the learned transformation from the latent space back to the observation space will learn the associated non-linear interaction between the stochastic forcing $\xi$, the mean flow $U(y, t)$, that we will attempt to probe to learn nature of this interaction and to quantify to what extent different instantaneous system states are driven by deterministic or stochastic dynamics.

As touched upon earlier, another rationale for using ROMs is model explainability. Deep Learning models are often seen as black-box models that are difficult to interpret, however, a ROM would provide the latent manifold where the reduced modes can be explored, as well the identification of regimes [51] and exploring the parameterised state space [64]. Naturally, there are still difficulties in interpreting latent spaces learned through non-linear transformations, however, with the greater volumes of data now available, clustering methods to identify non-linear patterns have developed significantly. We will also leverage the ability to produce large ensembles at very low computational cost to evaluate how the Deep Learning model has learned the dynamics, identify biases and produce models that are robust to these.

## 1.5 Autoencoder Neural Networks with Variational Bayes

The simplest form of Neural Network is the Feed-Forward Neural Network, comprised of fully-connected layers- that is each neuron is connected to every neuron in the following layer, with each connection scaled by a coefficient, or weight, $w$, and shifted by a bias, $b$. At each neuron a non-linear activation function, $\sigma$, is applied- with common choices being the $tanh(x)$ function and the rectified linear unit, $relu(x)$. This allows for the stacking of non-linear functions where the output of a neuron, $y_k$ is described by the following summation:

$$y_k = \sigma(\sum_i w_{ki} x_i + b_i) \tag{1.7}$$

for the $k^{th}$ neuron in a layer where $i = 1, 2, ..., n$ where $n$ is the number of neurons in the previous layer, and $x_i$ their corresponding outputs.

An Autoencoder, or more specifically an Undercomplete Autoencoder, is a Neural network, or Multi-Layer Perceptron, that contains an information bottleneck, corresponding to a layer smaller in size than the input and outputs, forcing the model to only learn the relevant features to make an accurate reconstruction. We will use the Encoder-Decoder structure of an Autoencoder to represent our coordinate transformations. The Encoder, with weights, $\phi$, will parameterise $\mathbf{z} = \mathcal{E}_\phi(\mathbf{U})$, and the Decoder, with weights $\theta$, will parameterise $\mathbf{U} = \mathcal{D}_\theta(\mathbf{z})$, where $\mathcal{E}$ and $\mathcal{D}$ are either single layers or stacks of layers as defined in (1.7). The goal of the training procedure is to find the optimal parameters, $(\phi, \theta)$ minimise the cost, or loss, function, $\mathcal{L}$, with inputs $\mathbf{U}$ and output from the decoder $\tilde{\mathbf{U}}$:

$$arg\min_{\phi,\theta} \mathcal{L}(\mathbf{U}, \tilde{\mathbf{U}}(\phi, \theta)), \tag{1.8}$$

where $\mathcal{L}$ can take various forms.

In order to find the optimal weights, we train the neural network via backpropagation [**?**], with the gradient of the loss functions, with respect to the model weights, computed for each later, integrating backwards from the final layer to the input layer, updating the weights accordingly:

$$\phi_{t+1} = \phi_t - \alpha \frac{\partial \mathcal{L}(\mathbf{U}, \tilde{\mathbf{U}}(\phi_t, \theta_t))}{\partial \phi} \tag{1.9}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\mathbf{U}, \tilde{\mathbf{U}}(\phi_t, \theta_t))}{\partial \theta} \tag{1.10}$$

where $\alpha$ is the learning rate, reduced throughout training and $\mathcal{L}(X, \theta_t)$ is the loss function between

the output, $\tilde{Y}$, and the target, $Y$, at iteration $t$, with training attempting to find the $\theta^*$ that minimises the loss function.

As mentioned previously, the basic form of an Autoencoder is a deterministic mapping, as the weights and biases are fixed when performing inference. As mentioned, this makes this form of the model unsuitable for the task of learning a stochastically forced system. We, therefore, use a modification of the Autoencoder, the Variational Autoencoder (VAE) [65], to find a stochastic manifold, in which we can sample. Here the latent representation of the system, contained in $h$ is mapped to a distribution, defined by a mean, $\mu$, and standard deviation, $\sigma$, for each latent variable, allowing one to sample from the distribution, where the sample is taken from $z_i = \mu_i + \sigma_i \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$, where $i$ is the latent dimension, $i \in [1, D_{\mathcal{M}}]$.

As will be shown, the form of the marginal likelihood (prior distribution) of the observed data, parameterised by the latent space, is unknown, and therefore we use a Variational Bayesian learning method to derive a lower bound for the marginal likelihood- the evidence lower bound (ELBO). This is a combination of the negative log-likelihood function and the Kullback-Leibler divergence ($D_{KL}$), which is a measure of loss when $q_\phi(z)$ is used to approximate $p_\theta(z)$, with a $\beta$ term introduced to provide a weighting between these two loss terms (not to be confused with the $\beta$ term that gives the system its name, in (1.3)):

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{reconstruction} + \beta \mathcal{L}_{prior} \tag{1.11}$$

$$-ELBO(\theta, \phi, \varphi) = \mathbb{E}_{z \sim q_\phi(z|U)}[log p_\theta(u|z)] + \beta D_{KL}(q_\phi(z|U), p_\theta(z|U)) \tag{1.12}$$

Using an assumption of Gaussian distribution, one can show that the Mean Square Error (MSE) is equivalent to the negative log-likelihood function (see appendix A.1 for details), and thus we define our loss function as follows:

$$\mathcal{L}(\theta, \phi) = MSE + \beta D_{KL} \tag{1.13}$$

$$= \frac{1}{B} \sum_{i=1}^{B} \left( \|\tilde{U}_i - U_i\|^2 + \beta \mathbb{E}_{z \sim q_\phi(z|U)} \left[ q_\phi(z_i|U_i) log \frac{q_\phi(z_i|U_i)}{p_\theta(z_i|U_i)} \right] \right) \tag{1.14}$$

where $\tilde{U} = \mathcal{D}_\theta(U_{t+1}|RNN_\varphi(\mathcal{E}_\phi(z_{t:t-l}|U_{t:t-l})))$ is the output of the model, D is the dimensionality of $z$ - the size of the latent layer and B is the training batch size.

In practice, information about $p_\theta(z|U)$ is not known, so the true $D_{KL}$ cannot be evaluated. Instead, we enforce that $p_\theta(z)$ takes the form of a standard normal distribution, $\mathcal{N}(0, 1)$ and the $D_{KL}$ term in the ELBO loss acts as a regulariser which moves $q_\phi(z|U)$ towards the form $\mathcal{N}(0, 1)$. We are able to make this assumption on $p_\theta(z)$, as the mapping applied by the Encoder is not yet defined and will be learned via training. Using this assumption the objective function is evaluated as the following (see appendix A.2 for details):

$$\mathcal{L}(\theta, \phi) = \frac{1}{B} \sum_{i=1}^{B} \left( \|\tilde{U}_i - U_i\|^2 + \frac{\beta}{D} \sum_{j=1}^{D_{\mathcal{M}}} \left( \frac{1}{2} (log \sigma_{i,j}^2 - \mu_{i,j}^2 - \sigma_{i,j}^2 + 1) \right) \right) \tag{1.15}$$

The parameter $\beta$ weights the reconstructive term in the loss function with the $D_{KL}$, and we can

vary $\beta$ to attempt to capture more or less variation within the training set, with $\beta$ varying how the model learns the standard deviation, $\sigma$, when in each of the latent dimensions.

Typically Autoencoders are used to learn the latent representation of the dataset before the decoder is removed and the latent variables, $\mathbf{z}$, are used for some downstream task. While VAEs are generative models, the transformation from the latent space to the output space is learned, and new outputs can be generated by sampling $\mathbf{z}$, negating the requirement for the encoder after training. In our context, both the encoder and decoder are retained- with the encoder mapping initial conditions to the latent space and the decoder transforming predicted evolutions back to our observation space after some time stepping procedure has taken place in the latent space. We, therefore, require an appropriate parameterisation of $\partial_t \mathbf{z} = F(\mathbf{z}, t)$, and we choose to use a Recurrent Neural Network (RNN) [66] for this.

## 1.6 Probabilistic Neural Networks

Recurrent Neural Networks (RNN) are forms of Neural Networks, that are tailored towards the tasks whereby the input data is sequential. With the avail of big data, much of which takes the form of text, RNNs were initially developed for the task of natural language processing, which may take the form of speech recognition or language translation. Within the Feed-Forward Neural Networks mentioned above, data is propagated forward from the input to the output, with each layer a function only of the preceding layers. RNNs instead use information from their previous output as an input when evaluating the following item in a sequence, in a process called unrolling. At each time step, for an input sequence $x = (x_1, x_2, ..., x_T)$, the RNN computes the following:

$$h_t = tanh(x_t W^T + h_{t-1} U^T) \text{ with} \tag{1.16}$$

$$o_t = tanh(h_t V^T) \tag{1.17}$$

where $h_t$ is the hidden state at time $t$, $x_t$ is the input from the sequence at time $t$ and $h_{t-1}$ is the hidden state of the at time $t-1$ and $o_t$ is the output of the RNN at time $t$, with $(U, V, W)$ the weight matrices, with we denote with $\varphi$. The hidden state refers to the hidden layer of the network, as was $y_k$ in (1.7) in the case of the Feed-Forward Neural Network. By using $h_{t-1}$ the RNN is able to use information from its representation of the previous time step in the sequence to make its prediction on the next, making these networks very well suited to time series data. Like with Feed-forward networks, the hidden states can be stacked to give a deep neural network. The weights $\varphi$ are optimised in the same way, as was outlined in (1.9 & 1.10) using backpropagation.

In Section 2.2 we combine the features of VAEs and RNNs to produce a network that evolves the system within the latent space in a probabilistic manner, sampling stochastically to produce individual realisations. As mentioned earlier, this is key when looking to learn the dynamics of a stochastically forced system like the beta plane system, where the time evolution of the jets depends both on the nonlinear two-way interaction between the mean flow and the eddies. Crucially, the time history of the stochastic forcing, the current state or the recent history of the system does not predict the forward evolution but instead determines a distribution of possible time evolutions. A probabilistic network allows the modelling of this distribution and the production of a family of trajectories in order to capture the variation of the system.

## 1.7 Adversarial Training

In the VAE, the $D_{KL}$ regulariser ensures that the distribution of the latent variables is as expected, so we look to introduce an additional regulariser term for the model outputs to ensure they are from a distribution, close to (ideally identical to) the training data. Here we will utilise an approach from Generative Adversarial Networks (GAN) that has proved successful in state-of-the-art generative models.

These models utilise an implicit loss, in the form of a classifier network (a secondary neural network) that is trained to distinguish between data from the training set and outputs from the neural network models. This then informs the generating network (which in our case is the VAE) whether its samples are similar to that of the training dataset. As this secondary network is also differentiable, through backpropagation the weights of the generator network are updated to produce results closer to that of the targets from the training dataset, while the classifier improves in its ability to distinguish between the two classes, leading to both networks improving until an equilibrium is reached where the generator is able to produce samples that the classifier cannot distinguish correctly - with these samples being drawn from the same distribution as the training dataset, reducing or removing any biases or errors that are associated generative model to ensure that they correctly model the underlying system or distribution they are attempting to mimic.

The classifier network referred to as the Discriminator with weights $\chi$, is trained using the following objective function

$$\mathcal{L}_D(\chi, \theta, \phi, \varphi) = \mathbb{E}_{U\sim p(U)}[logD_\chi(U)] + \mathbb{E}_{\tilde{U}\sim p_{\theta,\phi,\varphi}(\tilde{U})}[log(1 - D_\chi(\tilde{U})] \tag{1.18}$$

where $\tilde{U}$ is the output from the generative model (in our case that is a VAE), outlined above. The goal of the discriminator is for the first term in (1.18), which is the average prediction of the discriminator on real sampled data from the training dataset, to be as large as possible, corresponding to correctly identifying real data with a high probability. The real data, $U$ from the dataset is assigned a label 1, while the fake, generated data $\tilde{U}$ is assigned a label 0. The Discriminator outputs a value that lies between $[0, 1]$, attempting to classify the data correctly.

The second goal of the discriminator is to correctly identify generated data from the VAE as fake with high confidence, corresponding to $logD_\chi(\tilde{U})$ being as small as possible. To turn the expression into a maximisation problem, the term $(log(1-D_\chi(\tilde{U}))$ is used in (1.18). The goal of the generative model is for $log(1-D_\chi(\tilde{U}))$ to be as large as possible- corresponding to the Discriminator thinking that the generated data is real with high confidence, setting up the following optimisation problem:

$$arg\min_{VAE}\max_{D}\mathcal{L}_{Disc} = \mathbb{E}[logD_\chi(U)] + \mathbb{E}[log(1 - D_\chi(\tilde{U})] \tag{1.19}$$

In training with this additional adversarial approach, naturally, the learning complexity increases and a common problem with these networks is struggling convergence during training. This is often a symptom of a generator or discriminator being 'too powerful', stopping the other network from improving. We here use adversarial training in addition to a generative network, the VAE. This model generates predictions, conditioned on inputs of previous time steps, as opposed to producing new samples from noise - this makes the modelling challenge simpler than for stand-alone GANs, improving chances of convergence, as will be discussed in Section 2.3.

Much of the modelling in Deep Learning comes down to the choices made in the model architecture,

that is the configuration and choice of layers and non-linear activation, as well as the prescription of the loss function and training objectives. There are a number of aspects of Deep Learning required to implement these methods that are omitted for the sake of brevity, however, knowledge of these is not required to follow the concepts outlined here. These include, but are not limited to, learning rate optimisations, learnable biases, weight initialisation, batch normalisation and weight regularisation. A great reference on these can be found here [67].

# 2 Deep Learning to Model the Beta Plane System

By using a Reynolds Decomposition on the beta plane system (1.3) we obtain the following coupled system:

$$\left(\frac{\partial}{\partial t} - \frac{1}{Re}\nabla^2\right) U = -(u'v')_y \tag{2.1}$$

$$\mathcal{L}(U, k)u' = \xi(t) \tag{2.2}$$

Where $\mathcal{L}$ is a non-linear operator, or Linear operator for the quasi-linear case, see Section 3, where $k$ is the wavenumber. This presents two potential problems that can be tackled with machine learning:
1. Learn the dynamics of $U$ (2.1), implicitly learning (2.2) for the non-linear case - which will be discussed in this section.
2. Alternatively one could learn a form of the inverse of $\mathcal{L}$ for the quasi-linear case and to obtain a model for the evolution of $u'$ which can be used as a parameterisation, that, when coupled with (2.1), gives a ROM of the system - this which will be discussed in Appendix A.4.

## 2.1 Autoencoder - Spatial Representation

As mentioned in Section 1.4, we assume that coupling an Autoencoder, a Deep Learning model with an information bottleneck should produce a far better ROM than linear models, at equivalent order. Here we look to produce an improvement over Singular Value Decomposition (SVD), also known as Principal Component Analysis (PCA) in the statistical learning community, by utilising the non-linear nature of the Autoencoder. This dimensionality reduction is a purely spatial representation, ignoring time-correlations, for the time being.

The inputs to the network were randomly sampled mini-batches from the training dataset, time evolution histories of $U(y, t)$ the zonally averaged zonal velocity outlined in equation 1.3, obtained from numerical integration. The system at each time step is a one-dimensional vector of size $D = 256$, with the target outputs of the network the same as the inputs and the training process attempting to determine these transformation mappings such that the difference between the left and right hand sides of:

$$\tilde{\mathbf{U}} = \mathcal{D}_\theta(\mathcal{E}_\phi(\mathbf{U})) \tag{2.3}$$

The size of the latent space $D_\mathcal{M}$, i.e. the embedding or bottleneck layer in the Autoencoder corresponding to the output size of the Encoder was varied, while equivalently the number of modes SVD/PCA was varied to give a comparison. Figure 2 shows the equivalent RMSE losses evaluated on a validation set of $U(y, t)$ profiles not seen during training, that had been encoded and decoded by the Autoencoder and that had been reduced and transformed back via SVD/PCA for a variety of latent dimensions $D_\mathcal{M}$ where $D_M < D = 256$. It can be seen that the Autoencoder was able to find a reduced representation of the system much more effectively than the linear methods, with an error reduction of approximately two orders of magnitude for each size of embedding layer.

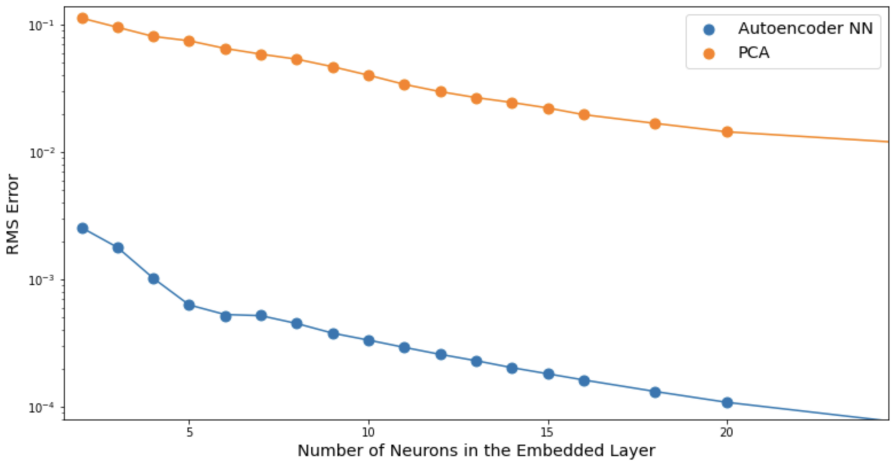


**Figure 2:** *Hi*

Using Autoencoders in this way is not novel, however it does confirm that the beta plane system can be appropriately approximated using reduced representations. When a standard Autoencoder was coupled to an RNN, the model could not effectively represent the spatial-temporal dynamics, and failed to give reasonable predictions of time evolutions. We hypothesised that the reason for this was that model the was not set up to account for this stochasticity, as an Autoencoder searches for a stationary manifold. We then looked to implement the Variational Autoencoder outlined in Section 1.5.

## 2.2 Variational Autoencoder with RNN

As outlined in Section 1.5, in order to find a ROM of the beta plane system, we assume that the system's dynamics lie on an internal statistical manifold $\mathcal{M}$ of dimension $D_{\mathcal{M}}$ embedded in a higher-dimensional space. Here we look for a neural network parametererisation of $z = \mathcal{E}_\phi(\mathbf{U})$ in the form of our Encoder, with weights, $\phi$, that maps $\mathbf{U}(\mathbf{y}, t)$ onto coordinates $\mathbf{z}$ on $\mathcal{M}$ and correspondingly a neural network parametererisation of $\mathbf{U} = \mathcal{D}_\theta(\mathbf{z})$ in the form of the Decoder network, with weights $\theta$. We therefore also require a neural network parametererisation for the latent dynamics $\partial_t \mathbf{z} = F(\mathbf{z}, t)$ on $\mathcal{M}$.

The inputs to the network were randomly sampled mini-batches from the training dataset, short time evolution histories of lenght $l$, of $U(y, t : t - l)$ the zonally averaged zonal velocity outlined in equation 1.3, obtained from numerical integration. The system at each time step is a one-dimensional vector of size $D = 256$, with the target outputs the following time step $U(y, t + 1)$.

Where this work differs from other work that uses a coupled VAE-RNN architecture [68], is the placement of the stochastic layers within model. It is common place for the Encoder to output $\mu$ and $\sigma$ vectors, before a sampling operation, $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$, gives a single vector that can be input to the Decoder [69](In reality, we choose to output the log of the variance $log(\sigma^2)$ rather than the standard deviation before calculating the standard deviation, $\sigma$; but for notation we will continue to denote the outputs from the neural network as $\mu\&\sigma$). This however, would be incompatible, with our goal of evolving the stochastic dynamics of the system on the manifold $\mathcal{M}$, as any future time steps would generated in a purely deterministic way.

Another approach that is used is to combine the RNN with the encoding part of the network, so that the network contains a VAE at every time step of the RNN [68]. This does turn the RNN into a probabilistic model, however this involves encoding and then decoding for each time step that one wishes to produce, which again is not satisfactory, as we wish to be able to encode our initial conditions before evolving the system in the latent space in a probabilistic manner, before decoding the predictions to the observation space.

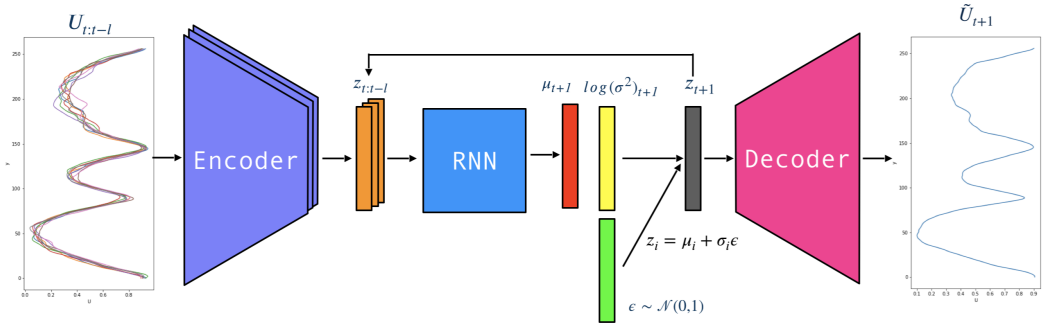


**Figure 3:** *A schematic showing the architecture of a combined RNN-VAE, outlining (1.4-1.6). The arrows, show the forward pass of the data, with the loop displaying when the prediction $z_{t+1}$ is used by the RNN to make a forecast on the next time step. The green block details the random noise from a standard normal distribution that samples our learned distribution in the latent space.*

We use an architecture whereby, the network is given a short time series of length $l$, $[U_t : U_{t-l}]$ that is passed to a deterministic encoder to return a latent representation of that sequence:

$$[z_t, ..., z_{t-l}] := \mathcal{E}_\phi([U_t, ..., U_{t-l}]) \tag{2.4}$$

These are then input to an RNN, where the output of the RNN are a mean and standard deviation, for each latent variable, for the following time step:

$$\mu_{t+1}, \sigma_{t+1} := RNN([z_t, ..., z_{t-l}]) \tag{2.5}$$

Using the sampling operation $\tilde{z}_{t+1} = \mu_{t+1} + \sigma_{t+1}\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$, we obtain a probabilistic forecast for the next time step, in latent space, $\tilde{z}_{t+1}$ (the tilde denoting that is is a prediction). To obtain an ensemble, a different sample can be taken. $\tilde{z}_{t+1}$ can either be appended to the time series of latent variables to give $[\tilde{z}_{t+1}, z_t, ..., z_{t-l+1}]$ to produce a projection for the next time step, or can be input to the deterministic decoder:

$$\tilde{U}_{t+1} := \mathcal{D}_\theta(\tilde{z}_{t+1}) \tag{2.6}$$

to output one possible prediction of the next time step of $U_{t+1}$, in observation space, given to the probability distribution of the parameters of the stochastic manifold. A diagram of the forward pass of the Neural Network architecture can be seen in Fig 3

To ensure that the various components of the network learn as intended, we alter the loss functions

**Algorithm 1** RNN-VAE Training Algorithm

$\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\varphi}\leftarrow$ initialise network parameters
**for** epoch in epochs
  **for** mini-batch in batches
    $\boldsymbol{U}_{(t:t-l)} \leftarrow$ random mini-batch from training dataset

    # Forward Pass though the networks
    $\boldsymbol{Z}_{(t:t-l)} \leftarrow \mathcal{E}_\phi(\boldsymbol{U}_{(t:t-l)})$
    $\boldsymbol{\mu}_{(t+1)}, \boldsymbol{\sigma}_{(t+1)} \leftarrow RNN(\boldsymbol{Z}_{(t:t-l)})$
    $\tilde{\boldsymbol{Z}}_{(t+1)} \leftarrow$ Sample from prior: $\boldsymbol{Z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \boldsymbol{\mathcal{N}}(0, 1)$
    $\tilde{\boldsymbol{U}}_{(t+1)} \leftarrow \mathcal{D}_\theta(\tilde{\boldsymbol{Z}}_{(t+1)})$

    $\boldsymbol{Z}_{(t+1)} \leftarrow \mathcal{E}_\phi(\boldsymbol{U}_{(t+1)})$
    $\boldsymbol{U}_{M(t)} \leftarrow \mathcal{D}_\theta(\mathcal{E}_\phi(\boldsymbol{U}_{(t)}))$

    # Calculate Losses
    $\mathcal{L}_{prior} \leftarrow D_{KL}(q(\boldsymbol{Z}|\boldsymbol{U}), p(\boldsymbol{Z}))$
    $\mathcal{L}_{reconstruction} \leftarrow \|\tilde{\boldsymbol{U}}_{(t+1)} - \boldsymbol{U}_{(t+1)}\|^2$
    $\mathcal{L}_{AE} \leftarrow \|\tilde{\boldsymbol{U}}_{(t)} - \boldsymbol{U}_{M(t)}\|^2$
    $\mathcal{L}_z \leftarrow \|\tilde{\boldsymbol{Z}}_{(t+1)} - \boldsymbol{Z}_{(t+1)}\|^2$

    # Update parameters according to gradients
    $\boldsymbol{\theta}_{Encoder} \xleftarrow{+} -\nabla_{\boldsymbol{\theta}_{Encoder}} (\mathcal{L}_{reconstruction} + \kappa\mathcal{L}_{AE})$
    $\boldsymbol{\phi}_{Decoder} \xleftarrow{+} -\nabla_{\boldsymbol{\phi}_{Decoder}} (\mathcal{L}_{reconstruction} + \kappa\mathcal{L}_{AE})$
    $\boldsymbol{\varphi}_{RNN} \xleftarrow{+} -\nabla_{\boldsymbol{\varphi}_{RNN}} (\mathcal{L}_{reconstruction} + \beta\mathcal{L}_{prior} + \lambda\mathcal{L}_z)$

stated in (1.15). First, we want the Encoder and Decoder to only map from our observation space to the latent space and back, and not to evolve their inputs forward in time- this is tasked to the RNN. To ensure such, we pass inputs through the Encoder and Decoder, without the RNN to produce an output, exactly as those obtained in Section 2.1, $\hat{U}_t := \mathcal{D}_\theta(\mathcal{E}_\phi(U_t))$ (we differentiate between a prediction from an autoencoded representation by using the tilde and hat respectively). The encoder, decoder and RNN are trained simultaneously, to ensure that learn temporal features when finding the latent representation, however, this additional loss constraint ensures that the encoder and decoder are only performing a spatial transformation.

As the variational component of the model is no longer part of the encoder-decoder structure, but part of the RNN, weights of the encoder and decoder, $\phi$ & $\theta$, do not depend on the values of $\mu$ & $\sigma$ in the latent space and accordingly the loss function for the Encoder and Decoder is computed as follows:

$$\mathcal{L}_{AE}(U_t, U_{t+1}, \tilde{U}_{t+1}, \hat{U}_t; \theta, \phi; \kappa) = \|\tilde{U}_{t+1} - U_{t+1}\|^2 + \kappa\|\hat{U}_t - U_t\|^2 \tag{2.7}$$

where $U_{t+1}$ is the output of the combined VAE-RNN in (2.6) and $\kappa$ is a weighing parameter.

Correspondingly, the loss function for the RNN component of the model contains a MSE between $\tilde{U}_{t+1}$ and $U_{t+1}$, as well as the $D_{KL}$ for the Variational Bayes. This additional term is important as $U_{t+1}$ is only one possible evolution of the system given $U_t$ (ensembles of time evolutions given

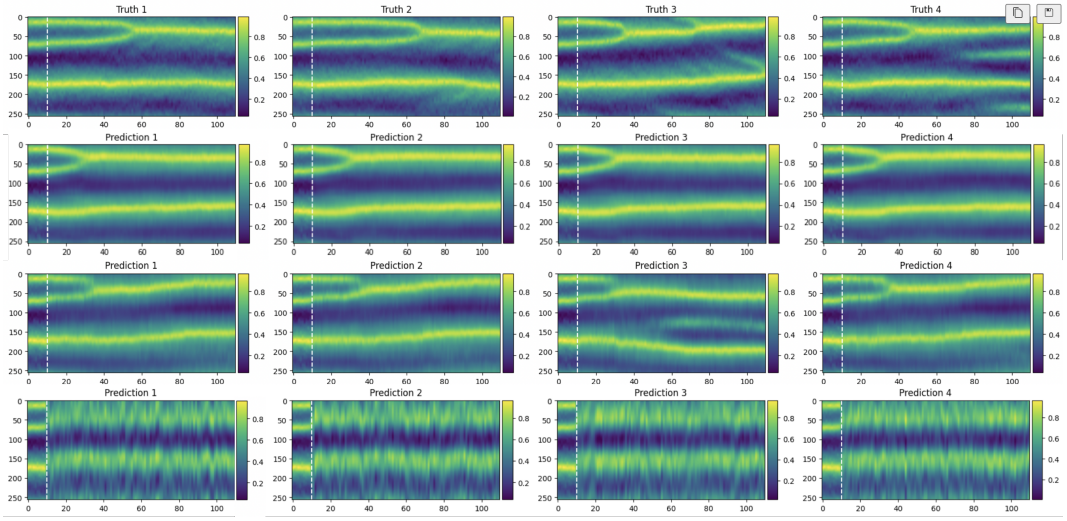


**Figure 4:** *A series of plots displaying the output from the VAE-RNN over a forecast period of 100 time-steps. a) Shows the Truth data obtained via numerical integration b) Shows an ensemble of outputs from the VAE, $\beta = 0.1$having been initially shown the 10 time-steps to the left of the dotted line as initial conditions, each forecasting the next 100 time-steps. The ensemble was produced by sampling the noise $\eta$ in the latent space. c) The same information as above, for the case where $\beta = 0.6$. d) The same information as above, for the case where $\beta = 1$. All the Neural Networks had a latent dimension of 64.*

identical initial conditions, but with different realisations of the stochastic forcing can be seen in the top four plots of 4), and the Variational Bayes allows us to include this stochasticity into the system.

Non-linear transformations are not distance-preserving, however, this is an advantageous property to have, both for interpretability of the latent space, but we also observed that it aids in numerical stability, during training, as well as inference. We assert that two states in observation space, if close together, should also be close together in our latent space. While this cannot be formally asserted in the model, what we can do is add a component to the loss function that intends to minimise the distance in latent space between our prediction, in latent space, $\tilde{z}_{t+1}$, and the encoded representation of $z_{t+1} := \mathcal{E}_\phi(U_{t+1})$ from our 'truth' training set Given that $\tilde{z}_{t+1}$ is one particular future time step, $\tilde{z}_{t+1}$ and $z_{t+1}$ should not be identical, but their distances should be as small as possible, in the same way, we impose this on $\tilde{U}_{t+1}$ and $U_{t+1}$ in (2.7) (we will evaluate how well the network captures the variation in the system due to the stochastic forcing in Section 2.4). Putting this together the loss function for the RNN is computed as follows:

$$\mathcal{L}_{RNN}(U_{t+1}, \tilde{U}_{t+1}, z_{t+1}, \tilde{z}_{t+1}, \mu, \sigma; \varphi; \beta, \lambda) =$$
$$\|\tilde{U}_{t+1} - U_{t+1}\|^2 + \lambda\|\tilde{z}_{t+1} - z_{t+1}\|^2 + \frac{\beta}{D}\sum_{j=1}^{D}\left(\frac{1}{2}(log\sigma_{i,j}^2 - \mu_{i,j}^2 - \sigma_{i,j}^2 + 1)\right) \quad (2.8)$$

It is important to note that the Encoder, Decoder and RNN are trained together so that the Encoder and Decoder appropriately learn spatial-temporal features, as opposed to just spatial features. The training algorithm is outlined in Algorithm 1.
The parameters $\{\beta, \kappa \ \& \ \lambda\} \in \mathbb{Z}$ are obtained through a hyperparameter sweep that uses Bayesian

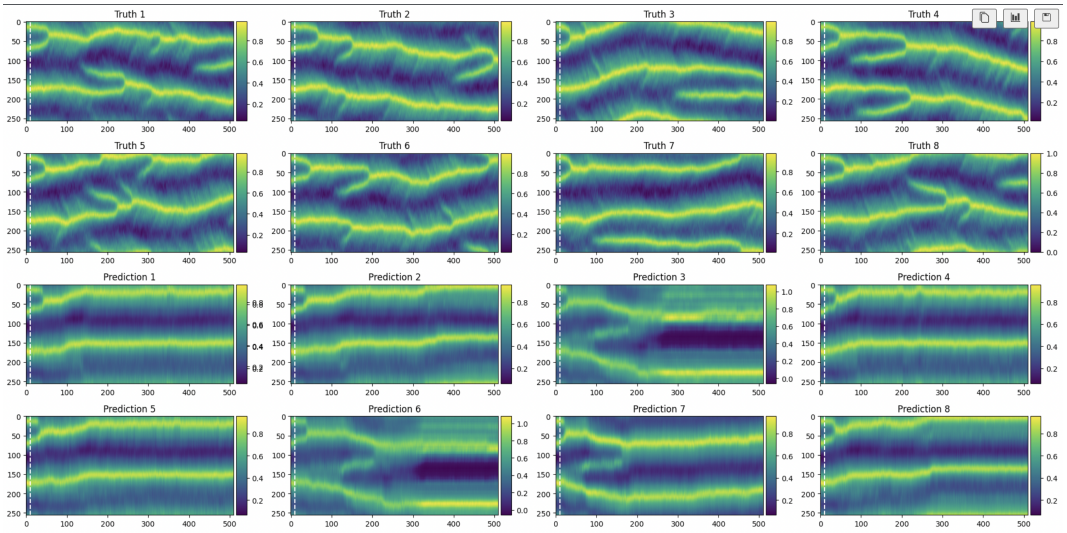


**Figure 5:** *Time-Latitude plots showing an ensemble of numerical integrations, each with different realisations of the random noise after t=10 (indicated by the dotted line), and an ensemble of 8 predictions made by the RNN-VAE when $\beta = 0.6$, for a forecast period of 500 timesteps, each shown the same initial 10 time-steps, the size of the latent space of the Neural Network was 128.*

optimisation [70], to find the optimal values, rather than through a grid search that would have been computationally expensive given the size of the search space. The details of this are omitted for brevity, but we will explore the impacts of varying one of these parameters, $\beta$.

Figure 4 shows predictions made by the RNN-VAE, for varying values of $\beta$. When $\beta \ll 1$ we only have a small $D_{KL}$ component in the loss function, leading to, predominately, a reconstruction task, with no variation in our predicted dynamics. (When $\beta$=0 we are unable to achieve meaningful learning as the model is trying to fit a deterministic mapping to the stochastically produced training set). When $\beta = 1$, the reconstructive $MSE$ and the measure of distribution distance $D_{KL}$ are equally weighted - one may expect this to achieve our goal of capturing variation within the numerical model, however, this appears to increase the vertical shift between time-steps, not the variation in dynamics, with the system predicting what is possible, the mean configuration across the entire dataset, with noise, ignoring the previous system history.

Due to the stochastic nature of the system, it is important to make comparisons between ensembles, as shown in Figure 4, rather than comparing individual realisations of both numerical integration and RNN-VAE outputs. While the configuration with $\beta = 0.6$, appears to give reasonable results, with unique dynamics visible in Prediction 3 with the formation of a new jet- the RNN-VAE fails to make longer-term predictions, with the models falling into different regime 'basins' after a set number of time-steps into the future, and this is likely due to the smoothing (a common result of VAEs due to the nature of the MSE objective function) causing errors to grow with each time step iteration, see Figure 5. Here, at later time steps, the model is asked to make predictions on inputs that lie outside the distribution of the training dataset due to growing errors, motivating including additional complexity in the RNN-VAE model architecture, to ensure that model outputs are statistically comparable to the training data.

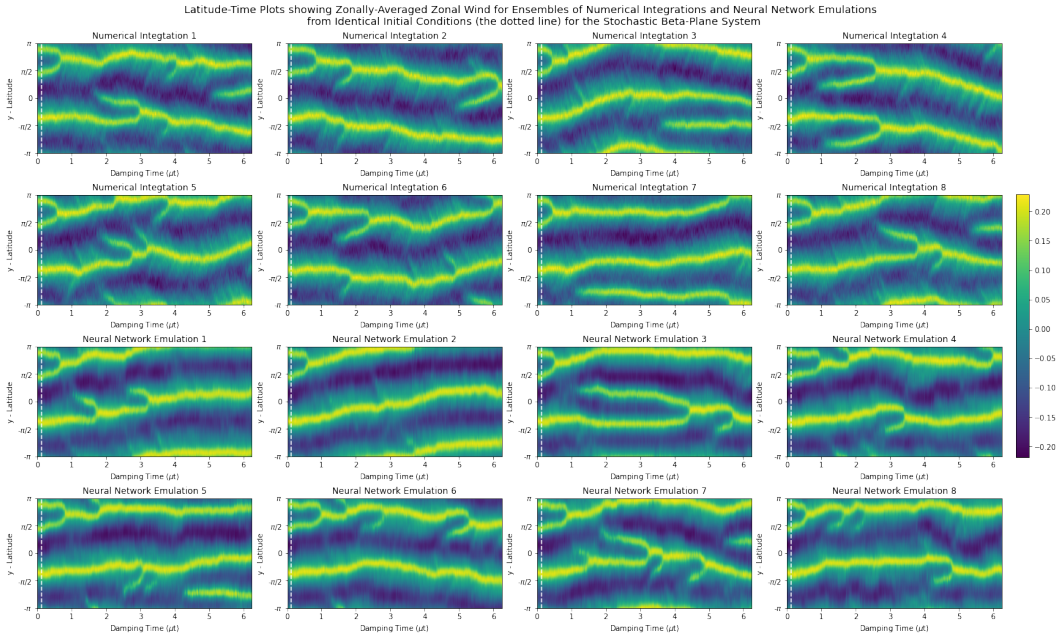


**Figure 6:** *Time-Latitude plots showing an ensemble of numerical integrations, each with different realisations of the random noise after t=10 (indicated by the dotted line), and an ensemble of 8 predictions made by the RNN-Adversarial-VAE when* $\beta = 0.6$*, for a forecast period of 500 timesteps -corresponding to 6.25 damping time units, each shown the same initial 10 time-steps, before the dotted line, the size of the latent space of the Neural Network was 128.*

## 2.3 Adversarial Regularisation

In order to improve the model we additionally train with an adversarial component, as outlined in Section 1.??. Here we simultaneously train a discriminator network that learns how to classify real (data from the training set of numerical integrations, $U$) and generated (outputs of the VAE, $\tilde{U}$), to force the VAE to alter how it generates outputs so that it can produce outputs the discriminator can no longer classify correctly, and thus are statically indistinguishable from the training data.

The discriminator is trained using (1.19). The same loss function is used to update the VAE, to encourage the network to produce samples that the Discriminator cannot correctly distinguish. As the first term in equation 1.19 does not contain parameters pertaining to the VAE (namely $\theta, \phi$ and $\varphi$), the following additional loss term is used:

$$\mathcal{L}(\tilde{U}) = \mathbb{E}_{\tilde{U}}[log(1 - D(\tilde{U}_i)] \tag{2.9}$$

where the gradient with respect to the Discriminator, $D$, is not being taken, it is used here for inference only. This term is appended to the loss functions (2.7) and (2.8), weighted by a parameter $\sigma$, which was again optimally found via a hyperparameter sweep.

The results of this RNN-Adversarial-VAE are shown in Figure 6 Initially, the results look very promising with a plausible range of realistic-looking dynamics that, by eye, could be considered indistinguishable from the numerical integrations. They display a marked improvement over RNN-VAE outputs in Figure 5, however, when we produce a very long emulation, as shown in Figure 7, we see that the system falls into unrealistic basins, with stationary dynamics (interestingly the system does right itself, before falling into another basin).

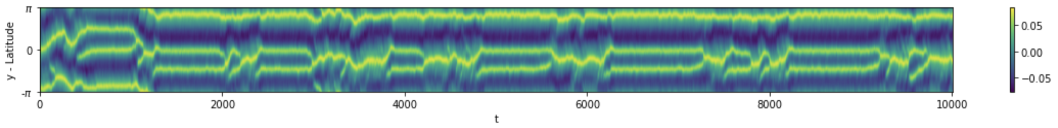


**Figure 7:** *Time-Latitude plot of a single prediction made by the RNN-Adversarial-VAE when $\beta = 0.6$, for a forecast period of 10000 timesteps, each shown the same initial 10 time-steps, the size of the latent space of the Neural Network was 128.*

**Figure 8:** *Spectral Density Energy Plot time-averaged spectra, averaged over 200 time steps. The green line shows results for data from numerical integration, while red line shows results from the RNN-Adversarial-VAE*

We hypothesised that the outputs of the model, while visually are very similar to the training data obtained via numerical integration, must exhibit some statistical bias, that the discriminator in adversarial training was unable to identify. More details on verification metrics are outlined in Section 2.4, however, we determined that the model exhibited a spectral bias.

## 2.4 Spectral Bias

Figure 8 shows that our model showed a spectral bias- this is that the model was correctly able to reproduce the lower frequencies of $U$, but unable to match the higher frequencies. Various reasons for why generative models suffer from this are given by [71], with the most compelling being that the lower magnitudes of the high frequencies are ignored by the discriminator as they contain a much smaller proportion of the energy. In order to overcome this, we instead input the *log* of the magnitudes of the components of the following Fourier series:

$$\mathbf{S} = \{a_k | k \in \mathbb{Z} \ with \ k < N_y/2\} \ \ where \ \ a_k = log\left(\left|\int_0^\infty U(y)e^{-iky}dy\right|\right) \tag{2.10}$$

to the discriminator to explicitly match the energy spectra, using the *log* to increase the relative magnitudes of the higher frequency modes. The architecture of the Discriminator was the same as previously and took the form of a Feed Forward Neural Network with Fully-Connected layers. An algorithm of the training process can be seen in Algorithm 2.

The results of this RNN-VAE with spectral adversarial training are shown in Figure 9. These initial results look very similar to those of 7, however, results in Figure 10 are incredibly promising, with these very long emulations successfully producing plausible dynamics, by eye, and not falling into unrealistic attractor basins. We show this for 3 different initial conditions to show that the neural network is robust to different system states. We are able to show this approach allows for the spectral properties of the model to be learned correctly in Figure 11.
One goal of this research was to produce a model that was significantly computationally more efficient than numerical integration. For the plots in Figure 9 each prediction took 3.9 seconds to produce 500 future time-steps, while the numerical integration takes $\sim$ 270 minutes to produce a time-series over the same period: $\sim$4150 times faster (all conducted on CPU, using the same 2 cores on a DAMTP PC - inference time for each prediction using the GPU on a 2021 MacBook Pro was 3.02 milliseconds, a speed-up of $\sim$53600 times - granted that the numerical code, written in MATLAB, had not been fully optimised to run faster, however, it was able to utilise multi-thread parallelisation).

**Algorithm 2** RNN-Adversarial-VAE Training Algorithm

$\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\varphi}$ $\boldsymbol{\chi}\leftarrow$ initialise network parameters
**for** epoch in epochs
  **for** mini-batch in batches
    $\boldsymbol{U}_{(t:t-l)} \leftarrow$ random mini-batch from training dataset

    # Forward Pass though the networks
    $\boldsymbol{Z}_{(t:t-l)} \leftarrow Encoder(\boldsymbol{U}_{(t:t-l)})$
    $\boldsymbol{\mu}_{(t+1)}, \boldsymbol{\sigma}_{(t+1)} \leftarrow RNN(\boldsymbol{Z}_{(t:t-l)})$
    $\tilde{\boldsymbol{Z}}_{(t+1)} \leftarrow$ Sample from prior: $\boldsymbol{Z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \boldsymbol{\mathcal{N}}(0, 1)$
    $\tilde{\boldsymbol{U}}_{(t+1)} \leftarrow$ Decoder$(\tilde{\boldsymbol{Z}}_{(t+1)})$
    $\boldsymbol{S} \leftarrow \log(|FFT(\boldsymbol{U}_{(t+1)})|)$
    $\tilde{\boldsymbol{S}} \leftarrow \log(|FFT(\tilde{\boldsymbol{U}}_{(t+1)})|)$

    $\boldsymbol{Z}_{(t+1)} \leftarrow Encoder(\boldsymbol{U}_{(t+1)})$
    $\boldsymbol{U}_{M(t)} \leftarrow Decoder(Encoder(\boldsymbol{U}_{(t)}))$

    # Calculate Losses
    $\mathcal{L}_{prior} \leftarrow D_{KL}(q(\boldsymbol{Z}|\boldsymbol{U}), p(\boldsymbol{Z}))$
    $\mathcal{L}_{reconstruction} \leftarrow \|\tilde{\boldsymbol{U}}_{(t+1)} - \boldsymbol{U}_{(t+1)}\|^2$
    $\mathcal{L}_{adversarial} \leftarrow \mathbb{E}[log(1 - D(\tilde{\boldsymbol{S}})]$
    $\mathcal{L}_{Discriminator} \leftarrow \mathbb{E}[logD_{\chi}(\boldsymbol{S})] + \mathbb{E}[log(1 - D_{\chi}(\tilde{\boldsymbol{S}})]$

    $\mathcal{L}_{AE} \leftarrow \|\tilde{\boldsymbol{U}}_{(t)} - \boldsymbol{U}_{M(t)}\|^2$
    $\mathcal{L}_{z} \leftarrow \|\tilde{\boldsymbol{Z}}_{(t+1)} - \boldsymbol{Z}_{(t+1)}\|^2$

    # Update parameters according to gradients
    $\boldsymbol{\theta}_{Encoder} \xleftarrow{+} -\nabla_{\boldsymbol{\theta}_{Encoder}} (\mathcal{L}_{reconstruction} + \kappa\mathcal{L}_{AE})$
    $\boldsymbol{\phi}_{Decoder} \xleftarrow{+} -\nabla_{\boldsymbol{\phi}_{Decoder}} (\mathcal{L}_{reconstruction} + \kappa\mathcal{L}_{AE} + \gamma\mathcal{L}_{Adversarial})$
    $\boldsymbol{\varphi}_{RNN} \xleftarrow{+} -\nabla_{\boldsymbol{\varphi}_{RNN}} (\mathcal{L}_{reconstruction} + \beta\mathcal{L}_{prior} + \lambda\mathcal{L}_{z})$
    $\boldsymbol{\chi}_{D} \xleftarrow{+} -\nabla_{\boldsymbol{\varphi}_{RNN}} (\mathcal{L}_{Discriminator})$

## 2.5 Evaluation Metrics

In order to verify how well the model has done, we must produce an evaluation metric that quantifies the model performance when making longer-time predictions. By definition, the loss function only evaluates the model's ability to make a prediction of a single time step, and given the system's stochastic nature, we are less concerned with a particular forecast, but with the statistical properties of the system. Therefore we need a new metric to evaluate the ability of the VAE to produce long-time statistical properties of the system attractor.

To provide a detailed representation of the system attractor, we use the joint probability density function (PDF) of the pairwise values of $U$, $U_y$, $U_t$ - to explore spatial correlations, with $U_y$ and $U_t$ not explicitly learned by the model and we hope that the model has been able to learn this distribution implicitly. Figure 11 shows this PDF a log scale for the numerical integration and our RNN=Adversarial-VAE. Here each point is the pairwise likelihood for each grid-point in y, containing samples for 100 time-steps, $t = [10 : 110]$, pooled for 8 different realisations of the random noise, which we shall define as $p(\mathbf{U}, \mathbf{U}_y, \mathbf{U}_t)$ and for 8 different predictions from the model,

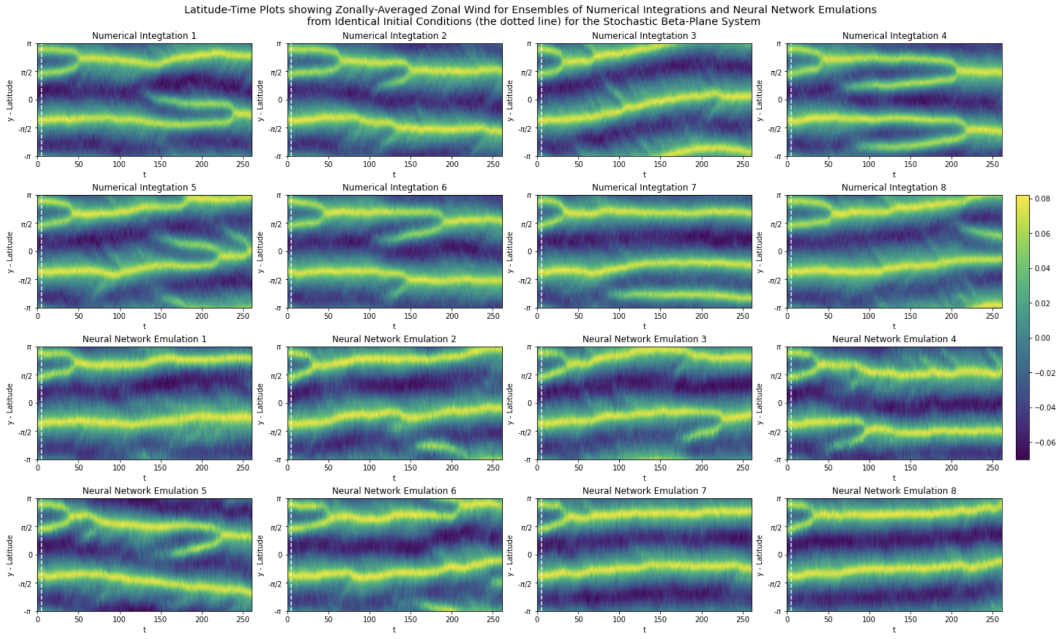


**Figure 9:** *Time-Latitude plots showing an ensemble of numerical integrations, each with different realisations of the random noise after t=10 (indicated by the dotted line), and an ensemble of 8 predictions made by the RNN-VAE with spectral adversarial training, when $\beta = 0.6$, for a forecast period of 500 timesteps -corresponding to 6.25 damping time units, each shown the same initial 10 time-steps, before the dotted line, the size of the latent space of the Neural Network was 128.*

defined as $q(\tilde{\mathbf{U}}, \tilde{\mathbf{U}}_y, \tilde{\mathbf{U}}_t)$. We can similarly define a metric for the time derivatives of the field, to evaluate temporal correlations, however, this has not yet been conducted.

The closer the two distributions, the better for the performance of the model in reconstructing the data. To quantify this and evaluate a scalar metric from the PDFs, we wish to determine the $D_{KL}$ between the PDF for the truth data over the prediction time and the output data from the VAE using the definition of the KL in (3.4):

$$metric = D_{KL}(p,q) = q(\tilde{\mathbf{U}}, \tilde{\mathbf{U}}_y, \tilde{\mathbf{U}}_t) log \frac{q(\tilde{\mathbf{U}}, \tilde{\mathbf{U}}_y, \tilde{\mathbf{U}}_t)}{p(\mathbf{U}, \mathbf{U}_y, \mathbf{U}_t)} \tag{2.11}$$

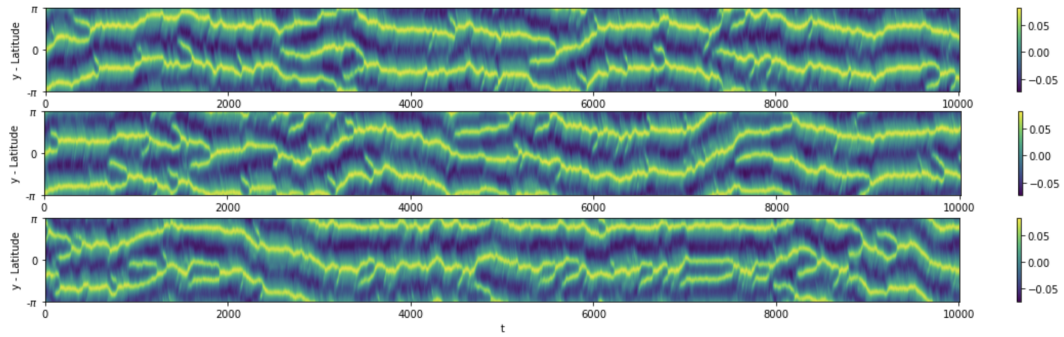


**Figure 10:** *Time-Latitude plots showing predictions made by the the RNN-VAE with spectral adversarial training of zonally averaged zonal wind $U(y,t)$ when $\beta = 0.6$, for a forecast period of 10000 timesteps, 10,000-time steps, from a-c) identical 2 jet state d) a 3 jet state initial conditions. The size of the latent space of the Neural Network was 128.*

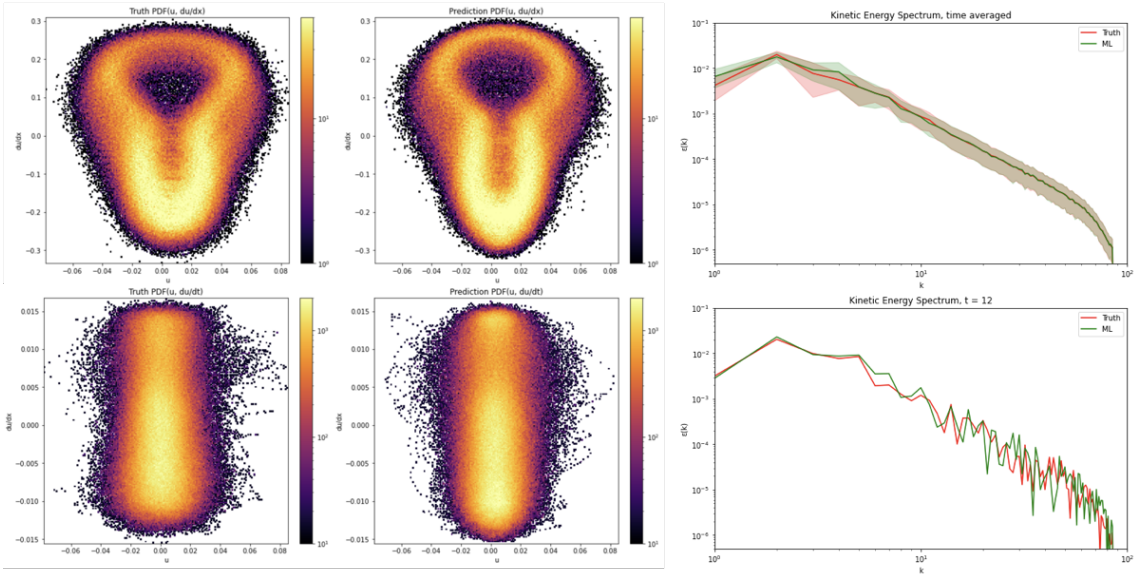


**Figure 11:** *The two top left plots on the show the PDFs of the pairwise values of $U$ and $U_y$ for the truth data (extreme left) and the spectral adversarial RNN-VAE, second to the left, and correspondingly, PDFs of the pairwise values of $U$ and $U_t$ below. The top right plot shows the Spectral Density Energy Plot time-averaged spectra, averaged over 200 time steps. The green line shows results for data from numerical integration, while the red line shows results from the RNN-VAE with spectral adversarial training, with associated standard deviation envelopes. The bottom right plot shows the same information as above, except for an single instantaneous time step.*

Neural Networks learn through stochastic gradient descent and are initialised with random initial conditions. As a result, no two models trained with identical architectures and learning parameters will be identical (without setting a seed). As a result, it is commonplace to train an ensemble of networks for each configuration and choose the best-performing model- where best performing results in the best metric score. We can use the PDF defined above as one metric of performance and compare the ensembles of models, using the $D_{KL}(p, q)$ between purely numerical integrations as the optimal value.

# 3 Concluding Remarks

We show that using a probabilistic neural network we are able to learn the dynamics of a stochastically forced system, namely the beta plane system, demonstrated via long-time evolution emulation, and by comparing corresponding statistical and spectral properties. The deep learning approach is able to produce forecasts four orders of magnitude faster than numerical integration, and the probabilistic model is able to produce ensembles of f possible time evolutions to capture the variability of the stochastic beta plane system.

There is still much work that needs to be done before publishing these results and outlined below are two obvious future directions of research that will be explored during this PhD:

## 3.1 Exploring Latent Space

One of the goals of the projects is to use the ROM to explore the system dynamics. An initial exploration will be to understand how the size of the latent space affect model performance, hoping to find a minimal size $D_M^*$ in which additional nodes no longer make a meaningful improvement to the model, corresponding to the underlying degrees of freedom of the Beta-Plane system and dimension of the internal manifold that the system lies on.

The ROM approach provides a latent space that could yield insight into both model robustness as well as underlying system dynamics. One question that arises is that of quantification of stability of different system states - how predictable are they. We will look to use both the ability to cheaply produce an ensemble as well as the latent variables to try and identify states stability [64] to produce a method that corresponds to considering Lyapunov exponents for a stochastic dynamical system [72].

In POD, the reduced modes are orthogonal by nature, however the latent space of an Autoencoder or VAE are not. To try and enforce this, the value of $\beta$ in equation 3.5 can be varied. $\beta$ acts as a regulariser that forms a trade-off between reconstruction fidelity and the quality of learning independent representations. Here the goal is to produce statistically-independent latent variables, by minimising the distance $D_{KL}[p(z|U)||\prod_i p(z_i|U)]$ between $q_\phi(z|U)$ and the product of its marginals. [73] looked to verify this by calculating the determinant of the cross-correlation matrix $\mathbf{R}$ for the latent variables, where R is:

$$R_{ii} = 1 \; and \; R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}} \tag{3.1}$$

for $1 \leq i \neq j \leq D$, where $D$ is the size of the latent space and $\mathbf{C}$ is the covariance matrix of the latent variables. $det_{\mathbf{R}=1}$ when all the variables are completely uncorrelated ($R_{ij} = 0$) and zero when they are completely correlated ($R_{ij} = 1$). $det_{\mathbf{R}}$ can be compared against the metric we evaluated in equation 3.6 for varying values of $\beta$, to visualise the trade-off between reconstruction fidelity and the quality of learning independent representations. [73] also explore how to rank the VAE modes by zeroing out all the latent variables except the ith variable for each $z_i$. The mode with the largest energy percentage when decoded into state-space is ranked as the first mode and so on, to determine the number of modes required to span the latent space adequately and best reproduce the model.

## 3.2 Moving to a Stratified System

Once a robust model has been finalised, future work could include forecasting of the full velocity field of the beta-Plane system as well as a two-layer model that can generate turbulence without stochastic forcing. Comparative exploration between the two systems will be made - and it may be that the 3D model is still best captured best by a stochastic ROM due to sub-grid variability. This could also include the exploration of other geophysical process models, or more complex geometries and parameterisations including those used in operational forecasts.

# References


[1] Cope. L. The dynamics of geophysical and astrophysical turbulence. *(Doctoral thesis), DAMTP, University of Cambridge. 2020.*

[2] MacCracken M. C. Grotch, S. L. The use of general circulation models to predict regional climatic change. *Journal of Climatet*, 4, 1991.

[3] Huize Wang and Robin Wordsworth. Extremely long convergence times in a 3d GCM simulation of the sub-neptune gliese 1214b. *The Astrophysical Journal*, 891(1):7, feb 2020.

[4] Philippe Lopez and Marco Matricardi. Validation of ifs+rttov/mfasis0.64-m reflectances against observations from goes-16, goes-17, msg-4 and himawari-8. (903), 09 2022.

[5] Anton Beljaars, Peter Bechtold, M. Köhler, J.-J. Morcrette, A.M. Tompkins, P. Viterbo, and Nils Wedi. *The numerics of physical parameterization.* PhD thesis, Shinfield Park, Reading, 2004 2004.

[6] Raju Pathak, Hari Prasad Dasari, Samah El Mohtar, Aneesh C. Subramanian, Sandeep Sahany, Saroj Kanta Mishra, Omar M. Knio, and Ibrahim Hoteit. Uncertainty quantification and bayesian inference of cloud parameterization in the ncar single column community atmosphere model (scam6). In *Frontiers in Climate*, 2021.

[7] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20, 1963.

[8] EDWARD N. LORENZ. The predictability of a flow which possesses many scales of motion. *Tellus*, 21(3):289–307, 1969.

[9] Frank B. Lipps. A note on the beta-plane approximation. *Tellus*, 1964.

[10] Thomas Oudar, Julien Cattiaux, and Hervé Douville. Drivers of the northern extratropical eddy-driven jet change in cmip5 and cmip6 models. *Geophysical Research Letters*, 47(8):e2019GL086695, 2020. e2019GL086695 10.1029/2019GL086695.

[11] Rachel Robinson, Jacob Scheff, and Nicholas Golden. Cmip6 captures the satellite-era jet slowdown and arctic amplification - yet projects future jet speedup and tropical amplification, 2023.

[12] Sihua Huang, Bin Wang, and Zhiping Wen. Dramatic weakening of the tropical easterly jet projected by cmip6 models. *Journal of Climate*, 33(19):8439 – 8455, 2020.

[13] Paul Edwin Curtis, Paulo Ceppi, and Giuseppe Zappa. Role of the mean state for the southern hemispheric jet stream response to co2 forcing in cmip6 models. *Environmental Research Letters*, 15(6):064011, may 2020.

[14] J. Dorrington, K. Strommen, and F. Fabiano. Quantifying climate model representation of the wintertime euro-atlantic circulation using geopotential-jet regimes. *Weather and Climate Dynamics*, 3(2):505–533, 2022.

[15] B. J. Harvey, P. Cook, L. C. Shaffrey, and R. Schiemann. The response of the northern hemisphere storm tracks and jet streams to climate change in the cmip3, cmip5, and cmip6 climate models. *Journal of Geophysical Research: Atmospheres*, 125(23):e2020JD032701, 2020. e2020JD032701 2020JD032701.

[16] TN Palmer. Towards the probabilistic earth-system simulator: A vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 138(665):841–861, 2012.

[17] T.N. Palmer, Roberto Buizza, F. Doblas-Reyes, T. Jung, Martin Leutbecher, G.J. Shutts, M. Steinheimer, and Antje Weisheimer. Stochastic parametrization and model uncertainty. (598):42, 10 2009.

[18] David John Gagne, Hannah M. Christensen, Aneesh C. Subramanian, and Adam H. Monahan. Machine learning for stochastic parameterization: Generative adversarial networks in the lorenz '96 model. *Journal of Advances in Modeling Earth Systems*, 12(3), mar 2020.

[19] Arthur P. Guillaumin and Laure Zanna. Stochastic-deep learning parameterization of ocean momentum forcing. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002534. e2021MS002534 2021MS002534.

[20] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson Ellen Clancy, Alberto Arribas†, and Shakir Mohamed. Skillful precipitation nowcasting using deep generative models of radar. *Nature*, 597:672–677, 2021.

[21] Peter B. Rhines. Waves and turbulence on a beta-plane. *Journal of Fluid Mechanics*, 69(3):417–443, 1975.

[22] Gareth P. Williams. Planetary circulations: 1. barotropic representation of jovian and terrestrial turbulence. *Journal of Atmospheric Sciences*, 35(8):1399 – 1426, 1978.

[23] Geoffrey K. Vallis and Matthew E. Maltrud. Generation of mean flows and jets on a beta plane and over topography. *Journal of Physical Oceanography*, 23(7):1346 – 1362, 1993.

[24] C-G. Rossby. Relation between variations in the intensity of the zonal circulation of the atmosphere and the displacements of the semi-permanent centers of action. *Journal of Marine Research*, 2:38–55, 1939.

[25] J. G. Charney. On the scale of atmospheric motions. *Geofys. Publ. Oslo., 17(2):1–17.*, 1948.

[26] R. K. Scott and L. M. Polvani. Equatorial superrotation in shallow atmospheres. *Geophysical Research Letters*, 35(24), 2008.

[27] V.W. Ekman. *On the Influence of the Earth's Rotation on Ocean-currents: With a Biography of the Author*. Almqvist & Wiksells boktr., 1951.

[28] Oliver Bühler. *Waves and Mean Flows*. Cambridge Monographs on Mechanics. Cambridge University Press, 2 edition, 2014.

[29] Kaushik Srinivasan and William Young. Zonostrophic instability. *Journal of Atmospheric Sciences*, 69:1633–1656, 05 2012.

[30] D. G. Dritschel and M. E. McIntyre. Multiple jets as pv staircases: The phillips effect and the resilience of eddy-transport barriers. *Journal of the Atmospheric Sciences*, 65(3):855 – 874, 2008.

[31] P. H. Haynes, D. A. Poet, and E. F. Shuckburgh. Transport and mixing in kinematic and dynamically consistent flows. *Journal of the Atmospheric Sciences*, 64(10):3640 – 3651, 2007.

[32] Rintoul S. R. Sokolov, S. Multiple jets of the antarctic circumpolar current south of australia. *Journal of Physical Oceanography*, 37, 2007.

[33] Tim Woollings, Abdel Hannachi, and Brian Hoskins. Variability of the north atlantic eddy-driven jet stream. *Quarterly Journal of the Royal Meteorological Society*, 136(649):856–868, 2010.

[34] James M. Hyman and Basil Nicolaenko. The kuramoto-sivashinsky equation: A bridge between pde's and dynamical systems. *Physica D: Nonlinear Phenomena*, 18(1):113–126, 1986.

[35] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan 2015.

[36] Tapio Schneider, Shiwei Lan, Andrew Stuart, and João Teixeira. Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophysical research letters*, 44(24):12,396–12,417, 2017.

[37] F. Chevallier, F. Chéruy, N. A. Scott, and A. Chédin. A Neural Network Approach for a Fast and Accurate Computation of a Longwave Radiative Budget. *Journal of Applied Meteorology*, 37(11):1385–1397, November 1998.

[38] Vladimir M. Krasnopolsky, Michael S. Fox-Rabinovitz, and Dmitry V. Chalikov. New approach to calculation of atmospheric model physics: Accurate and fast neural network emulation of longwave radiation in a climate model. *Monthly Weather Review*, 133(5):1370 – 1383, 2005.

[39] Pierre Gentine, M. Pritchard, Stephan Rasp, Gael Reinaudi, and Galen Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45:5742 – 5751, 2018.

[40] Stephan Rasp, Michael S. Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39):9684–9689, 2018.

[41] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[42] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, January 2021.

[43] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[44] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[45] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[46] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[47] Jussi Leinonen, Daniele Nerini, and Alexis Berne. Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network. *IEEE Transactions on Geoscience and Remote Sensing*, 59(9):7211–7223, 2021.

[48] Jussi Leinonen, Daniele Nerini, and Alexis Berne. Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network. *IEEE Transactions on Geoscience and Remote Sensing*, 59(9):7211–7223, sep 2021.

[49] Henry Addison, Elizabeth Kendon, Suman Ravuri, Laurence Aitchison, and Peter Watson. Machine learning emulation of a local-scale uk climate model. In *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*, 2022.

[50] Predictive learning as a network mechanism for extracting low-dimensional latent space representations. *Nature Communications*, 12(1):1417–1417, 2021.

[51] M-A Nikolaidis, B F Farrell, P J Ioannou, D F Gayme, A Lozano-Durán, and J Jiménez. A POD-based analysis of turbulence in the reduced nonlinear dynamics system. *Journal of Physics: Conference Series*, 708:012002, apr 2016.

[52] Clarence W. Rowley, Tim Colonius, and Richard M. Murray. Model reduction for compressible flows using pod and galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1):115–129, 2004.

[53] Zhenghai Wang, Dunhui Xiao, Fangxin Fang, R. Govindan, and Christopher Pain. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86, 07 2017.

[54] Alec J. Linot and Michael D. Graham. Deep learning to discover and predict dynamics on an inertial manifold. *CoRR*, abs/2001.04263, 2020.

[55] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120:024102, Jan 2018.

[56] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

[57] Alberto Racca and Luca Magri. Statistical prediction of extreme events from small datasets, 2022.

[58] Mitsunori Noguchi. Geometry of statistical manifolds. *Differential Geometry and its Applications*, 2(3):197–222, 1992.

[59] Todd L. Parsons and Tim Rogers. Dimension reduction for stochastic dynamical systems forced onto a manifold by large drift: a constructive approach with examples from theoretical biology, 2015.

[60] Xue-Mei Li. Stochastic differential equations on noncompact manifolds: moment stability and its topological consequences. 2019.

[61] Sebastian Ullmann, Christopher Müller, and Jens Lang. Stochastic galerkin reduced basis methods for parametrized linear convection–diffusion–reaction equations. *Fluids*, 6(8), 2021.

[62] Xingye Kan, Jinqiao Duan, Ioannis G. Kevrekidis, and Anthony J. Roberts. Simulating stochastic inertial manifolds by a backward-forward approach. *SIAM Journal on Applied Dynamical Systems*, 12(1):487–514, 2013.

[63] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[64] Jacob Page, Michael P. Brenner, and Rich R. Kerswell. Revealing the state space of turbulence using machine learning. *Phys. Rev. Fluids*, 6:034402, Mar 2021.

[65] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation, 2021.

[66] Sajid A. Marhon, Christopher J. F. Cameron, and Stefan C. Kremer. *Recurrent Neural Networks*, pages 29–65. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[67] Kensuke Nakamura, Bilel Derbel, Kyoung-Jae Won, and Byung-Woo Hong. Learning-rate annealing methods for deep neural networks. *Electronics*, 10(16), 2021.

[68] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[69] Zijian Niu, Ke Yu, and Xiaofei Wu. Lstm-based vae-gan for time-series anomaly detection. *Sensors*, 20(13), 2020.

[70] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In Hugo Jair Escalante and Katja Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 3–26. PMLR, 06–12 Dec 2021.

[71] Katja Schwarz, Yiyi Liao, and Andreas Geiger. On the frequency bias of generative models, 2021.

[72] Olivier Martin. Lyapunov exponents of stochastic dynamical systems. *Journal of Statistical Physics*, 41(1-2):249–261, October 1985.

[73] Hamidreza Eivazi, Soledad Le Clainche, Sergio Hoyas, and Ricardo Vinuesa. Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows, 2021.

[74] F. Bouchet, J. B. Marston, and T. Tangarife. Fluctuations and large deviations of reynolds stresses in zonal jet dynamics. *Physics of Fluids*, 30(1):015110, 2018.

[75] Navid C. Constantinou, Brian F. Farrell, and Petros J. Ioannou. Emergence and equilibration of jets in beta-plane turbulence: Applications of stochastic structural stability theory. *Journal of the Atmospheric Sciences*, 71(5):1818 – 1842, 2014.

# A Appendices

## A.1

If samples $\mathbf{X} = \{\mathbf{x}_i ... \mathbf{x}_n\}$, drawn from distribution $p(\mathbf{x})$, are assumed to be i.i.d (independently identically distributed), the conditional log-likelihood estimator $\theta_{ML} = arg\max_\theta P(Y|X,\theta)$, where $\mathbf{Y} = \{\mathbf{y}_i ... \mathbf{y}_n\}$ are the observed targets. We can decompose $\theta_{ML}$ into:

$$\theta_{ML} = arg\max_\theta \prod_{i=1}^{n} P(y_i|\mathbf{x}_i,\theta) \tag{A.1}$$

$$= arg\max_\theta \sum i = 1^n P(y_i|x_i,\theta) \tag{A.2}$$

Assuming that our distribution is Gaussian:

$$p(u,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}) \tag{A.3}$$

Inserting this into (A.2) for batch size, B:

$$\sum_{i=1}^{m} logp(q(z_i|u_i)|p(z_i) = -mlog\sigma - \frac{m}{2}log(2\pi) - \frac{1}{B}\sum_{i=1}^{B}\frac{\|\tilde{y}_i - y_i\|^2}{2\sigma^2} \tag{A.4}$$

The first two terms are constant and will not be learned, and thus we can replace the cross entropy loss with the Mean Squared Error used for the Autoenocder as our reconstructive loss.
Thus we define our loss function as:

$$-ELBO(\theta,\phi) = MSE + \beta D_{KL} \tag{A.5}$$

$$= \frac{1}{B}\sum_{i=1}^{B}\left(\|\widehat{U}_i - U_i\|^2 + \frac{\beta}{D}\sum_{j=1}^{D}\left(\mathbb{E}_{z\sim q_\phi(z|u)}\left[q_\phi(z_{i,j}|u_i)log\frac{q_\phi(z_{i,j}|u_i)}{p_\theta(z_{i,j}|u_i)}\right]\right)\right) \tag{A.6}$$

**A.2**

$$D_{KL} = \mathbb{E}_{z\sim q_\phi(z|u)}\left[log\frac{q_\phi(z|u)}{p_\theta(z|u)}\right] \tag{A.7}$$

$$= \mathbb{E}_{z\sim q_\phi(z|u)} log q_\phi(z|u) - \mathbb{E}_{z\sim q_\phi(z|u)} log p_\theta(z|u) \tag{A.8}$$

$$= \int q_\phi(z|u) log q_\phi(z|u) dz - \int q_\phi(z|u) log p_\theta(z|u) dz \tag{A.9}$$

Taking the first term in (A.9) and assuming that $q_\phi(z|u)$ takes the form:

$$\int q_\phi(z|u) log q_\phi(z|u) dz = \int q_\phi(z|u) log \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} dz \tag{A.10}$$

$$= \int q_\phi(z|u)[-\frac{(z-\mu)^2}{2\sigma^2} - log(2\pi\sigma^2)^{\frac{1}{2}})dz \tag{A.11}$$

$$= -\frac{1}{2\sigma^2}\int (z-\mu)^2 q_\phi(z|u)dz - \frac{1}{2} log(2\pi\sigma^2)\int q_\phi(z|u)dz \tag{A.12}$$

Using the definition $\int (x-\mu)^2 p(x)dx = \sigma^2$ and that $\int p(x)dx = 1$:

$$\int q_\phi(z|u) log q_\phi(z|u) dz = -\left(\frac{1}{2\sigma^2}.\sigma^2 + \frac{1}{2}\log(2\pi\sigma^2).1\right) \tag{A.13}$$

$$= \frac{1}{2}(1 + \log(2\pi\sigma^2)) \tag{A.14}$$

Taking the second term in (A.9) and asserting that $p_\theta(z)$ takes the form $\mathcal{N}(0,1)$:

$$-\int q_\phi(z|u) log p_\theta(z) dz = -\int q_\phi(z) \log \frac{1}{(2\pi)^{\frac{1}{2}}} e^{-\frac{z^2}{2}} dz \tag{A.15}$$

$$= -\int q_\phi(z)[-\frac{z^2}{2} - \frac{1}{2}\log(2\pi)]dz \tag{A.16}$$

$$= -\frac{1}{2}(\int z^2 q_\phi(z)dz + \log(2\pi)\int q_\phi(z)dz) \tag{A.17}$$

$$\tag{A.18}$$

Using the definition $\int p(z)dz = 1$ and as $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0,1)$, $\mathbb{E}_z z = \mu + \sigma$, $\int z dz = \int \mu + \sigma dz$:

$$-\int q_\phi(z|u) log p_\theta(z|u) dz = \frac{1}{2}(\int [(z-\mu)^2 + 2\mu z - \mu^2] q_\phi(z)dz + \log(2\pi).1) \tag{A.19}$$

$$= \frac{1}{2}(\int (z-\mu)^2 q_\phi(z)dz + 2\mu\int z q_\phi(z)dz - \mu^2\int q_\phi(z)dz + \log(2\pi)) \tag{A.20}$$

$$\tag{A.21}$$

Using $\int zp(z)dz = \mu$:

$$-\int q_\phi(z|u)logp_\theta(z|u)dz = \frac{1}{2}(\sigma^2 + 2\mu^2 - \mu^2 + \log 2\pi) \tag{A.22}$$

$$= \frac{1}{2}(\sigma^2 + \mu^2 + \log 2\pi) \tag{A.23}$$

Adding (A.14) and (A.23):

$$D_{KL}(q_\phi(z|u), p_\theta(z)) = -\frac{1}{2}(1 + \log(2\pi\sigma^2)) + \frac{1}{2}(\sigma^2 + \mu^2 + \log 2\pi) \tag{A.24}$$

$$= -\frac{1}{2}(log\sigma^2 - \mu^2 - \sigma^2 + 1) \tag{A.25}$$

As we are taking the gradient of the $D_{KL}$, the term $\frac{1}{2}log2\pi$ can be omitted.

## A.3 Implementation of the Numerical Model of the Beta-Plane System

### A.3.1 Vorticity Field

Direct numerical simulation of the vorticity equations is implemented using a pseudo-spectral algorithm together with a standard 2/3 dealiasing rule. Time integration is performed in Fourier space using a second-order Runge-Kutta scheme, while nonlinear terms are evaluated in physical space. The vorticity equation is used to integrate the vorticity, $\zeta$, forwards in time, while the elliptic equation $\zeta = \nabla^2\psi$ is used to determine the streamfunction, $\psi$ , from the vorticity field.

Integration takes place over a two-dimensional, doubly-periodic, square domain $(x, y) \in [-\pi, \pi)^2$, with each coordinate discretised with $N = 256$ grid points. The time step, $\Delta t$, is held fixed throughout each simulation run, selecting an optimal value within the limits of numerical stability, with smaller values required for larger zonostrophy, with this typically lying in the range $0.0025 \ll \Delta t \ll 0.01$.

All simulations are initialised with $u = \zeta = 0$ and all parameters were held fixed within each simulation. The time taken for the total kinetic energy to reach a statistically steady state during the initial spin-up period depends on the linear damping rate, $\mu$. This is usually achieved by time $\mu t = 2 - 3$.

Each of the variables can be written in terms of its discretised Fourier transform relating spatial coordinates $\mathbf{x} = (x, y)$ to wavevectors $\mathbf{k} = (k_x, k_y)$. For the case of vorticity (1.1), this is given by:

$$\tilde{\zeta}(\mathbf{k}, t) = \sum_{\mathbf{x}=-\pi}^{\pi} \zeta(\mathbf{x}, t)e^{-i\mathbf{k}.\mathbf{x}}, \qquad \zeta(\mathbf{x}, t) = \frac{1}{N^2} \sum_{\mathbf{k}=-N/2}^{N/2} \tilde{\zeta}(\mathbf{k}, t)e^{i\mathbf{k}.\mathbf{x}}, \tag{A.26}$$

with the allowed wavenumbers in each direction:

$$k_x; k_y \in \frac{1}{L_D}\left(-\frac{N}{2}+1, -\frac{N}{2}+2, ..., 0, 1, ..., \frac{N}{2}\right), \tag{A.27}$$

where it assumed that N is even. Each vorticity equation is a stochastic partial differential equation that can be written in terms of linear and nonlinear operators:

$$\frac{\partial}{\partial t}\zeta(\mathbf{x},t) = \mathcal{L}(\zeta(\mathbf{x},t)) + \mathcal{N}(\zeta(\mathbf{x},t),\zeta(\mathbf{x},t)) + \xi(\mathbf{x},t). \tag{A.28}$$

Equation (1.1) is used to integrate the vorticity forwards in time while the elliptic equation $\zeta = \nabla^2\psi$ is used to determine the streamfunction from the vorticity field. Applying the Fourier transform (2.1) both of these equations, we obtain the transformed pair:

$$\frac{\partial}{\partial t}\tilde{\zeta}(\mathbf{k},t) = \mathcal{L}(\tilde{\zeta}(\mathbf{k},t)) + \widetilde{\mathcal{N}(\zeta,\zeta)}(\mathbf{k},t) + \tilde{\xi}(\mathbf{k},t), \tag{A.29}$$

$$\tilde{\zeta}(\mathbf{k},t) = -k^2\tilde{\psi}(\mathbf{k},t), \tag{A.30}$$

where we note that the Fourier transform commutes with the linear operator but does not commute with the nonlinear operator. Consequently, the linear and stochastic terms can be time-stepped entirely in Fourier space using standard schemes, however the nonlinear terms must be computed in physical space before being transformed back to Fourier space. The scheme that we use is a second-order Runge-Kutta algorithm:

$$\frac{\tilde{\zeta}(\mathbf{k},t+\Delta t)-\tilde{\zeta}(\mathbf{k},t)}{\Delta t} = \frac{1}{2}\mathcal{L}\left(\tilde{\zeta}(\mathbf{k},t)+\tilde{\zeta}(\mathbf{k},t+\Delta t)\right) + \widetilde{\mathcal{N}(\zeta,\zeta)}(\mathbf{k},t+\frac{1}{2}\Delta t) + \tilde{\xi}(\mathbf{k},t), \tag{A.31}$$

in which the stochastic term is piecewise constant on discrete time steps. In order to compute the nonlinear terms at the mid-point $t+\frac{1}{2}\Delta t$ between time steps, we first compute an estimate of the vorticity at time $t+\Delta t$ in which the nonlinear terms are evaluated at time t:

$$\tilde{\zeta}_{est}(\mathbf{k},t+\Delta t) = \frac{\left(1+\frac{1}{2}\Delta t\mathcal{L}\right)\tilde{\zeta}(\mathbf{k},t) + \left[\widetilde{\mathcal{N}(\zeta,\zeta)}(\mathbf{k},t) + \tilde{\xi}(\mathbf{x},t)\right]\Delta t}{1-\frac{1}{2}\Delta t\mathcal{L}}, \tag{A.32}$$

An estimate of the Fourier-transformed vorticity field at the mid-point $t+\frac{1}{2}\Delta t$ is then computed as

$$\tilde{\zeta}(\mathbf{k},t+\frac{1}{2}\Delta t) = \frac{1}{2}\left(\tilde{\zeta}_{est}(\mathbf{k},t+\Delta t)+\tilde{\zeta}(\mathbf{k},t)\right) \tag{A.33}$$

from which we can re-evaluate the nonlinear terms after reverting to physical space. Finally, the algorithm permits the computation of the vorticity field at the new time $t+\Delta t$ using a rearrangement of (A.31):

$$\tilde{\zeta}(\mathbf{k}, t+\Delta t) = \frac{\left(1+\frac{1}{2}\Delta t\mathcal{L}\right)\tilde{\zeta}(\mathbf{k},t) + \left[\widetilde{\mathcal{N}(\zeta,\zeta)}(\mathbf{k}, t+\frac{1}{2}\Delta t) + \tilde{\xi}(\mathbf{x},t)\right]\Delta t}{1-\frac{1}{2}\Delta t\mathcal{L}}, \tag{A.34}$$

### A.3.2 Forcing

To ensure that zonal flows arise spontaneously rather than being directly forced, the fluid is stirred using a stochastic vorticity force $\xi(x, y, t)$ with zero mean, $\langle\xi(\mathbf{x}, t)\rangle = 0$ and a two-point, two-time correlation function described by $\Xi$. The forcing is associated with a decorrelation time $\tau_f$, which the limit $\tau_f \to 0$ corresponds to white noise - in which the forcing and the system are uncorrelated, leading the correlation function becoming delta-correlated in time:

$$\langle\xi(\mathbf{x}_1, t_1)\xi(\mathbf{x}_2, t_2)\rangle = \Xi(r)\delta(t_2 - t_1), \tag{A.35}$$

where the angular brackets here denote an ensemble average and as a further idealisation, we shall restrict attention to spatially homogeneous and isotropic forcing distributions such that $\Xi$ depends only on the two-point separation distance:

$$r = |\mathbf{x}_1 - \mathbf{x}_2| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{A.36}$$

$\Xi(r)$ is related to its Fourier modes, $\tilde{\Xi}(k)$, according to:

$$\tilde{\Xi}(k) = 2\pi\int_0^\infty \Xi(r)J_0(kr)rdr, \qquad \Xi(r) = \frac{1}{2\pi}\int_0^\infty \tilde{\Xi}(k)J_0(kr)kdk, \tag{A.37}$$

using the fact that the two-dimensional Fourier transform of a circularly symmetric function is equivalent to a Hankel transform of order zero. Here, $k = \sqrt{k_x^2 + k_y^2}$ is the isotropic wavenumber in Fourier space, $J_0$ is the Bessel function of order zero and $\tilde{\Xi}$ is Hermitian (i.e. $\tilde{\Xi}(\mathbf{k}) = \tilde{\Xi}^*(\mathbf{-k})$) so that $\Xi$ is real-valued.

A *narrow-band* forcing, consists of an annulus of wavevectors is excited in Fourier space, with mean radius $k = k_f$ and thickness $2\delta \ll k_f$. Here the further idealisation of the ring forcing is considered, taking the limmit $\delta k \to 0$, in which the energy is injected onto the circle $k = k_f$ in Fourier space. Spectral theories of zonal jet formation require a separation of scales between the forcing wavenumber, $k_f$, and the scale at which the jets equilibrate in order that a steep zonostrophic inverse cascade can develop with a large accumulation of energy at the largest zonal scales.

The stochastic forcing $\xi(\mathbf{x}, t)$ is numerically implemented as a first-order Markov process, approximated to be piecewise constant on discrete time steps $t_n < t < t_{n+1}$. Applying the discretised Fourier theorem (equivalent to equation(A.27)),

$$\tilde{\xi}(\mathbf{k},t) = \sum_{\mathbf{x}=-\pi}^{\pi} \xi(\mathbf{x},t)^{-i\mathbf{k}.\mathbf{x}}, \qquad \xi(\mathbf{x},t) = \frac{1}{N^2} \sum_{\mathbf{k}=-N/2}^{N/2} \tilde{\xi}(\mathbf{k},t)^{i\mathbf{k}.\mathbf{x}}, \tag{A.38}$$

the Fourier coeffcients $\tilde{\xi}$ are given by $\tilde{\xi}(\mathbf{k},t) = \mathcal{A}\tilde{\eta}(\mathbf{k},t)$ with constant amplitude $\mathcal{A} \in \mathbb{R}$ and Markovian coeffcients $\tilde{\eta}$ satisfying the Hermitian property (i.e. $\tilde{\eta}(\text{-}\mathbf{k},t) = \tilde{\eta}^*(\mathbf{k},t)$) to again ensure that the forcing function $\xi(\mathbf{x},t)$ is real. These coefficients are updated at each time step tn using the scheme

$$\tilde{\eta}(\mathbf{k},t_0) = X_0, \qquad \tilde{\eta}(\mathbf{k},t_n) = \gamma\tilde{\eta}(\mathbf{k},t_{n-1}) + \sqrt{1-\gamma^2}X_n, \tag{A.39}$$

where the $X_n$ are independent, identically distributed random variables given by $X_n = e^{i\theta_n}$ with uniformly distributed random phase $\theta_n \in [0, 2\pi)$, mean $\langle X_n \rangle = 0$ and covariance $\langle X_m X_n^* \rangle = \delta_{mn}$. The parameter $0 \leq \gamma \leq 1$ is related to the decorrelation time $\tau_f$ of the forcing by $\gamma = 1 - \Delta t/\tau_f$,where $\gamma = 0$ (corresponding to $\tau = \Delta t$) approximates Gaussian white noise using a Markov process, $\gamma = 1$ corresponds to deterministic forcing, and $0 < \gamma < 1$ represents forcing with a Markovian time-dependence. This forcing scheme ensures that the variance of its Fourier coeffcients remains constant over time, given by $\langle \tilde{\xi}(\mathbf{k},t)\tilde{\xi}(\mathbf{k},t) \rangle = \mathcal{A}^2$.

We numerically implement idealised ring forcing by selecting only those wavevectors such that $|k - k_f| < \delta k$ (where $\delta k \ll k_f$) and specifying that their Fourier coeffcients have equal amplitude. For short decorrelation times, we can assume that the stochastic forcing is independent of the state of the system, allowing the rate of energy injection, $\epsilon$, to be prescribed via the amplitudes $\mathcal{A}$ of the Fourier coeffcients of the forcing function. If, in addition, we assume ergodicity such that spatial and ensemble averages are equivalent, then the relationship between $\epsilon$ and $\mathcal{A}$ can be derived from $\epsilon = -\langle \xi\psi \rangle$,

$$\epsilon = \frac{1}{2}\left(\frac{1+\gamma}{1-\gamma}\right)\frac{\mathcal{N}\mathcal{A}^2\Delta t}{N^4 k_f^2}, \qquad \mathcal{A} = \sqrt{\frac{2\epsilon N^4 k_f^2}{\mathcal{N}\Delta t}\left(\frac{1-\gamma}{1+\gamma}\right)}, \tag{A.40}$$

Consequently, the stochastic forcing function is defined as

$$\xi(\mathbf{x},t) = \sqrt{\frac{2\epsilon k_f^2}{\mathcal{N}\Delta t}\left(\frac{1-\gamma}{1+\gamma}\right)} \sum_{|k-k_f|<\delta k} \tilde{\eta}(\mathbf{k},t)e^{i\mathbf{k}.\mathbf{x}}. \tag{A.41}$$

Here $\mathcal{N}$ corresponds to the number of wavevectors that are forced in Fourier space and N is the spatial resolution in the zonal and latitudinal directions. The dependence $\mathcal{A} \propto 1/\Delta t$ ensures that the forcing is delta-correlated in the limit $\gamma \to 1$ These expressions break down in the limit $\Delta t \to 0$ (corresponding to steady forcing), therefore we emphasise their applicability purely in the regimes of short decorrelation time $\tau)f = \Delta t/(1-\gamma)$, with $\gamma = 0$ used.

We implement idealised ring forcing numerically by forcing an annulus of wavevectors in

Fourier space centred around a mean radial wavenumber $k_f$ with thickness $2k = 2$. In order to avoid forcing purely zonal or purely meridional flows, we choose not to force wavevectors of the form $\mathbf{k} = (0, k)$ and $\mathbf{k} = (k, 0)$.

### A.3.3 Hyperviscosity

Hyperviscosity is used for maintaining numerical stability by acting as a sink for the build-up of energy in small scale fluctuations whilst largely unaffecting the larger scales. Hyperviscosity, $\nu_n \nabla^{2n} \zeta$, is incorporated with the aim of removing the build-up of enstrophy in the vicinity of the largest retained wavenumber, $k_{max}$. The resolution-dependent prefactor of this hyperviscosity term is implemented as

$$\nu_n = \frac{(-1)^{n+1}\nu}{k_{max}^{2n}}, \tag{A.42}$$

where $\nu$ is the amplitude and $\eta$ is the hyperviscosity index. Within this scheme, suitable choices for $\nu$ and $\eta$ must be made in order to fully define the numerical models. If $\eta$ is too small then significant damping at smaller wavenumbers will occur which will affect the numerical results. Conversely, if $\eta$ is too high then there will be virtually no damping for almost all the modes except for those closest to $k_{max}$ which will be damped significantly.

## A.4 Modelling Fluctuation Fields

### A.4.1 Quasi-Linear Beta Plane System

As written previously, by using a Reynolds Decomposition on the beta plane system (1.3) we obtain the following coupled system:

$$\left(\frac{\partial}{\partial t} - \frac{1}{Re}\nabla^2\right) U = -(u'v')_y \tag{A.43}$$

$$\mathcal{L}(U,k)u' = \xi(t) \tag{A.44}$$

Where $\mathcal{L}$ is a non-linear operator, or Linear operator for the quasi-linear (QL) case. The work in this section was conducted in parallel to the work in Section 2. If it proved too difficult a challenge to learn $u$, learning $u'$ implicitly, we would instead learn the form of the inverse of $\mathcal{L}$, to model $u' = \mathcal{G}(U, k, \xi(t))$ for the quasi-linear case and to obtain a model for the evolution of $u'$ and using this as a parameterisation that, when coupled with (3.1) yeilds a ROM of the system.

We first explored whether we were able to produce a reduced form of the quasi-linear approximation of $u$, via truncation in the zonal wavenumbers to both reduce the computation of the model, to more efficiently produce training examples for a Deep Learning model, as well as to have a fewer degrees-of-freedom representation of the system, potentially making the system more analytically tractable while reducing the number of parameters in a Neural Network attempting to model the field.

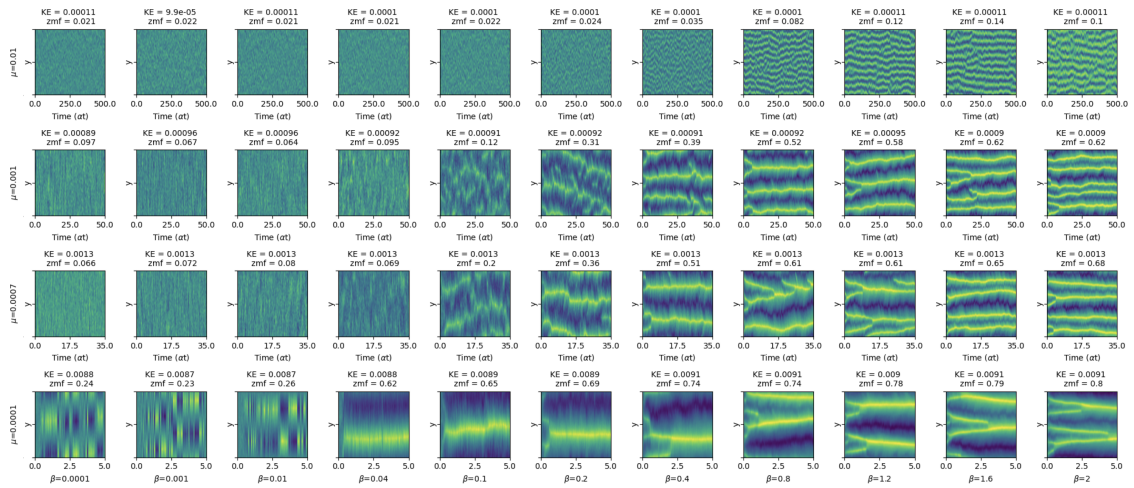


**Figure 12:** Latitude-time plots showing the full QL system forced via an annulus with principal wavenumber **k**=16. The plot displays differing behaviour over a range of damping rates $\mu$, decreasing from top to bottom and a range of $\beta$ values which increase from left to right. These were run for the case where $\varepsilon = 10^{-6}$. Values of total energy and zmf are printed above each plot.

Applying an eddy-eddy mean decomposition to the variables in (1.3) and taking a zonal average, the zonal mean vorticity equation is:

$$\frac{\partial \overline{\zeta}}{\partial t} + \frac{\partial}{\partial y}(\overline{v'\zeta'}) = -\mu\overline{\zeta} + \nu_n \frac{\partial^{2n}}{\partial^{2n} y}\overline{\zeta}, \tag{A.45}$$

while the eddy vorticity equation is:

$$\frac{\partial \overline{\zeta}}{\partial t} - \frac{\partial \overline{\psi}}{\partial y}\frac{\partial \zeta'}{\partial x} + \left(\beta + \frac{\partial \zeta'}{\partial y}\right)\frac{\partial \psi'}{\partial x} + \left[\frac{\partial \psi'}{\partial x}\frac{\partial \zeta'}{\partial y} - \frac{\partial \psi'}{\partial y}\frac{\partial \zeta'}{\partial x} - \frac{\partial}{\partial y}\left(\overline{\frac{\partial \psi'}{\partial x}\zeta'}\right)\right] = \xi - \mu\zeta' + \nu_n \nabla^{2n}\zeta'. \tag{A.46}$$

The QL model is obtained by removing the eddy-eddy interactions, achieved by defining terms in the square brackets in (3.4) to be zero, whilst retaining the coupling between the mean flow and the eddies.

This lack of scattering leads prevents cascades in the zonal wavenumbers, leading us to believe that a wavenumber truncation in this direction can be made, which will explored further in the next section. The validity for using a QL system has been demonstrated in studies of jet formation when the barotropic flow is stochastically forced [29] [74], as well as showing that such flows can reproduce jet emergence and transience [75]. In the absence of forcing and dissipation, the QL model respects the conservation of energy and enstrophy, however, it should be noted that it destroys the exact material conservation of potential vorticity. Both the NL and QL vorticity equations can be solved alongside the elliptic equation, $\zeta = \nabla^2\psi$ , subject to the prescription of the vorticity forcing $\xi$.

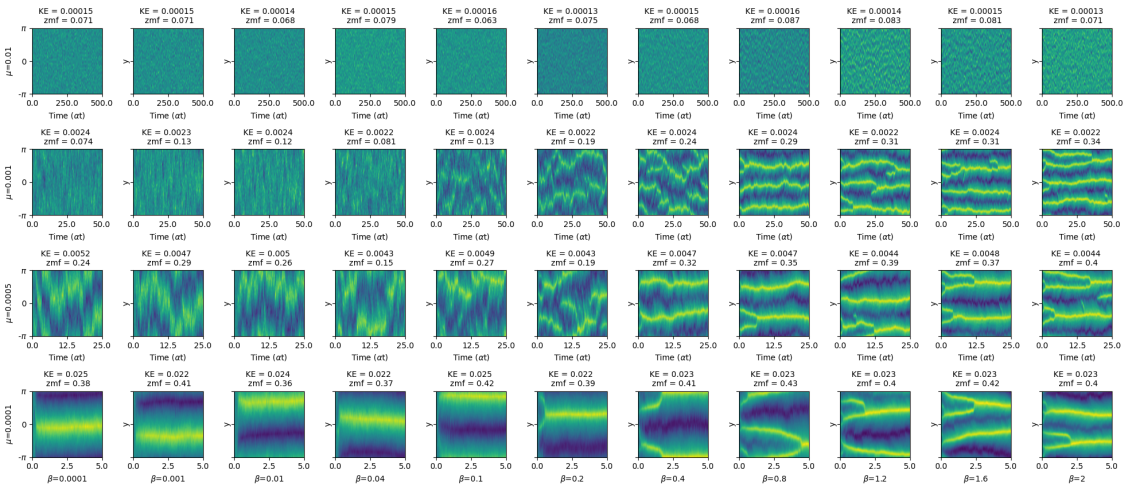


**Figure 13:** Latitude-time plots showing a truncated QL system, forced via a projection of an annulus with principal wavenumber **k**=16 onto a truncated number of wavenumbers in $x$, where the number of wavenumbers in $x$ is 4.The plot displays differing behaviour over a range of damping rates $\mu$, decreasing from top to bottom and a range of $\beta$ values which increase from left to right. These were run for the case where $\varepsilon = 10^{-6}$. Values of total energy and zmf are printed above each plot.

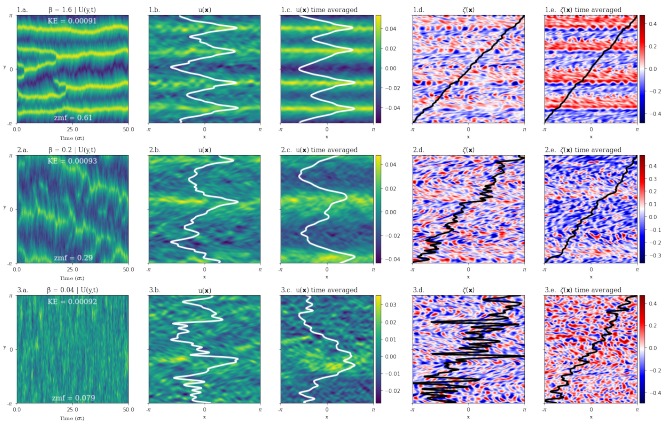

**Figure 14:** Zonostrophic flow regimes in the QL model in which $k_f = 16$, $N_y = N_x = 256$, $\mu = 0.01$ and $\varepsilon = 10^{-5}$. Plotted from left to right are:
a) a latitude-time plot of the zonal mean velocity, with KE and zmf values printed;
b) a snapshot of the zonal velocity field $u(\mathbf{x})$ at the final time step with the white line showing the instantaneous jet velocity profile $U(y)$;
c) the time-averaged zonal velocity field and jet velocity profile over the final 20 time steps as well as the colour bar that corresponds to plots a-c in each row
d) The corresponding vorticity profile at the final time step with the black line showing the zonal mean potential vorticity $\overline{\zeta + \beta y}$;
e) the time-averaged zonal velocity field and zonal mean potential vorticity over the final 20 as well as the colour bar that corresponds to plots d-e in each row
Top Row: 1) Zonostrophic regime $\beta = 1.6$, $R_\beta = 2.34$
Middle Row: 2) Transitional regime $\beta = 0.2$, $R_\beta = 1.9$
Bottom Row: 2) Frictional regime $\beta = 0.2$, $R_\beta = 1.62$.

### A.4.2 Zonal Wavenumber Truncation

This truncation reduces the number of grid points over which the integration (taking place over the same square, two-dimensional, doubly-periodic domain: $(x, y) \in [-\pi, \pi)^2$) is made. The full system uses N=256 grid points in $x$ and $y$, however, we here impose that the number of grid points in $x$, the zonal direction, is $N_x$ and retaining full resolution $y$ is $N_y = 256$ with $N_x \ll N_y = 256$. In wavenumber space, to avoid aliasing, we employ the 2/3 rule. This means that the number of grid points in $x$ must be at least 3/2 times larger than the specified number of wavenumbers in x. To force a single wavenumber, along with the corresponding negative wavenumber, we are therefore required to include 4 grid points in x ($Nx = 4 > \frac{3}{2}\ .\ 2$).

As the full QL system is forced by an annulus in spectral space, with a principal forcing wavenumber, $\mathbf{k} = 16$, we must naturally alter the nature of the forcing. The forcing is still a rapidly decorrelating, spatially homogeneous, stochastic forcing which is assumed to be a random function of both space and time, as described in section 3 and section 4.2. However, here energy is injected into the system using a different forcing structure, due to the reduced number of wavenumbers in x.

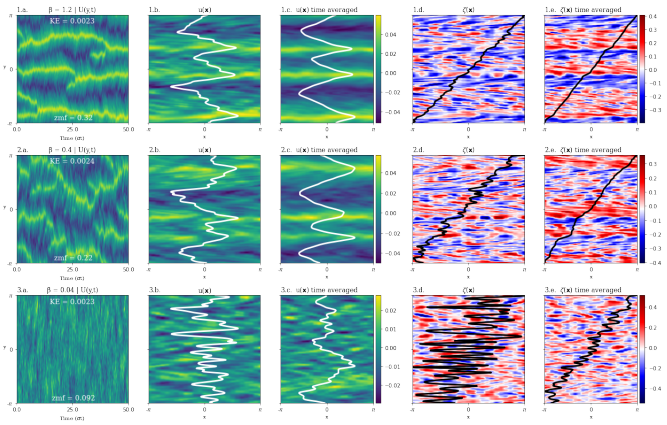

**Figure 15:** Set of plots similar to Figure 14, however for the case where the system is forced via a projection of an annulus with principal wavenumber **k**=16 onto a truncated number of wavenumbers in $x$, where the number of wavenumbers in $x$ is 4. Here where $\mu = 0.01$ and $\varepsilon = 10^{-5}$. Top Row: 1) $\beta = 1.2$, Middle Row: 2) $\beta = 0.4$, Bottom Row: 2) $\beta = 0.004$.

Initially, we forced all wavenumbers in $y$, up to the $k_f = 16$ of the annulus forcing, but only force a single wavenumber in $x$, however we found that this did not produce behaviour consistent with the full QL system, which can be seen in Figure 12. Alternatively, we looked at a forcing scheme whereby we had projected the shape of the annulus forcing of the original system onto our reduced set of wavenumbers in $x$. For a chosen value of $k_x = n$ (for $n \in N$, where $N$ is the chosen number of wavenumbers in $x$), a band of $k_y$ values are associated. These bottom of this band was determined by finding the midpoint between $k_{x=n}$ and $k_{x=n+1}$ and evaluating where the bottom of the annulus would lie for this value of $k_x$. Correspondingly, the top value of the band was determined by finding the midpoint between $k_{x=n}$ and $k_{x=n-1}$, and the top of the annulus for this value of $k_x$ was calculated.

In order to evaluate whether we were in the correct parameter regime to make comparisons to the QL system, we vary both $\mu$, the damping rate, and $\beta$ to observe the behaviour in this truncated scheme. This was chosen over varying values of $\varepsilon$, the energy injection rate, due to numerical instabilities that occurred when $\varepsilon$ was outside of a confined range. It must be noted that with the smaller values of $\mu$, the time for the system to equilibrate is greater.

On the whole, the behaviour, in Figure A.4.2, showing where the number of zonal wavenumbers in $x$ was reduced to 4, can be considered similar to that of the QL system, with the $\mu = 0.01$ set showing the same regimes of behaviour as $\beta$ varies. The exceptions to this truncated model displaying similar behaviour to the QL system can be seen at large $\mu$ and $\beta$, where in the QL system we saw the emergence of a large number of jets that are not apparent in this scheme. Similarly at small $\mu$ and $\beta$, the alternating sinusoidal zonally averaged velocity is not seen, instead, we see a single persistent jet - unexpected given

the small values of $\beta$, however, we did not look into these much, as the exploration of this reduced system was not the objective of the research. As $\beta$ increases we return the behaviour that is similar to that of the QL system, where the number of jets, as well as the zonostrophy of the system, increase - by the full velocity and vorticity fields for the 3 behaviour regimes we can compare properties to the full QL system. Each regime is categorised by the zonostrophy parameter, often the measure of the strength of the emergent zonal jet solutions:

$$R_\beta = \frac{\varepsilon^{1/20}\beta^{1/10}}{2^{1/2}\mu^{1/4}} \tag{A.47}$$

Figure 14, displaying latitude-time plots, full fields and time-averaged fields of velocity and vorticity for the full QL system and Figure 15, showing the same for the truncated QL system with 4 zonal wavenumbers both display Zonostrophic, Transitional and Frictional regimes. For this study it was most important to replicate the Zonostrophic regimes with this reduced system, with the merging and nucleating behaviour displayed in 15 and satisfactory to this truncated system.

The case where 2 wavenumbers in $x$ with the projected annulus in $k_y$, seen in Figure 16, displays very different behaviour to that of the full QL system, with no emergence of a Frictional regime, but more importantly a Zonostrophic regime at high $\beta$ in which undesired artefacts arise, making it an unsuitable ROM of the full system. Having found a reduced zonal wavenumber representation of the quasi-linear approximation of the beta plane system, where a minimum of 4 zonal wavenumbers were required to see behaviour like that of the full system, this corresponded to a computational speed-up of 4.3 times. The next steps would be to use a VAE and an adversarial VAE could be used to effectively find a finite-dimensional closure for the fluctuation fields, $u'$.

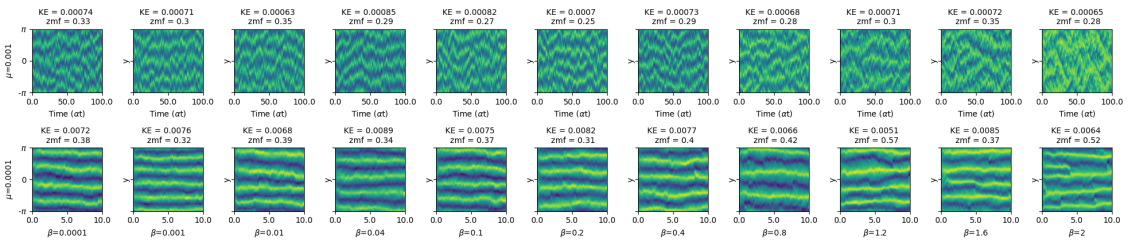


**Figure 16:** Latitude-time plots showing a truncated QL system, forced via a projection of an annulus with principal wavenumber **k**=16 onto a truncated number of wavenumbers in $x$, where the number of wavenumbers in $x$ is 2.The plot displays differing behaviour over a range of damping rates $\mu$, decreasing from top to bottom and a range of $\beta$ values which increase from left to right. These were run for the case where $\varepsilon = 10^{-5}$. Values of total energy and zmf are printed above each plot. The white plots have not been generated, while the discontinuity seen when $\beta = 4$ and $\mu = 0.0001$ is due to an overwriting error rather than any exhibited behaviour.