

## 773 A Macroscale drift parametrization

774 We parametrize the macroscale drift  $f_\theta(\zeta, \phi(\eta))$  using a static graph  $G = (V, A)$ , where  $V$  represents  
 775 the set of  $n_c$  macroscale grid points, and  $A \in \{0, 1\}^{n_c \times n_c}$  is the adjacency matrix. The macroscale  
 776 state  $\zeta \in \mathbb{R}^{n_\zeta}$  (where  $n_\zeta = n_c d_u$ , with  $d_u$  channels per grid point) can be viewed as the features  
 777 associated with the  $n_c$  nodes in  $G$ . The matrix  $A$  defines the local neighborhood of influence for each  
 778 point  $i \in V$ :  $A_{ij} = 1$  if point  $j$  influences point  $i$ , and  $A_{ij} = 0$  otherwise.

779 For the special case when  $d_u = 1$  (i.e., single-channel state) on one-dimensional spatial grids,  
 780  $\zeta \in \mathbb{R}^{n_c}$  (so  $n_\zeta = n_c$ ). The  $j$ -th component of the drift vector  $f_\theta(\zeta, \phi(\eta)) \in \mathbb{R}^{n_\zeta}$  is then given by a  
 781 localized function  $\hat{f}_\theta$ :

$$[f_\theta(\zeta, \phi(\eta))]_j = \hat{f}_\theta(\zeta_{j-q}, \dots, \zeta_{j+q}, \phi(\eta)) \text{ for } j = 1, \dots, n_\zeta, \quad (10)$$

782 where  $q \in \mathbb{N}$  defines the stencil half-width. Boundary conditions are handled as appropriate for the  
 783 specific test case.

784 It is worth noting that if the governing equations are known,  $f_\theta$  can be potentially derived using a  
 785 spatial discretization method, such as a finite difference (FD) scheme, on a coarse macroscale spatial  
 786 mesh. We do not adopt this approach for two primary reasons.

787 Firstly, even when an FD scheme is derived directly for the coarse grid spacing  $(\Delta x)_{\text{coarse}}$ , from the  
 788 original PDE, its accuracy is limited by truncation errors inherent to coarse discretizations. Standard  
 789 low-order FD stencils may provide a poor numerical approximation of the original PDE operator on  
 790 such a grid. Secondly, and more crucially, the effective governing dynamics on a coarse grid often  
 791 differ significantly in their functional form from the original PDE. The process of coarse-graining  
 792 means that unresolved subgrid-scale physics exert an influence on the resolved macroscale dynamics.  
 793 This influence can manifest as complex, state-dependent modifications to the original PDE terms or  
 794 even as entirely new effective terms (often referred to as closure terms or subgrid-scale parametriza-  
 795 tions). A simple FD discretization of the original PDE, regardless of the care taken in choosing stencil  
 796 coefficients for  $(\Delta x)_{\text{coarse}}$ , is generally incapable of representing these emergent subgrid-scale effects.  
 797 In contrast, by learning the drift function directly  
 798 from observed trajectories, our approach allows  
 799 the model to capture these effective coarse-grid  
 800 dynamics, including any implicit subgrid-scale  
 801 contributions, without being restricted to the  
 802 structure of the original PDE. This flexibility is  
 803 key to accurately modeling multiscale systems  
 804 on a computationally tractable coarse grid.

805 To demonstrate that our representation enables  
 806 interpretable models of macroscale dynamics  
 807 to be learned, we perform a numerical study  
 808 on the KdV test problem from Section 5 of  
 809 the main paper. Figure 9 presents a sensitiv-  
 810 ity study of the localized drift  $\hat{f}_\theta$  (from Eq. (10)  
 811 with  $q = 2$ ) for an implicit-scale model. This  
 812 model, corresponding to our full framework  
 813 with microscale dimension  $n_\eta = 0$  (see "Re-  
 814 marks on closure models and implicit-scale mod-  
 815 els" in Section 4 of the main paper), was trained  
 816 on the KdV dataset. For each subplot in Fig-  
 817 ure 9, corresponding to an input stencil loca-  
 818 tion  $k \in \{-2, \dots, 2\}$  (since  $q = 2$  here), the  
 819 macroscale state component  $\zeta_{j+k}$  is varied over  
 820  $[-1, 1]$ , while other  $\zeta$  components at the stencil  
 821 are held at zero. The resulting learned drift is  
 822 compared to the drift term derived from a stan-  
 823 dard FD approximation of the KdV equation  
 824 (details of the FD scheme are in Appendix G.1).  
 825 Since the learned macroscale state  $\zeta$  may have different magnitudes or units than the original physical

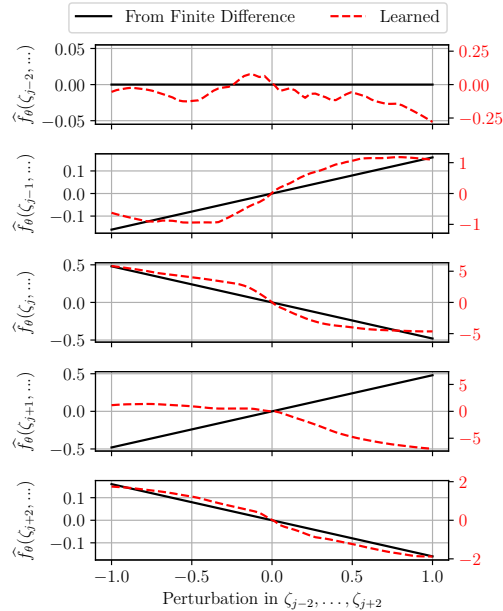


Figure 9: Sensitivity of the localized macroscale drift  $\hat{f}_\theta$  to perturbations of its stencil inputs  $\zeta_{j-2}, \dots, \zeta_{j+2}$ . The drift  $\hat{f}_\theta$  is from an implicit-scale model trained on the KdV dataset.

state due to the learned encoder, the outputs of both the learned drift and the FD-derived drift are rescaled to a common range in Figure 9 to enable a direct visual comparison.

It can be seen from Figure 9 that the drift derived using a FD scheme, not being optimized for the coarse grid, indeed differs from the learned drift function. The unresolved small-scale features introduce complex effects into the coarse-grid macroscale dynamics (such as apparent nonlinearities or memory effects) which a standard FD scheme cannot capture. These results motivate the present approach where the drift functions are learned directly from data.

## B Validity of Markovian latent dynamics

Our stochastic multiscale modeling framework models the latent dynamics  $z = (\zeta, \eta)$  with an Itô SDE, which is inherently a Markov process. However, when coarse-graining complex physical systems, the dynamics of the resolved variables often exhibit memory effects, rendering them non-Markovian. As mentioned in the Introduction, the Mori-Zwanzig formalism [12] provides a theoretical basis for understanding these memory effects in the context of closure modeling.

The Mori-Zwanzig formalism shows that the *exact* evolution of a set of resolved variables  $\bar{y}$  (analogous to our macroscale state  $\zeta$ ) interacting with unresolved variables  $\tilde{y}$  can be described by a generalized Langevin equation:

$$\frac{d}{dt}\bar{y}(t) = \underbrace{\mathcal{M}\bar{y}(t)}_{\text{Markov}} - \underbrace{\int_0^t K(s)\bar{y}(t-s)ds}_{\text{Memory}} + \underbrace{F(t)}_{\text{Noise}}, \quad (11)$$

where  $\mathcal{M}\bar{y}$  is a Markovian term that captures the resolved dynamics,  $F$  is a noise term arising from the unresolved dynamics associated with  $\tilde{y}$ , and the convolution integral involving the memory kernel  $K(s)$  captures the non-Markovian feedback from the history of  $\bar{y}(t)$  due to its coupling with  $\tilde{y}(t)$ .

To obtain a finite-dimensional Markovian representation suitable for learning SDE models, the memory integral in (11) must be approximated. One common approach (e.g., [40]) is to introduce a hierarchy of auxiliary variables  $w_0, w_1, \dots, w_m$  to represent the memory term and its derivatives. For instance, defining  $w_0(t) = \int_0^t K(s)\bar{y}(t-s)ds$ , the generalized Langevin equation becomes part of an extended system as shown below:

$$\frac{d\bar{y}(t)}{dt} = \mathcal{M}\bar{y}(t) - w_0(t) + F(t) \quad (12)$$

$$\frac{dw_0(t)}{dt} = K(0)\bar{y}(t) + \int_0^t K'(s)\bar{y}(t-s)ds, \quad (13)$$

where  $K'$  denotes the time derivative of the kernel. This procedure can be iterated by defining  $w_1(t)$  for the integral in (13) and accounting for the dynamics of  $w_1$ , which leads to

$$\begin{aligned} \frac{d\bar{y}(t)}{dt} &= \mathcal{M}\bar{y}(t) - w_0(t) + F(t) \\ \frac{dw_0(t)}{dt} &= K(0)\bar{y}(t) + w_1(t) \\ \frac{dw_1(t)}{dt} &= K'(0)\bar{y}(t) + \int_0^t K''(s)\bar{y}(t-s)ds, \end{aligned} \quad (14)$$

This iterative process generates a hierarchy of auxiliary variables  $w_0, w_1, \dots, w_m$  that represent the memory effects. The augmented state  $(\bar{y}, w_0, w_1, \dots, w_m)$  becomes approximately Markovian if this hierarchy is truncated at a level  $m$  where the remaining integral term is negligible. This truncation is an exact representation if the kernel is a polynomial of degree  $p-1$ , in which case the hierarchy terminates after  $p$  auxiliary variables.

In our multiscale modeling framework, we do not explicitly construct these Mori-Zwanzig auxiliary variables. Instead, we hypothesize that our learned latent SDE system for  $z = (\zeta, \eta)$  addresses the memory challenge in two ways. Firstly, the explicit microscale state  $\eta$  aims to capture some of the fast-evolving components that would otherwise contribute to the memory kernel  $K(t)$  in a model for  $\zeta$  alone. By conditioning the evolution of  $\zeta$  on  $\eta$  (and vice-versa) within a coupled Markovian

862 SDE, the effective memory in the  $\zeta$  dynamics may be significantly reduced. Secondly, the learned  
 863 drift and diffusion functions of our SDE, operating on the combined state  $z$ , are sufficiently flexible  
 864 to implicitly represent any residual short-term memory effects within the chosen finite-dimensional  
 865 latent space.

866 In summary, while the Markovian assumption for latent dynamics is an approximation for most  
 867 physical systems, it can be justified if the learned latent space  $z$  is structured and expressive enough  
 868 to capture the dominant interactions and short-term memory. The success of our model in practice  
 869 (see Section 5 of the main paper) lends empirical support to this assumption for the systems studied.

## 870 C Multiscale likelihood derivation

871 We summarize below the algebraic manipulations used to derive the multiscale log-likelihood in  
 872 Section 4.1 for the case of product of Gaussian experts. The Product of Experts (PoE) likelihood is  
 873 defined as:

$$p_\theta(y|\zeta, \eta) = \frac{1}{Z(\zeta, \eta)} p_1(y|\zeta) p_2(y - \mathcal{T}_p(\zeta) | \eta). \quad (15)$$

874 For the case of Gaussian experts, we have  $p_1(y|\zeta) = \mathcal{N}(y|\mathcal{T}_p(\zeta), \Sigma_\theta^\zeta)$  and  $p_2(y - \mathcal{T}_p(\zeta) | \eta) =$   
 875  $\mathcal{N}((y - \mathcal{T}_p(\zeta)) | \mathcal{D}_\theta(\eta), \Sigma_\theta^\eta)$ , i.e.,

$$\begin{aligned} p_1(y|\zeta) &= \frac{1}{Z_1} \exp\left(-\frac{1}{2}(y - \mathcal{T}_p(\zeta))^T (\Sigma_\theta^\zeta)^{-1} (y - \mathcal{T}_p(\zeta))\right), \\ p_2(y - \mathcal{T}_p(\zeta) | \eta) &= \frac{1}{Z_2} \exp\left(-\frac{1}{2}((y - \mathcal{T}_p(\zeta)) - \mathcal{D}_\theta(\eta))^T (\Sigma_\theta^\eta)^{-1} ((y - \mathcal{T}_p(\zeta)) - \mathcal{D}_\theta(\eta))\right). \end{aligned} \quad (16)$$

(17)

876 where  $Z_1 = (2\pi)^{n_y/2} |\Sigma_\theta^\zeta|^{1/2}$  and  $Z_2 = (2\pi)^{n_y/2} |\Sigma_\theta^\eta|^{1/2}$  are the normalization constants of the two  
 877 Gaussian PDFs.

878 We define the precision matrices as  $S_\theta^\zeta = (\Sigma_\theta^\zeta)^{-1}$  and  $S_\theta^\eta = (\Sigma_\theta^\eta)^{-1}$ . For compactness in the  
 879 subsequent derivation of the product, we introduce the temporary notational shorthands:  $\mu_1 = \mathcal{T}_p(\zeta)$   
 880 and  $\mu_2 = \mathcal{T}_p(\zeta) + \mathcal{D}_\theta(\eta)$ . Using these definitions, the product of the two Gaussian PDFs can be  
 881 written as

$$\begin{aligned} p_1(y|\zeta) \cdot p_2(y - \mathcal{T}_p(\zeta) | \eta) &= \frac{1}{K} \exp\left(-\frac{1}{2}(y - \mu_1)^T S_\theta^\zeta (y - \mu_1) - \frac{1}{2}(y - \mu_2)^T S_\theta^\eta (y - \mu_2)\right) \\ &= \frac{e^C}{K} \exp\left(-\frac{1}{2}(y - \mu_y)^T S_y (y - \mu_y)\right), \end{aligned} \quad (18)$$

882 where  $S_y = S_\theta^\zeta + S_\theta^\eta$ ,  $\mu_y = S_y^{-1}(S_\theta^\zeta \mu_1 + S_\theta^\eta \mu_2)$ ,  $C = \frac{1}{2} \mu_y^T S_y \mu_y - \frac{1}{2} (\mu_1^T S_\theta^\zeta \mu_1 + \mu_2^T S_\theta^\eta \mu_2)$ , and  
 883  $K = (2\pi)^{n_y} |\Sigma_\theta^\zeta|^{1/2} |\Sigma_\theta^\eta|^{1/2}$ .

884 The normalization constant of the product of Gaussians is given by

$$Z(\zeta, \eta) = \frac{e^C}{K} \int \exp\left(-\frac{1}{2}(y - \mu_y)^T S_y (y - \mu_y)\right) dy = \frac{e^C}{K} (2\pi)^{n_y/2} |S_y^{-1}|^{1/2}, \quad (19)$$

885 which yields the following expression for the normalized PoE likelihood

$$p_\theta(y|\zeta, \eta) = (2\pi)^{-n_y/2} |S_y^{-1}|^{-1/2} \exp\left(-\frac{1}{2}(y - \mu_y)^T S_y (y - \mu_y)\right). \quad (20)$$

886 Recalling our definitions  $\mu_1 = \mathcal{T}_p(\zeta)$ ,  $\mu_2 = \mathcal{T}_p(\zeta) + \mathcal{D}_\theta(\eta)$ , and using the resulting expression for  
 887  $\mu_y = S_y^{-1}(S_\theta^\zeta \mu_1 + S_\theta^\eta \mu_2)$ , the log-likelihood can be written as

$$\begin{aligned} \log p_\theta(y|\zeta, \eta) &= -\frac{1}{2}(y - \mu_y)^T S_y (y - \mu_y) - \frac{n_y}{2} \log(2\pi) + \frac{1}{2} \log |S_y| \\ &= -\frac{1}{2} \|r(\zeta)\|_{S_\theta^\zeta}^2 - \frac{1}{2} \|r(\zeta) - \mathcal{D}_\theta(\eta)\|_{S_\theta^\eta}^2 + \frac{1}{2} \|\mathcal{D}_\theta(\eta)\|_\Lambda^2 \\ &\quad - \frac{n_y}{2} \log(2\pi) + \frac{1}{2} \log |S_\theta^\zeta + S_\theta^\eta|, \end{aligned} \quad (21)$$

where  $\Lambda = S_\theta^\eta (S_\theta^\eta + S_\theta^\zeta)^{-1} S_\theta^\zeta$  and  $r(\zeta) = y - \mathcal{T}_p(\zeta)$  denotes the residual between the full observation and the macroscale model prediction.

To derive the expression for the log-likelihood in (6) which provides valuable insights into the regularization terms arising from the PoE likelihood, we first simplify the sum of the second and third terms in (21) as follows:

$$\begin{aligned} -\frac{1}{2} \|r(\zeta) - \mathcal{D}_\theta(\eta)\|_{S_\theta^\eta}^2 + \frac{1}{2} \|\mathcal{D}_\theta(\eta)\|_\Lambda^2 &= -\frac{1}{2} r(\zeta)^T S_\theta^\eta r(\zeta) + r(\zeta)^T S_\theta^\eta \mathcal{D}_\theta(\eta) \\ &\quad - \frac{1}{2} \mathcal{D}_\theta(\eta)^T S_\theta^\eta \mathcal{D}_\theta(\eta) + \frac{1}{2} \mathcal{D}_\theta(\eta)^T \Lambda \mathcal{D}_\theta(\eta) \\ &= -\frac{1}{2} r(\zeta)^T S_\theta^\eta r(\zeta) + r(\zeta)^T S_\theta^\eta \mathcal{D}_\theta(\eta) \\ &\quad - \frac{1}{2} \mathcal{D}_\theta(\eta)^T (S_\theta^\eta - \Lambda) \mathcal{D}_\theta(\eta). \end{aligned} \quad (22)$$

Noting that

$$\begin{aligned} \Lambda' &= S_\theta^\eta - \Lambda = S_\theta^\eta - S_\theta^\eta (S_\theta^\eta + S_\theta^\zeta)^{-1} S_\theta^\zeta = S_\theta^\eta \left[ I - (S_\theta^\eta + S_\theta^\zeta)^{-1} S_\theta^\zeta \right] \\ &= S_\theta^\eta \left[ (S_\theta^\eta + S_\theta^\zeta)^{-1} (S_\theta^\eta + S_\theta^\zeta) - (S_\theta^\eta + S_\theta^\zeta)^{-1} S_\theta^\zeta \right] = S_\theta^\eta (S_\theta^\eta + S_\theta^\zeta)^{-1} S_\theta^\eta \end{aligned} \quad (23)$$

is an SPD matrix, the last term in (22) can be written as  $-(1/2) \|\mathcal{D}_\theta(\eta)\|_{\Lambda'}^2$ , which is non-positive by definition. Substituting (22) into (21) and combining the first two terms yields the expression for the log-likelihood in (6) that we reproduce here for clarity:

$$\begin{aligned} \log p_\theta(y | \zeta, \eta) &= -\frac{1}{2} \|r(\zeta)\|_{(S_\theta^\zeta + S_\theta^\eta)}^2 + (r(\zeta), \mathcal{D}_\theta(\eta))_{S_\theta^\eta} - \frac{1}{2} \|\mathcal{D}_\theta(\eta)\|_{\Lambda'}^2 \\ &\quad - \frac{n_y}{2} \log(2\pi) + \frac{1}{2} \log |S_\theta^\zeta + S_\theta^\eta|. \end{aligned}$$

The term  $(1/2) \|\mathcal{D}_\theta(\eta)\|_{\Lambda'}^2$  implements an automatic scale separation mechanism by strongly penalizing microscale components in directions already well-represented by the macroscale model and allowing the microscale state more freedom in directions poorly captured by the macroscale model. This creates an “adaptive regularization” effect, wherein the regularization strength for each component of the microscale state is automatically calibrated based on the relative confidence in macroscale versus microscale predictions for that component, as encoded in the precision matrices  $S_\theta^\zeta$  and  $S_\theta^\eta$ . This is performed automatically based on the learned precision matrices without requiring manual specification of scale boundaries.

## D Multiple trajectory training

The SVI algorithm presented in Section 4 is formulated for a dataset of only one time series. In this section, we extend the algorithm to handle multiple time series with potentially different initial conditions, which we refer to as trajectories. We assume that the multiscale model parameters are shared across all time series. We denote the multiple-trajectory dataset as  $\mathcal{Y} = \{\mathcal{Y}_k\}_{k=1}^{n_{tr}}$ , where  $\mathcal{Y}_k$  is the  $k$ -th trajectory  $\{t_{ik}, y_{ik}\}_{i=1}^{n_{t,k}}$ , and  $n_{tr}$  is the number of trajectories. The  $i$ -th time step of the  $k$ -th trajectory is denoted  $t_{ik}$ , and the corresponding observation is  $y_{ik}$ . The multiple-trajectory ELBO is then given by simply including an extra summation in (8),

$$\begin{aligned} \mathcal{L}(\varphi) &= \sum_{k=1}^{n_{tr}} \sum_{i=1}^{n_{t,k}} \mathbb{E}_{z_{ik}, \theta} [\log p_\theta(y_{ik} | z_{ik})] - \frac{1}{2} \sum_{k=1}^{n_{tr}} \int_0^{T_k} \mathbb{E}_{z_k(t), \theta} \|r_{\theta, \varphi}(z_k(t), t)\|_{C(t)}^2 dt \\ &\quad - D_{KL}(q_\varphi(\theta) \| p(\theta)), \end{aligned} \quad (24)$$

where  $T_k$  is the final time of the  $k$ -th trajectory,  $\theta \sim q_\varphi(\theta)$ ,  $z_{ik} \sim q_\varphi(z_k | t_{ik})$ , and  $z_k(t) \sim q_\varphi(z_k | t)$ . The log-likelihood term is evaluated at each observation time  $t_{ik}$  of each trajectory  $k$  and summed over all observations. The drift residual term is evaluated over the entire time span of each trajectory.

## E Reparametrized, amortized ELBO

In this section, we outline how our multiscale SVI scheme, presented in Section 4 of the main paper, leverages the reparametrization trick and amortized SVI scheme proposed by Course and Nair [33, 42] for efficient training.

A key challenge in evaluating the ELBO defined in (8) of the main paper is that we require a SDE solver to generate sample trajectories of the linear variational SDE (7). The reparametrization trick, proposed in [33], allows us to circumvent the need for an SDE solver in the training loop. The core idea is that the marginal distribution of the variational linear SDE takes the form  $q_\varphi(z | t) = \mathcal{N}(\mu_\varphi(t), \Sigma_\varphi(t))$ , where the dynamics of  $\mu_\varphi$  and  $\Sigma_\varphi$  are governed by the following ODEs<sup>2</sup>

$$\begin{aligned}\dot{\mu}_\varphi(t) &= -A_\varphi(t)\mu_\varphi(t) + b_\varphi(t), \quad \mu_\varphi(0) = \mu_0, \\ \dot{\Sigma}_\varphi(t) &= -A_\varphi(t)\Sigma_\varphi(t) - \Sigma_\varphi(t)A_\varphi(t)^T + L_\theta(t)L_\theta(t)^T, \quad \Sigma_\varphi(0) = \Sigma_0.\end{aligned}\tag{25}$$

The reparametrization trick in [33] leverages (25) to express  $A_\varphi$  and  $b_\varphi$ , in terms of  $\mu_\varphi$ ,  $\Sigma_\varphi$  and their time derivatives. This, in turn, allows the drift residual to be rewritten as  $r_{\theta,\varphi}(z(t), t) = \dot{\mu}_\varphi(t) - B(t)(z(t) - \mu_\varphi(t)) - \gamma_\theta(z(t))$ , where  $B(t) = \mathcal{V}^{-1}((\Sigma_\varphi(t) \oplus \Sigma_\varphi(t))^{-1} \mathcal{V}(L_\theta(t)L_\theta(t)^T - \dot{\Sigma}_\varphi(t)))$ ,  $\oplus$  denotes the Kronecker sum, the operator  $\mathcal{V} : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n^2}$  stacks the columns of a matrix into a vector, and  $\mathcal{V}^{-1} : \mathbb{R}^{n^2} \rightarrow \mathbb{R}^{n \times n}$  does the opposite. Consequently, by parametrizing the variational distribution  $q_\varphi(z | t)$  directly via  $\mu_\varphi$  and  $\Sigma_\varphi$ , all the ELBO terms can be evaluated by drawing samples from the multivariate Gaussian  $\mathcal{N}(\mu_\varphi(t), \Sigma_\varphi(t))$ . This allows the ELBO to be maximized without an SDE solver in the training loop, thereby decoupling the computational cost of training from the stiffness of the variational SDE.

To address the challenge of learning the variational distribution  $q_\varphi(z | t)$  (i.e.,  $\mu_\varphi(t), \Sigma_\varphi(t)$ ) over long time horizons, we adopt the amortization strategy from [42]. This involves partitioning the time series into shorter segments and learning the variational distribution locally for each segment. To illustrate this approach, consider a trajectory with observation times  $\{t_1, \dots, t_m, t_{m+1}, \dots, t_{n_t}\}$ , where  $t_1 = 0$  and  $t_{n_t} = T$ . We partition this sequence into  $n_m = \lceil n_t/m \rceil$  non-overlapping segments (sub-intervals) as follows<sup>3</sup>

$$\left\{ \mathcal{I}_k = [t_1^{(k)}, t_2^{(k)}, \dots, t_{m_k}^{(k)}] \right\}_{k=1}^{n_m}, \tag{26}$$

where  $t_1^{(k)} = t_{m_{k-1}+1}^{(k-1)}$  (for  $k > 1$ ),  $m_k = m$  for  $k < n_m$ , and  $m_{n_m} \leq m$ . The variational distribution is then amortized over these  $n_m$  segments, leading to the reformulated ELBO:

$$\begin{aligned}\mathcal{L}(\varphi) &= \sum_{k=1}^{n_m} \left( \sum_{j=1}^{m_k} \mathbb{E}_{z_j^{(k)}, \theta} \left[ \log p_\theta \left( y_j^{(k)} | z_j^{(k)} \right) \right] - \frac{1}{2} \int_{t_1^{(k)}}^{t_{m_k}^{(k)}} \mathbb{E}_{z(t), \theta} \|r_{\theta,\varphi}(z(t), t)\|_{C(t)}^2 dt \right) \\ &\quad - D_{\text{KL}}(q_\varphi(\theta) \parallel p(\theta)),\end{aligned}\tag{27}$$

where  $\theta \sim q_\varphi(\theta)$ ,  $z_j^{(k)} \sim q_\varphi(z | t_j^{(k)})$ ,  $z(t) \sim q_\varphi(z | t)$ , and  $y_j^{(k)}$  is the  $j$ -th observation in the  $k$ -th segment, corresponding to time  $t_j^{(k)}$ .

Within each segment  $\mathcal{I}_k$ , the parameters of the variational distribution,  $\mu_\varphi$  and  $\Sigma_\varphi$ , are defined as follows: At the discrete observation times  $t_j^{(k)}$  within the segment,  $\mu_\varphi(t_j^{(k)})$  and  $\Sigma_\varphi(t_j^{(k)})$  are given by applying the probabilistic encoder (defined in Section 3) to the observation  $y_j^{(k)}$ . To obtain a continuous-time representation for  $\mu_\varphi$  and  $\Sigma_\varphi$  (and their derivatives  $\dot{\mu}_\varphi, \dot{\Sigma}_\varphi$ ) for  $t \in [t_1^{(k)}, t_{m_k}^{(k)}]$ , which is necessary for evaluating the integral in (27), we employ a deep kernel interpolation scheme that operates on these encoder outputs, following [42]. The trajectory partitioning parameter  $m \in \mathbb{N}$  is a hyperparameter of the training process.

<sup>2</sup> Recall that the initial conditions  $\mu_0, \Sigma_0$  are given by the probabilistic encoder  $p_\theta(z | y(0))$ , as discussed in Section 4.2 of the main paper.

<sup>3</sup> Note that when  $n_t$  is not divisible by  $m$ , the last interval has fewer than  $m$  elements.

## F Computing the posterior predictive distribution

After the model parameters are estimated using the multiscale variational inference framework, the primary objective during inference is to generate probabilistic predictions over a given time horizon. Specifically, given an initial observation of the full state  $y(t_0) \in \mathbb{R}^{n_y}$  at time  $t_0$ , we seek the *posterior predictive distribution*  $p_\theta(y(t) | y(t_0))$  for any  $t > t_0$ . This distribution quantifies our prediction for  $y(t)$  and accounts for all sources of uncertainty captured by the model, including the initial state decomposition, the stochastic evolution of the latent variables over the interval  $[t_0, t]$ , and the final state reconstruction. The prediction process involves three fundamental stages that bridge the observable state space ( $y$ ) and the latent multiscale state space  $z = [\zeta^T, \eta^T]^T \in \mathbb{R}^{n_z}$ .

**Stage 1: Probabilistic encoding of the initial state** The inference process commences by mapping the initial observation  $y(t_0)$  into the latent space defined by the macroscale state  $\zeta \in \mathbb{R}^{n_\zeta}$  and the microscale state  $\eta \in \mathbb{R}^{n_\eta}$ . Since our model posits a latent representation, the true initial latent state  $z(t_0)$  corresponding to  $y(t_0)$  is inherently uncertain. The learned probabilistic encoder  $p_\theta(z | y)$  provides the distribution over possible initial latent states at time  $t_0$ :

$$p_\theta(z(t_0) | y(t_0)) = \mathcal{N}(z(t_0) | \mu_{t_0}, \Sigma_{t_0}), \quad (28)$$

where  $\mu_{t_0} \in \mathbb{R}^{n_z}$  and  $\Sigma_{t_0} \in \mathbb{R}^{n_z \times n_z}$  are determined by the learned encoder applied to  $y(t_0)$ :

$$\mu_{t_0} = \mathbb{E}[z(t_0) | y(t_0)] = \begin{bmatrix} \mathbb{E}[\zeta(t_0) | y(t_0)] \\ \mathbb{E}[\eta(t_0) | y(t_0)] \end{bmatrix} = \begin{bmatrix} \mathcal{T}_r(\mathcal{S}_\theta(y(t_0))) \\ \mathcal{E}_\theta(\mathcal{S}_\theta^\perp(y(t_0))) \end{bmatrix}, \quad \Sigma_{t_0} = \text{Cov}(z(t_0) | y(t_0)) = \Sigma_\theta^z.$$

Here,  $\Sigma_\theta^z$  represents the learned covariance matrix associated with the encoder, capturing the uncertainty introduced by the initial scale separation ( $\mathcal{S}_\theta, \mathcal{T}_r$ ) and microscale compression ( $\mathcal{E}_\theta$ ). This initial distribution  $\mathcal{N}(z(t_0) | \mu_{t_0}, \Sigma_{t_0})$  serves as the starting point for evolving the latent dynamics over the time interval  $(t_0, t]$ .

**Stage 2: Evolving latent state uncertainty via the learned multiscale SDE** The core of the model lies in the learned coupled system of multiscale Itô SDEs governing the evolution of the latent macroscale ( $\zeta$ ) and microscale ( $\eta$ ) states, represented compactly as:

$$dz = \gamma_\theta(z(s))ds + L_\theta(s)d\beta(s), \quad s \in [t_0, t], \quad (29)$$

where  $\gamma_\theta(z) \in \mathbb{R}^{n_z}$  is the learned drift function,  $L_\theta(s) \in \mathbb{R}^{n_z \times (n_\zeta + n_\eta)}$  is the learned time-dependent diffusion matrix, and  $\beta \in \mathbb{R}^{n_\zeta + n_\eta}$  is a standard Wiener process. Propagating the initial distribution  $p_\theta(z(t_0) | y(t_0))$  through this SDE from time  $t_0$  to  $t$  yields the distribution of the latent state at the prediction time  $t$ , denoted by  $p_\theta(z(t) | y(t_0))$ . It is worth noting that even if the initial distribution  $p_\theta(z(t_0) | y(t_0))$  is Gaussian, the distribution  $p_\theta(z(t) | y(t_0))$  after evolution through the nonlinear SDE (29) is *non-Gaussian*. Its probability density function, formally governed by the Fokker-Planck equation associated with (29), rarely admits a closed-form analytical solution. We will show next how this challenge can be addressed using Monte Carlo sampling.

**Stage 3: Probabilistic reconstruction using the PoE likelihood** The final stage connects the distribution of the latent state at time  $t$ , i.e.,  $p_\theta(z(t) | y(t_0))$ , back to the observable state space  $y(t) \in \mathbb{R}^{n_y}$ . This mapping is defined by the likelihood model  $p_\theta(y | z)$  learned during training. As described in Section 4.1 and Appendix C, this likelihood leverages the PoE perspective to enforce scale separation. In brief, the combination of the macroscale expert  $p_1(y | \zeta) = \mathcal{N}(y | \mathcal{T}_p(\zeta), \Sigma_\theta^\zeta)$  and the microscale expert  $p_2(y - \mathcal{T}_p(\zeta) | \eta) = \mathcal{N}(y - \mathcal{T}_p(\zeta) | \mathcal{D}_\theta(\eta), \Sigma_\theta^\eta)$  results in a single Gaussian conditional distribution for  $y$  given a specific latent state  $z$ :

$$p_\theta(y | z) = \mathcal{N}(y | \mu_y(z), \Sigma_y), \quad (30)$$

where the conditional mean  $\mu_y(z) \in \mathbb{R}^{n_y}$  and conditional covariance  $\Sigma_y \in \mathbb{R}^{n_y \times n_y}$  are

$$\mu_y(z) = \mathbb{E}[y | z] = \mathcal{T}_p(\zeta) + \Sigma_y \mathcal{S}_\theta^\eta \mathcal{D}_\theta(\eta) \text{ and } \Sigma_y = \text{Cov}(y | z) = (\mathcal{S}_\theta^\zeta + \mathcal{S}_\theta^\eta)^{-1}. \quad (31)$$

Here,  $\mathcal{S}_\theta^\zeta = (\Sigma_\theta^\zeta)^{-1}$  and  $\mathcal{S}_\theta^\eta = (\Sigma_\theta^\eta)^{-1}$  are the precision matrices (parametrized by  $\theta$ ) associated with the Gaussian macroscale and microscale experts, respectively,  $\mathcal{T}_p : \mathbb{R}^{n_\zeta} \rightarrow \mathbb{R}^{n_y}$  is the prolongation operator, and  $\mathcal{D}_\theta : \mathbb{R}^{n_\eta} \rightarrow \mathbb{R}^{n_y}$  denotes the microscale decoder.

992 The PoE likelihood introduces two key statistical nuances. Firstly, the conditional mean  $\mu_y(z)$  in  
 993 (31) is *not* simply the sum of the prolonged macrostate  $\mathcal{T}_p(\zeta)$  and the decoded microstate  $\mathcal{D}_\theta(\eta)$ . The  
 994 contribution from the microscale decoder  $\mathcal{D}_\theta(\eta)$  is adaptively weighted by the matrix  $W = \Sigma_y S_\theta^\eta =$   
 995  $(S_\theta^\zeta + S_\theta^\eta)^{-1} S_\theta^\eta$ . This weighting factor emerges naturally from combining the Gaussian experts in  
 996 the PoE framework and depends on the learned relative precisions  $(S_\theta^\zeta, S_\theta^\eta)$ . This ensures that the  
 997 microscale state primarily contributes to explaining aspects of  $y$  where the macroscale expert has  
 998 low precision. Secondly, the conditional covariance  $\Sigma_y$  in (31), which is independent of  $z$ , quantifies  
 999 the inherent uncertainty in reconstructing  $y$  even if  $z$  were known perfectly. This uncertainty stems  
 1000 from the combination and potential disagreement between the macroscale and microscale experts, as  
 1001 encoded in their respective covariance matrices  $\Sigma_\theta^\zeta$  and  $\Sigma_\theta^\eta$ .

1002 The final predictive distribution  $p_\theta(y(t) | y(t_0))$  is obtained by marginalizing the intermediate latent  
 1003 state  $z(t)$ , which yields

$$\begin{aligned} p_\theta(y(t) | y(t_0)) &= \int p_\theta(y(t) | z(t)) p_\theta(z(t) | y(t_0)) dz(t) \\ &= \int \mathcal{N}(y(t) | \mu_y(z(t)), \Sigma_y) p_\theta(z(t) | y(t_0)) dz(t), \end{aligned} \quad (32)$$

1004 where in the second step we used the PoE likelihood (30). The preceding integral is the expecta-  
 1005 tion of the conditional likelihood  $p_\theta(y(t) | z(t))$  with respect to the distribution of the latent state  
 1006  $p_\theta(z(t) | y(t_0))$  obtained by propagating the initial condition provided at  $t_0$  through the latent SDE  
 1007 (29). Unfortunately, this integral is analytically intractable due to the nonlinearity of the conditional  
 1008 mean  $\mu_y(z(t))$  and the non-Gaussian nature of the distribution  $p_\theta(z(t) | y(t_0))$ . In the present work,  
 1009 we use Monte Carlo sampling to approximate this integral and characterize the posterior predictive  
 1010 distribution as follows:

- 1011 1. *Sample initial latent states:* Draw  $N > 1$  independent samples from the initial latent distribution  
 1012 defined in Eq. (28), i.e.,  $z_i(t_0) \sim p_\theta(z(t_0) | y(t_0)) = \mathcal{N}(\mu_{t_0}, \Sigma_{t_0})$ , for  $i = 1, \dots, N$ .
- 1013 2. *Propagate samples via SDE:* Numerically integrate the learned SDE (29) forward in time from  
 1014  $t_0$  to  $t$  for each initial sample  $z_i(t_0)$ . This yields  $N$  samples of the latent state  $\{z_i(t)\}_{i=1}^N$ , which  
 1015 collectively form an empirical approximation of the intractable distribution  $p_\theta(z(t) | y(t_0))$  at time  $t$ .
- 1016 3. *Calculate conditional means:* For each latent sample  $z_i(t)$ , compute the corresponding conditional  
 1017 mean of the observable state  $y(t)$  using (31), i.e.,  $\mu_{y,i} = \mu_y(z_i(t)) = \mathcal{T}_p(\zeta_i(t)) + \Sigma_y S_\theta^\eta \mathcal{D}_\theta(\eta_i(t))$ .
- 1018 4. *Approximate predictive distribution:* The posterior predictive distribution  $p_\theta(y(t) | y(t_0))$  is approx-  
 1019 imated by the empirical distribution derived from these samples. Specifically, it can be viewed as a  
 1020 Gaussian mixture model with  $N$  components:  $p_\theta(y(t) | y(t_0)) \approx \frac{1}{N} \sum_{i=1}^N \mathcal{N}(y(t) | \mu_{y,i}, \Sigma_y)$ , where  
 1021 each component  $\mathcal{N}(y(t) | \mu_{y,i}, \Sigma_y)$  in this mixture represents the distribution of  $y(t)$  conditioned on  
 1022 a specific realization  $z_i(t)$  of the latent state evolution. All components share the same conditional  
 1023 covariance  $\Sigma_y$  (the reconstruction uncertainty from the PoE), but are centered at potentially different  
 1024 means  $\mu_{y,i}$  reflecting the uncertainty propagated through the latent dynamics.

1025 **Predictive mean and covariance** The overall moments of the posterior predictive distribution can  
 1026 be estimated from the Gaussian mixture approximation. The predictive mean is the expected value of  
 1027  $y(t)$  given the initial observation  $y(t_0)$ , which can be estimated by taking the sample average of the  
 1028 conditional means  $\mu_{y,i}$  obtained from each propagated latent state sample  $z_i(t)$ , i.e.,

$$\hat{\mu}_{pred}(t) = \mathbb{E}[y(t) | y(t_0)] \approx \frac{1}{N} \sum_{i=1}^N \mathbb{E}[y(t) | z_i(t)] = \frac{1}{N} \sum_{i=1}^N \mu_{y,i}. \quad (33)$$

1029 The predictive covariance which is the total covariance matrix of  $y(t)$  given  $y(t_0)$ , capturing the  
 1030 overall prediction uncertainty at time  $t$  can similarly be estimated as:

$$\begin{aligned} \hat{\Sigma}_{pred}(t) &= \text{Cov}(y(t) | y(t_0)) = \mathbb{E}_{z(t)}[\text{Cov}(y(t) | z(t))] + \text{Cov}_{z(t)}[\mathbb{E}(y(t) | z(t))] \\ &\approx \frac{1}{N} \sum_{i=1}^N \Sigma_y + \frac{1}{N-1} \sum_{i=1}^N (\mu_{y,i} - \hat{\mu}_{pred}(t))(\mu_{y,i} - \hat{\mu}_{pred}(t))^T. \\ &= \Sigma_y + \text{SampleCov}(\{\mu_{y,i}\}_{i=1}^N). \end{aligned} \quad (34)$$

The total predictive covariance is given by the sum of two distinct terms: (i) the average *reconstruction uncertainty*,  $\Sigma_y = (S_\theta^\zeta + S_\theta^\eta)^{-1}$ , which is independent of the latent state propagation reflecting the precision of the combined experts, and (ii) the *propagated uncertainty*,  $\text{SampleCov}(\{\mu_{y,i}\})$ , which arises from the variability of the latent state samples  $\{z_i(t)\}$  incorporating both the initial uncertainty encoded in  $\Sigma_{t_0}$  and the stochasticity introduced by the SDE dynamics over  $(t_0, t]$ . The proposed Monte Carlo sampling procedure provides a practical and statistically principled method for generating predictions and quantifying their uncertainty at any future time  $t > t_0$ . It fully leverages the structure of the learned stochastic multiscale model and the scale-separating PoE likelihood, without requiring further analytical approximations beyond the finite number of samples  $N$ .

## G Detailed setups for test cases

In this section, we provide detailed setups for the test cases, as well as their respective multiscale model architectures and training procedures.

### G.1 Korteweg-de Vries equation

This section provides the detailed setup for the Korteweg-de Vries (KdV) equation test case. So that this section is self-contained, some information is repeated from Section 5.

The Korteweg-de Vries (KdV) equation in 1D is given by

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \nu \frac{\partial^3 u}{\partial x^3} = 0, \quad (35)$$

where  $\nu = 0.02$  and the domain is  $x \in [0, 10]$  with periodic boundary conditions. We discretize the state in space with a spatial step  $\Delta x$  of 0.01, so the full state dimension is  $n_y = 1000$ . We then approximate the spatial derivatives with finite-difference schemes, obtaining

$$\frac{dy_j}{dt} = -\frac{y_{j+1} - y_{j-1}}{2\Delta x} y_j - \nu \frac{y_{j+2} - 3y_{j+1} + 3y_j - y_{j-1}}{\Delta x^3}, \quad (36)$$

for  $j = 1, \dots, n_y$ , where  $y_j$  denotes the state at the  $j$ -th spatial gridpoint. The finite-difference weights are the textbook results for the given uniformly-spaced gridpoints.<sup>4</sup> The initial condition is  $u(x, 0) = a \cos(\pi x) \exp\left(-\frac{(x-7.5)^2}{s^2}\right)$ , where  $a \sim \mathcal{N}(2, 0.01)$  and  $s \sim \mathcal{N}(1, 0.01)$ , sampled on the spatial grid to obtain  $y(0)$ . The simulation is run for  $t \in [0, 1]$  with a time step  $\Delta t$  of 0.001, giving 1001 time steps. The observations are corrupted by Gaussian noise with standard deviation 0.01. The training dataset is formed using 10 samples of initial conditions and their corresponding trajectories, and the validation and test sets are formed using 5 samples each. Our macroscale state is formed on a coarse  $n_\zeta = 20$ -point mesh.

### G.2 Burgers' equation in 2D

This section provides the detailed setup for the 2D Burgers' equation test case. So that this section is self-contained, some information is repeated from Section 5.

The viscous Burgers' equation in 2D is given by

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = \nu \nabla^2 u, \quad (37)$$

where  $\nu$  is the viscosity. We consider the case where the viscosity is  $\nu = 0.005$  and the domain is  $x \in [0, 1]$  with a zero Dirichlet boundary condition. We discretize the state in space with a spatial step  $\Delta x$  of  $1/127$ , so the full state dimension is  $n_y = 128^2 = 16384$ . We then approximate the spatial derivatives with finite-difference schemes, obtaining

$$\begin{aligned} \frac{dy_{i,j}}{dt} = & -\left(\frac{y_{i+1,j} - y_{i-1,j}}{2\Delta x} + \frac{y_{i,j+1} - y_{i,j-1}}{2\Delta x}\right) y_{i,j} \\ & + \nu \left(\frac{y_{i+1,j} - 2y_{i,j} + y_{i-1,j}}{\Delta x^2} + \frac{y_{i,j+1} - 2y_{i,j} + y_{i,j-1}}{\Delta x^2}\right), \end{aligned} \quad (38)$$

<sup>4</sup> Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51:699–706, 1988



for  $i = 1, \dots, 128$ ,  $j = 1, \dots, 128$ , where  $y_{i,j}$  denotes the state at the  $i, j$ -th spatial gridpoint. The initial condition is a Gaussian curve  $u(x_1, x_2, 0) = a \exp\left(-\frac{(x_1-0.3)^2 + (x_2-0.3)^2}{s^2}\right)$ , where  $a \sim \mathcal{N}(1, 0.01)$  and  $s \sim \mathcal{N}(0.2, 0.0001)$ , sampled on the spatial grid to obtain  $y(0)$ . The simulation is run for  $t \in [0, 1]$  with a time step  $\Delta t$  of 0.001, giving  $n_t = 1001$  time steps. The observations are corrupted by Gaussian noise with standard deviation 0.001. The training dataset is formed using 20 samples of initial conditions and their corresponding trajectories, and the validation and test sets are formed using 5 samples each. Our macroscale state is on a coarse  $n_\zeta = 8 \times 8$  grid.

### G.3 Cylinder flow in 2D

This section provides the detailed setup for the 2D cylinder flow test case. So that this section is self-contained, some information is repeated from Section 5.

This example is a 2D Von Kármán vortex street at  $Re = 160$  from Günther et al. [34] produced using the Gerris flow solver [35]. The dataset consists of a nondimensionalized 2D velocity field of order 1. The spatial domain of the provided dataset is  $[-0.5, 7.5] \times [-0.5, 0.5]$ , sampled on a uniform  $640 \times 80$  grid. For our analysis, the domain is truncated to  $[-0.5, 3.5] \times [-0.5, 0.5]$  on a  $D = 320 \times 80$  grid. The time domain is  $[0, 15]$ , divided into 1501 time steps and developing from a zero initial condition.

Because this dataset is a single trajectory, we partition the time interval. The training set consists of the interval  $[0, 13]$ , the validation set consists of  $(13, 14]$ , and the test set consists of  $(14, 15]$ . No noise is added to the data. Our macroscale state is on a coarse  $32 \times 8$  grid, giving it a dimension of  $n_\zeta = 2 \times 32 \times 8 = 512$ .

### G.4 Neural network architectures

The relevant quantities from the problem setup are the fine grid dimension  $n_f$ , the coarse grid dimension  $n_c$ , and the spatial domain dimension  $d$ . We assume that  $\sigma = (n_f/n_c)^{1/d}$  (or equivalently,  $\sigma = (n_y/n_z)^{1/d}$ ) is an integer. The neural network architectures for the macroscale state encoder/decoder, the microscale state encoder/decoder, and the drift functions are as follows.

**Encoder** The encoder is a variational encoder, so given a full state  $y$ , it returns a mean  $\mu_\theta^z$  and covariance  $\Sigma_\theta^z$  that represent a latent Gaussian distribution over the concatenated macroscale state and microscale state.

The macroscale mean is computed with a single-channel convolutional layer with stride  $\sigma$  and kernel size  $6\sigma + 1$  for KdV,  $(6\sigma + 1)^2$  for Burgers 2D, and  $(4\sigma + 1)^2$  for cylinder flow. Circular padding of length  $3\sigma$  is used for KdV,  $3\sigma$  for Burgers 2D, and  $2\sigma$  for cylinder flow. The output of the encoder is a single-channel state on a coarse grid with  $n_c$  points. This corresponds to the composition of the restriction  $\mathcal{T}_r$  and smoothing  $\mathcal{S}_\theta$ .

Table 2: Architecture of the microscale state encoder for each test case.

Korteweg-de Vries 1D			Burgers 2D			Cylinder flow 2D		
Convolutional layers			Convolutional layers			Convolutional layers		
Layer	Number of filters		Layer	Number of filters		Layer	Number of filters	
1	1		1	1		1	1	
2	4		2	16		2	16	
3	16		3	32		3	32	
4	64		4	64		4	64	
Linear layers			Linear layers			Linear layers		
Layer	In dim.	Out dim.	Layer	In dim.	Out dim.	Layer	In dim.	Out dim.
5	8000	$n_\eta$	5	256	$n_\eta$	5	640	$n_\eta$

The microscale mean is computed with a more sophisticated architecture. The convolutional operator that yields the macroscale mean shares its kernel with a similar operator that has a stride of 1. This represents the smoothing operator  $\mathcal{S}_\theta$  without restriction. This is applied to the input  $y$  to obtain  $\bar{y}$ , which is then subtracted from  $y$  to obtain the residual  $\tilde{y}$ . The residual is then encoded to obtain the microscale state  $\eta$  via the encoder  $\mathcal{E}_\theta$ . Its architecture is described in Table 2 for the three test cases. The activation between each layer is a ReLU function. For all convolution layers in the KdV case, the kernel size is 25, the stride is 2, and circular padding of length 12 is used. For all convolution layers in the Burgers 2D and cylinder flow cases, the kernel size is  $9 \times 9$ , the stride is 4, and circular padding of length 4 is used.

The concatenated (block-diagonal) covariance  $\Sigma_\theta^z$  is assumed diagonal and constant with respect to the input  $y$ . It is directly parametrized.

**Decoder** The decoder is a variational decoder, so given a latent state  $z$ , it returns a mean  $\mu_\theta^y$  and covariance  $\Sigma_\theta^y$  that represent a Gaussian distribution over the full state.

The mean is a sum of two terms, as shown in (31). The first term is the prolonged macroscale state  $\mathcal{T}_p(\zeta)$ , computed with a single-channel deconvolutional layer, whose parameters are shared with the macroscale encoder, thereby approximately inverting it. The second term depends on the decoded microscale state  $\mathcal{D}_\theta(\eta)$ , which has the architecture shown in Table 3. As with the microscale encoder, the activation between each layer is a ReLU function. For all deconvolution layers in the KdV case, the kernel size is 25, the stride is 2, and circular padding of length 12 is used. For all deconvolution layers in the Burgers 2D and cylinder flow cases, the kernel size is  $9 \times 9$ , the stride is 4, and circular padding of length 4 is used.

Table 3: Architecture of the microscale state decoder for each test case.

Korteweg-de Vries 1D			Burgers 2D			Cylinder flow 2D		
Linear layers			Linear layers			Linear layers		
Layer	In dim.	Out dim.	Layer	In dim.	Out dim.	Layer	In dim.	Out dim.
1	$n_\eta$	8000	1	$n_\eta$	256	1	$n_\eta$	640
Deconvolutional layers			Deconvolutional layers			Deconvolutional layers		
Layer	Number of filters		Layer	Number of filters		Layer	Number of filters	
2	64		2	64		2	64	
3	16		3	32		3	32	
4	4		4	16		4	16	
5	1		5	1		5	1	

As with the encoder,  $\Sigma_\theta^y$  is diagonal and directly parametrized, i.e., constant with respect to  $z$ .

**Macroscale drift** The macroscale drift function is a graph-structured neural network as described in Appendix A. The elementwise drift function  $\hat{f}_\theta$  in (10) is a feedforward neural network with input dimension  $d_u(2q+1)^d + n_\eta^*$  (recall that  $d_u$  is the number of state channels). For all test cases, we choose  $q = 2$ . The architecture of the drift function is shown in Table 4. The activation between each layer is a ReLU function.

In the KdV and Burgers 2D test cases, we do not leverage  $\phi$  to restrict the coupling in the latent SDE. We set it to identity, so  $n_\eta^* = n_\eta$ . In the cylinder flow case however, we use  $\phi$  to encode temporal and spatial information that influences the macroscale dynamics. This is because unlike the other test cases, the presence of the cylinder in the flow, modelled as a region of zero velocity, means that the macroscale dynamics are different inside and outside the region. Because the output of  $\phi$  passes to an elementwise drift function  $\hat{f}_\theta$ , there are corresponding coordinates  $x_1, x_2$  for the macroscale element. We define  $\phi = \eta + \phi^*(\eta, \cos(t), \sin(t), x_1, x_2)$ , where time  $t$  is passed through an encoding, and where  $\phi^*$  is a neural network shown in Table 5.

Table 4: Architecture of the macroscale drift function for each test case.

Korteweg-de Vries 1D			Burgers 2D			Cylinder flow 2D		
Linear layers			Linear layers			Linear layers		
Layer	In dim.	Out dim.	Layer	In dim.	Out dim.	Layer	In dim.	Out dim.
1	$5 + n_\eta^*$	128	1	$25 + n_\eta^*$	128	1	$50 + n_\eta^*$	128
2	128	128	2	128	512	2	128	512
3	128	1	3	512	128	3	512	128
			4	128	1	4	128	2

Table 5: Architecture of the coupling activation term for the macroscale drift in cylinder flow case.

Linear layers		
Layer	In dim.	Out dim.
1	$4 + n_\eta$	128
2	128	128
3	128	$n_\eta$

1134 **Microscale drift** Because the microscale state has no spatial grid structure like the macroscale  
1135 state, the microscale drift function is a simple fully-connected neural network. The input dimension  
1136 is  $n_\eta + n_\zeta^* + 2$ , where the  $+2$  is due to the fact that this drift is non-autonomous; time is an input of  
1137 the drift via the encoding  $\sin(t), \cos(t)$ . The architecture is shown in Table 6 for each test case. The  
1138 activation between each layer is a ReLU function. For the KdV case, the coupling term  $\psi$  is identity,  
1139 so  $n_\zeta^* = n_\zeta$ . For the Burgers 2D and cylinder flow cases,  $\psi$  is a linear map that projects the  $n_\zeta$ -dim.  
1140 macroscale state to  $n_\eta$ -dim., so  $n_\zeta^* = n_\eta$ .

Table 6: Architecture of the microscale drift function for each test case.

Korteweg-de Vries 1D			Burgers 2D			Cylinder flow 2D		
Linear layers			Linear layers			Linear layers		
Layer	In dim.	Out dim.	Layer	In dim.	Out dim.	Layer	In dim.	Out dim.
1	$n_\eta + n_\zeta^* + 2$	128	1	$2n_\eta + 2$	512	1	$2n_\eta + 2$	128
2	128	512	2	512	256	2	128	256
3	512	128	3	256	128	3	256	128
4	128	$n_\eta$	4	128	$n_\eta$	4	128	$n_\eta$

## 1141 G.5 Training

1142 Each model is trained with the Adam optimizer<sup>5</sup> and the loss function is the negative of the ELBO  
1143 from (8). The batch size is 128. Because this batch corresponds to samples from a time series  
1144 covering  $[0, T]$ , it covers a time interval of  $T \times 127/(n_t - 1)$ . The expected values are evaluated by  
1145 sampling from the variational distributions. One sample of  $z$  is drawn to evaluate the expectations.  
1146 To approximate the integral in the drift residual, we use 128 quadrature samples of  $t$ , matching the  
1147 batch size. The KdV models were trained for 200 epochs, Burgers 2D for 100 epochs, and cylinder  
1148 flow for 2000 epochs. This roughly equalized the number of optimization steps for each (16-22,000).

<sup>5</sup> Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations*, 2015

Each multiscale model is trained in a hierarchical manner. We begin by training an implicit-scale model with an initial learning rate of  $10^{-3}$ . We apply an exponential scheduler that decays the learning rate by 10% every 2000 optimization steps. We then initialize the multiscale model with  $n_\eta = 1$  using the trained parameters from the implicit-scale model; we apply Xavier normal initialization to all new parameters. The multiscale model is then trained with a reduced initial learning rate of  $10^{-4}$ , as its loss landscape is more sensitive to large step sizes than that of the implicit-scale model. For a multiscale model with  $n_\eta = 2$ , we repeat this process using the parameters from the trained model with  $n_\eta = 1$ , and so on for larger  $n_\eta$ .

We note that this hierarchical approach is not necessary to train a multiscale model; it is possible to train a model with large  $n_\eta$  in one shot using an initial learning rate of  $10^{-3}$ . The issue however is that the loss landscape is highly non-convex, so the training process may stall at a suboptimal local minimum. As a result, we would not obtain the approximately-monotonic reduction in error with increasing  $n_\eta$  in Table 1. The hierarchical initialization serves to stabilize the training process.

## H Additional plots for results

This section contains expanded plots of the results presented in Section 5.

Figures 10, 12, and 13/14 expand on Figures 4, 6, and 8 respectively. They show the scale separation of the multiscale framework applied to each of our three test problems, but they also show how the macroscale and microscale states evolve over time. Figures 13 and 14 show the  $x$  and  $y$  components of the velocity field respectively for the cylinder flow case.

Figures 11 and 17/18 complement Figures 5 and 7 respectively. For the KdV prediction plots in Figure 3, we were able to show the standard deviation bounds in the predictions, however this was not possible easily with the 2D test cases. Here, for the Burgers 2D and cylinder flow cases, we provide plots of the prediction standard deviation and the absolute error between the prediction mean and the observation.

Figures 15 and 16 compare the implicit-scale and multiscale model predictions. Unlike Figure 7, these show the entire spatial domain and both the  $x$  and  $y$  velocity components.

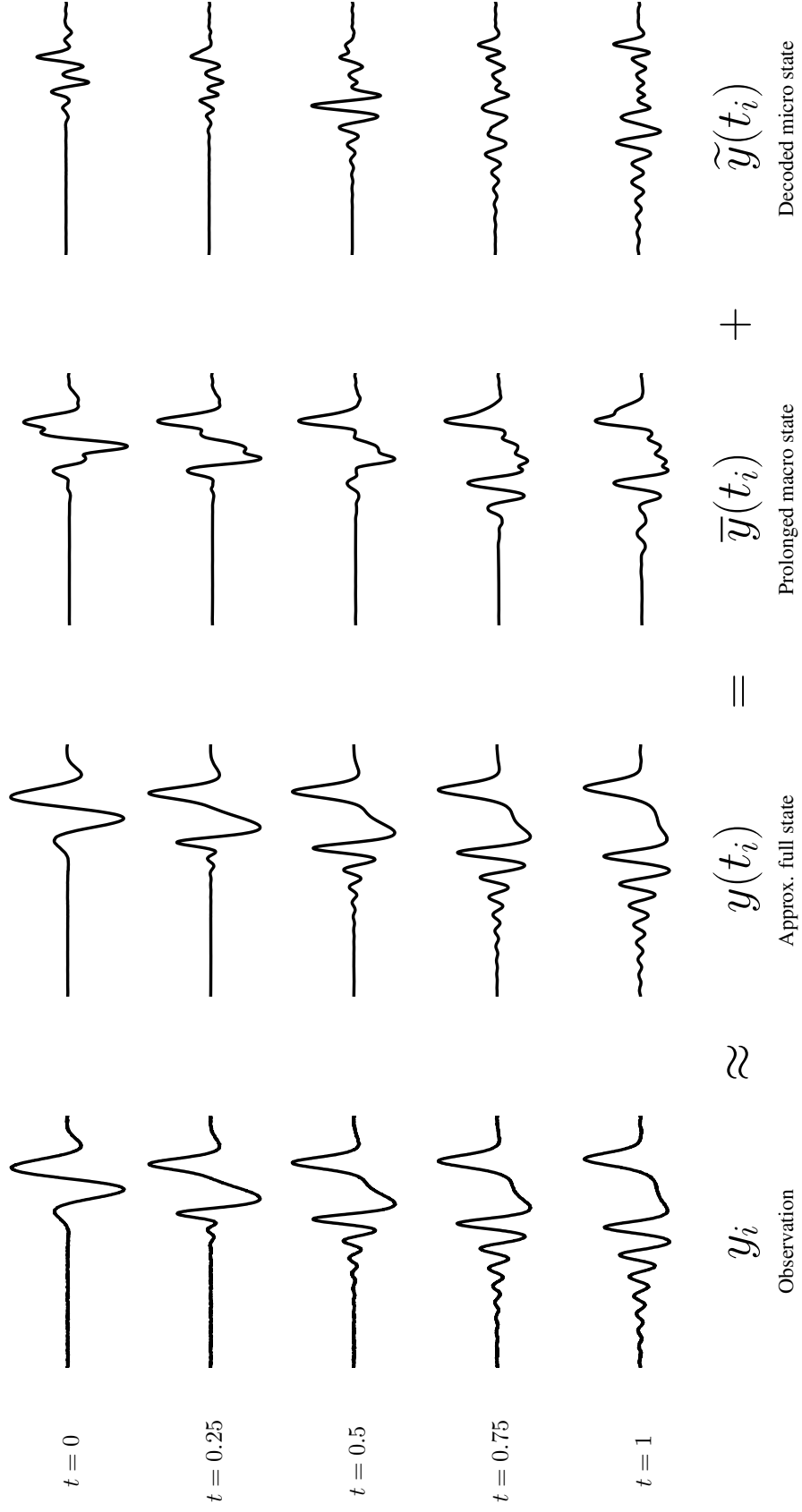


Figure 10: KdV equation: scale separation with  $n_\zeta = 20$  and  $n_\eta = 5$ . All plots on common  $x$  and  $y$ -axis scales.

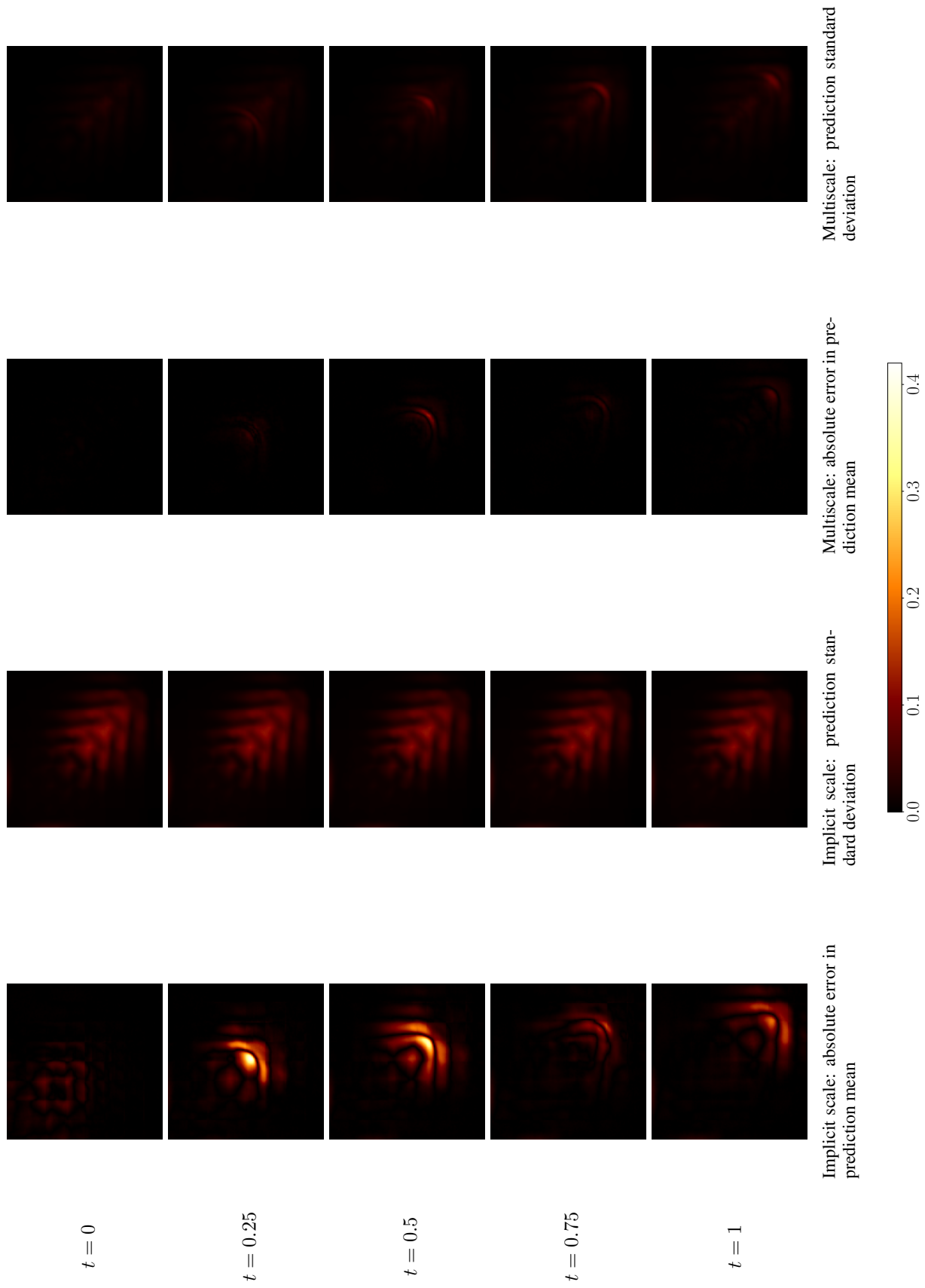


Figure 11: Burgers' equation in 2D: model comparison in spatial error and prediction standard deviation with  $n_\zeta = 8 \times 8 = 64$  and  $n_\eta = 5$ . All plots on common  $z$ -axis scale.

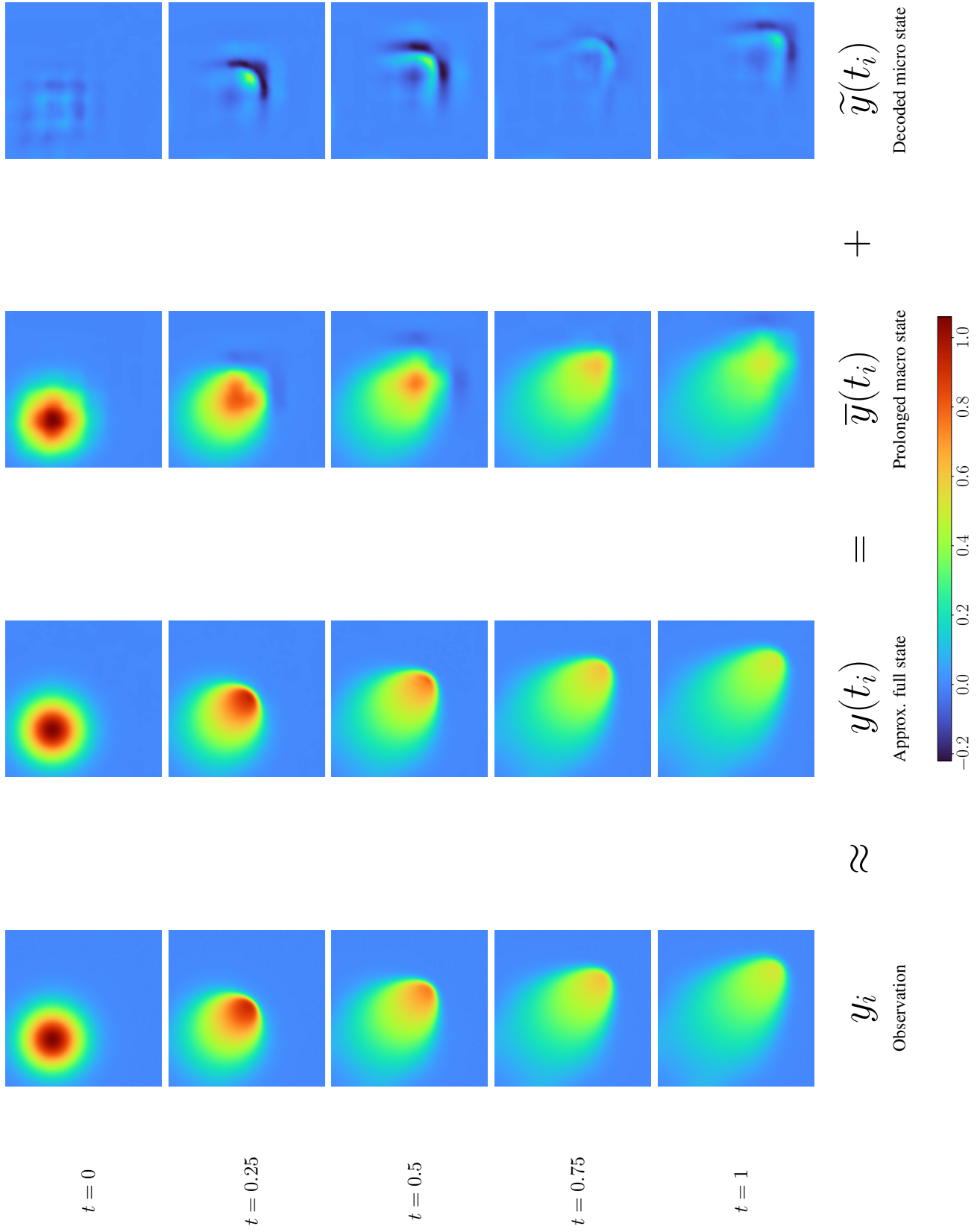


Figure 12: Burgers' equation in 2D: scale separation with  $n_\zeta = 8 \times 8 = 64$  and  $n_\eta = 5$ . All plots on common  $z$ -axis scale.

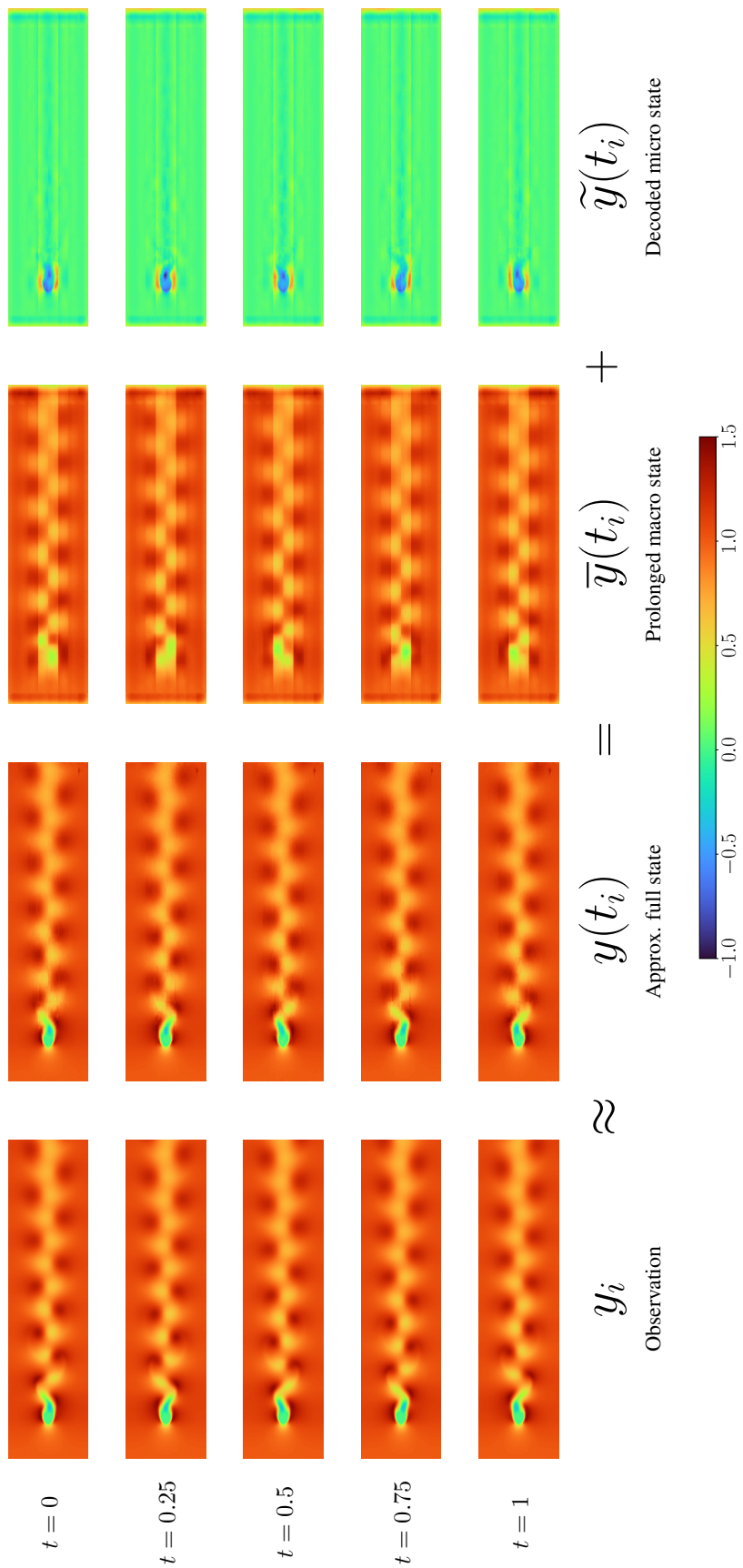


Figure 13: Cylinder flow in 2D: scale separation on velocity  $x$ -component with  $n_\zeta = 2 \times 32 \times 8 = 512$  and  $n_\eta = 5$ . All plots on common  $z$ -axis scale.



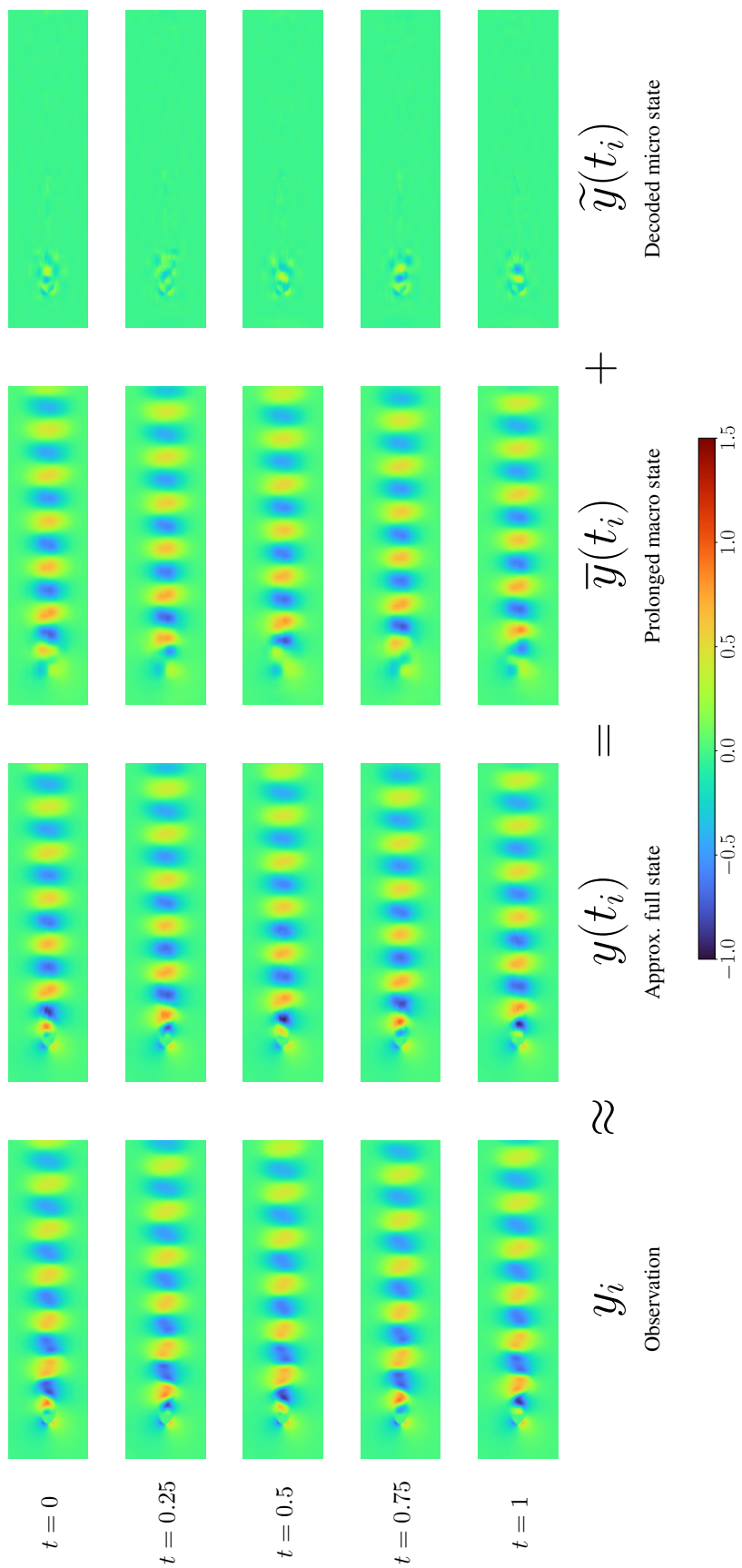


Figure 14: Cylinder flow in 2D: scale separation on velocity  $y$ -component with  $n_\zeta = 2 \times 32 \times 8 = 512$  and  $n_\eta = 5$ . All plots on common  $z$ -axis scale.

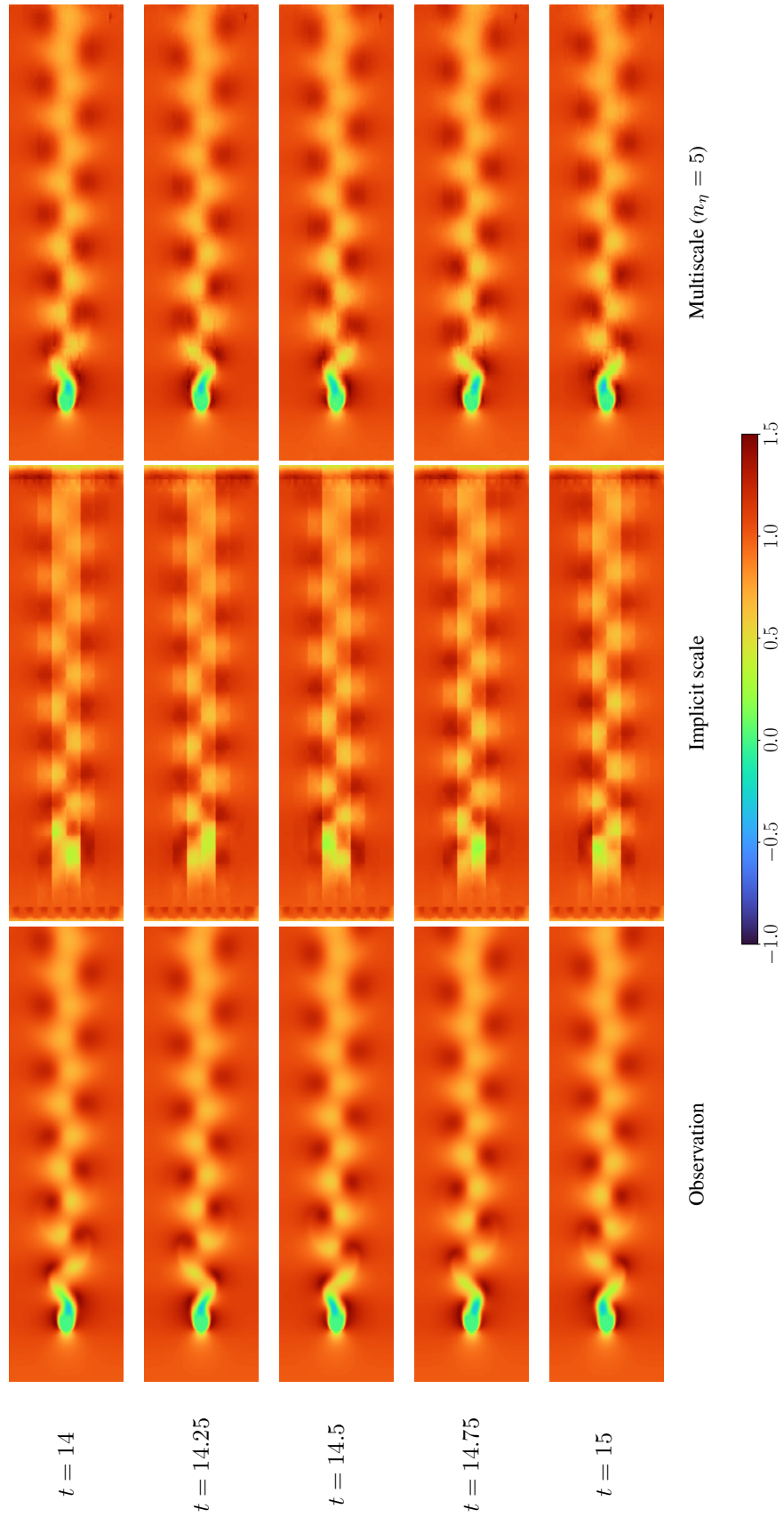


Figure 15: Cylinder flow in 2D: model comparison over test interval. The prediction mean of velocity  $x$ -component is shown here.

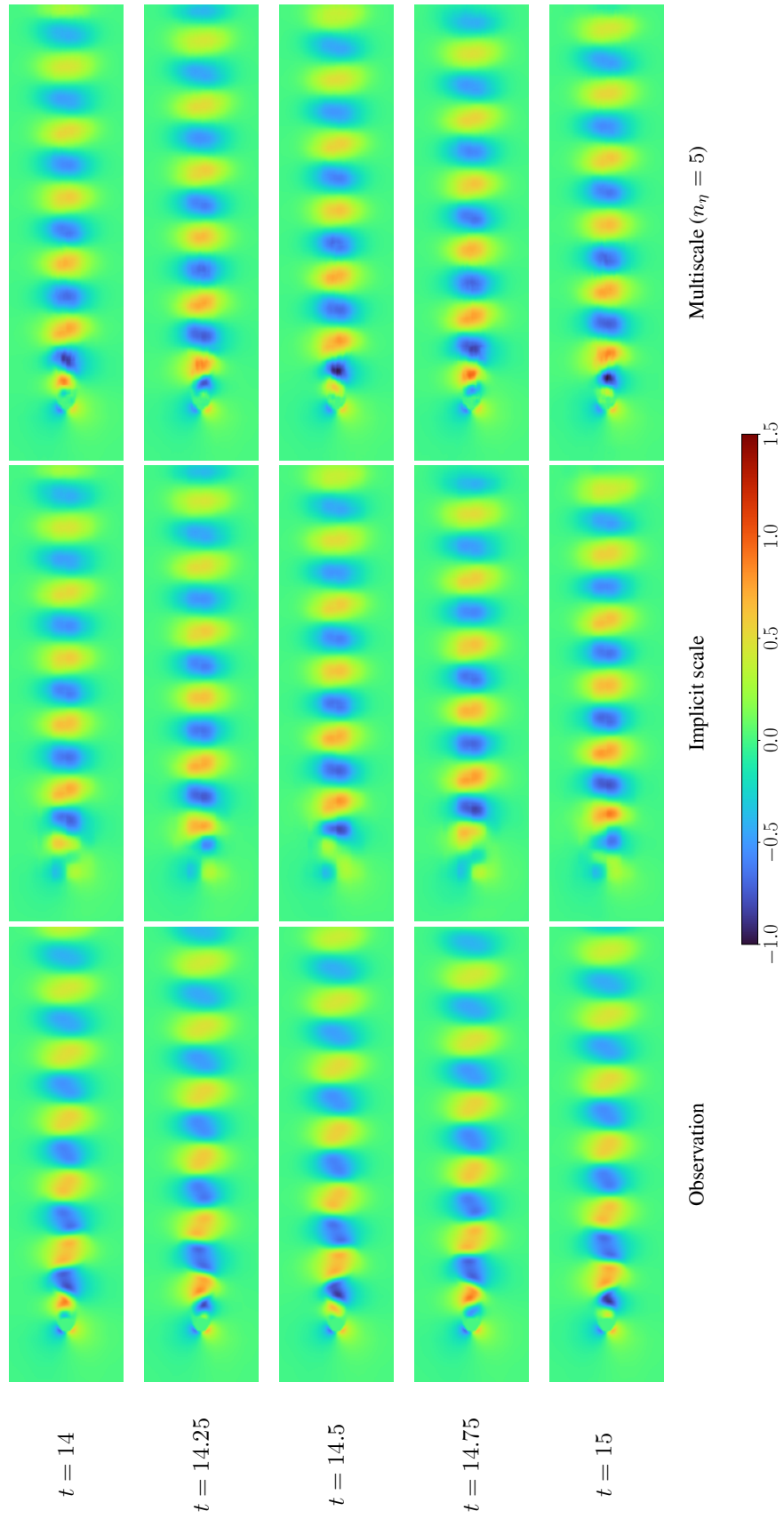


Figure 16: Cylinder flow in 2D: model comparison over test interval. The prediction mean of velocity  $y$ -component is shown here.

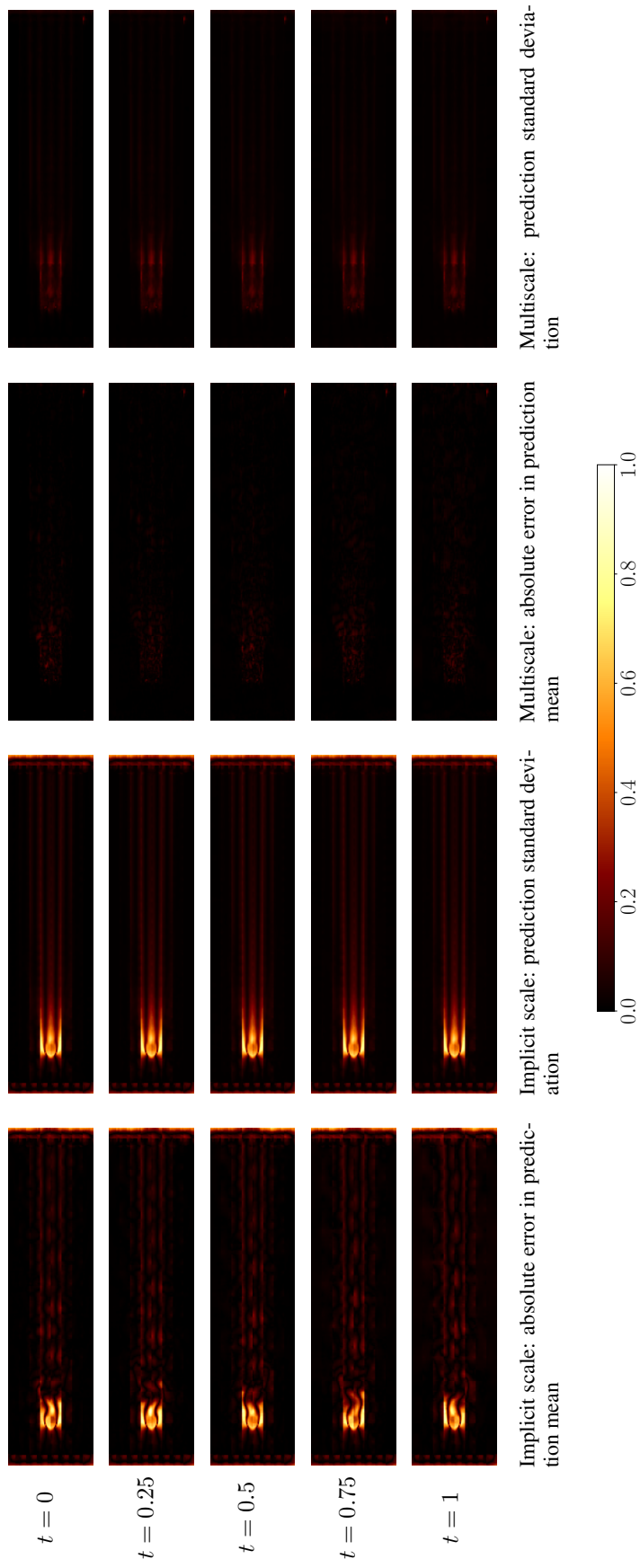


Figure 17: Cylinder flow in 2D: model comparison in spatial error and prediction standard deviation on velocity  $x$ -component with  $n_\zeta = 2 \times 32 \times 8 = 512$  and  $n_\eta = 5$ . All plots on common  $z$ -axis scale.

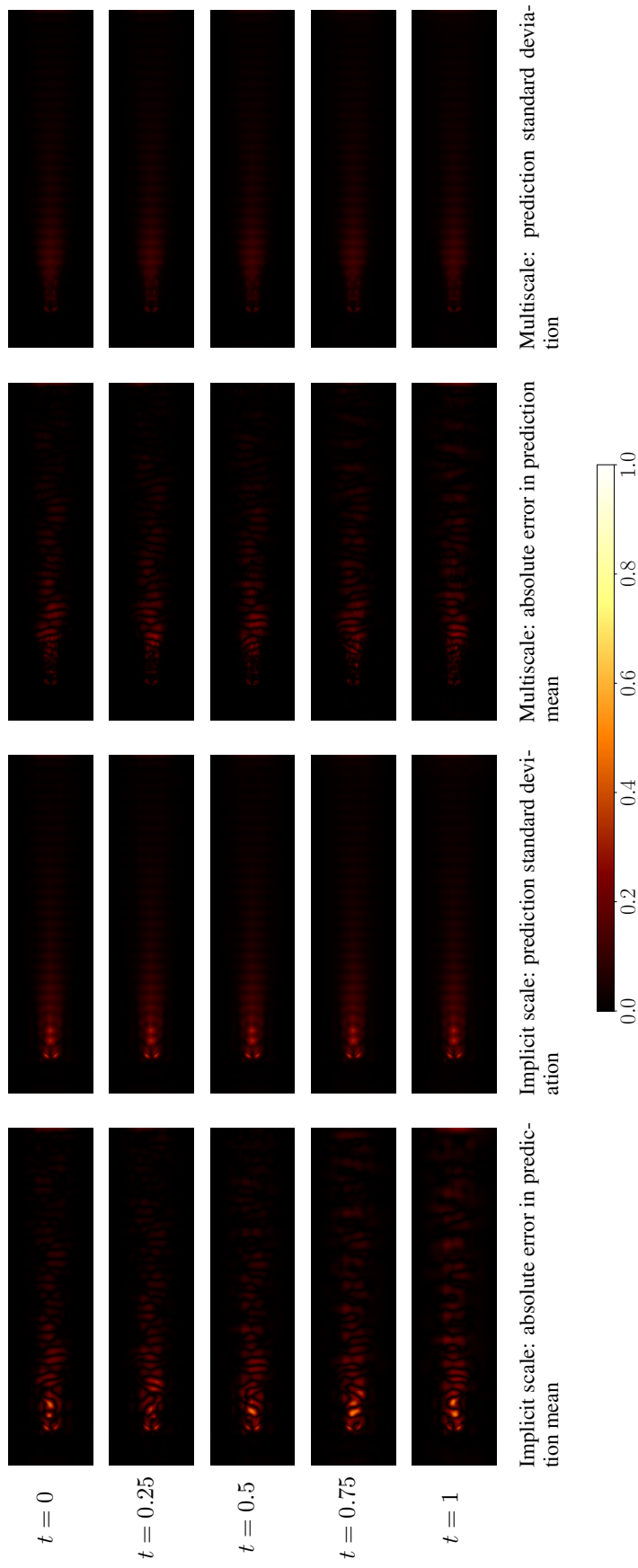


Figure 18: Cylinder flow in 2D: model comparison in spatial error and prediction standard deviation on velocity  $y$ -component with  $n_\zeta = 2 \times 32 \times 8 = 512$  and  $n_\eta = 5$ . All plots on common  $z$ -axis scale.

## I Learning smoothing kernels

In this work, we choose to use a convolution to obtain the macroscale state from the full state. With an appropriate kernel size and stride, the convolution smooths the full state and sparsely samples it on the coarse macroscale mesh. Prolonging the macroscale state onto the full mesh is accomplished by transposing the convolution used to obtain it. Because this operation is lossy, the prolonged macroscale state loses sub-grid-scale features of the full state.

If the kernel of the convolution was simply chosen to be Gaussian, then this operation would amount to a low-pass filter. We choose however to leave the kernel parametrized and to infer it from data. The optimal kernel for a particular dataset may turn out to be non-Gaussian, and indeed this is what we observed in our numerical studies. The kernel for the multiscale model ( $n_\zeta = 20$ ,  $n_\eta = 5$ ) trained on the KdV dataset is shown in Figure 19. Although the training process did not converge on a Gaussian kernel, it did converge on a smooth curve.

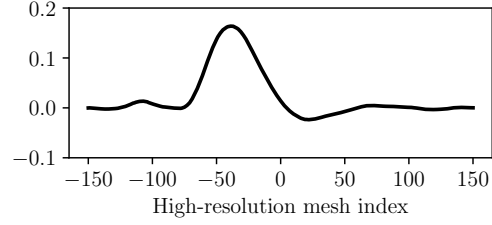


Figure 19: The macroscale smoothing kernel learned for the multiscale model ( $n_\zeta = 20$ ,  $n_\eta = 5$ ) trained on the KdV dataset. Note that adjacent gridpoints on the macroscale mesh are 50 gridpoints apart on the high-resolution mesh.

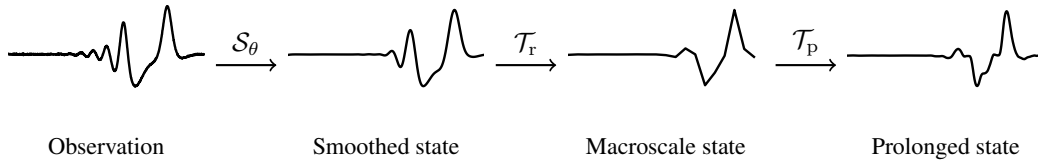


Figure 20: The macroscale smoothing, restriction, and prolongation operators visualized on a snapshot from the KdV dataset.

Figure 20 illustrates the macroscale smoothing, restriction, and prolongation. As our example state, we use the snapshot at  $t = 0.5$  from a test trajectory of the KdV dataset. We see how the smoothed state and especially the macroscale state on its coarsened mesh lose the small-scale features present in the original snapshot. The learned kernel however reintroduces some small-scale features in the prolongation, as determined by the training process to maximize the likelihood defined in Section 4.1.

Figure 21 shows a spectral plot of the example snapshot from Figure 20 compared to the prolonged macroscale state and a multiscale reconstruction. As expected, the prolonged macroscale state accurately follows the true spectrum above the spatial Nyquist frequency of the macroscale mesh. Below this frequency, however, the small-scale features introduced by the deconvolution approximately follow the true spectrum, although they cannot be directly resolved by the macroscale mesh. We hypothesize that it is in this effort to approximate sub-grid-scale features that the training process may prefer non-Gaussian kernels. As expected, the multiscale reconstruction follows the true spectrum much more closely, particularly for the sub-grid-scale components.

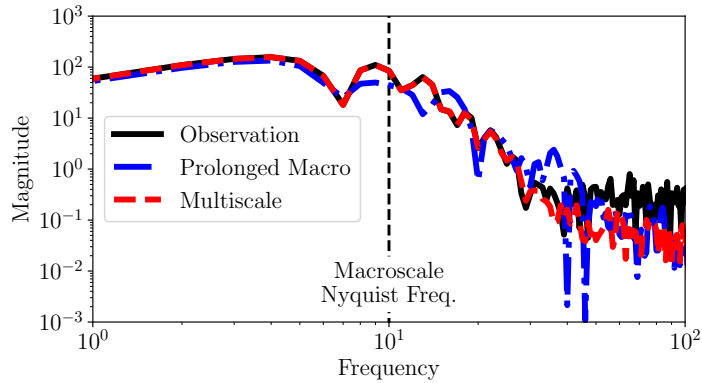


Figure 21: Spectral plot of observation, multiscale reconstruction ( $n_\zeta = 20$ ,  $n_\eta = 5$ ), and prolonged macroscale state using a snapshot from the KdV dataset.