Hybrid solver with local correction using Lagrangian latent memory

Gwendal Jouan*
Siemens Industries Software NV
3001 Leuven, Belgium
gwendal.jouan@siemens.com

Matthias Schulz*
Siemens Industry Software GmbH
13269 Berlin, Germany
schulz.matthias@siemens.com

Daniel Berger
Siemens AG
90475 Nürnberg, Germany
berger.daniel@siemens.com

Stefan Gavranovic
Siemens Industry Software GmbH
81739 München, Germany
stefan.gavranovic@siemens.com

Dirk Hartmann†
Siemens Industry Software GmbH
81739 München, Germany
hartmann.dirk@siemens.com

Abstract

Augmentation of differentiable Computational Fluid Dynamics (CFD) solvers with Neural Networks (NNs) has shown promising results. However, most approaches rely on Convolutional Neural Networks (CNNs) and Cartesian solvers with efficient access to all cell data. This choice poses challenges for industrial solvers that operate on unstructured meshes and with efficient access to neighboring cells only. In this work, we address this limitation using a novel architecture, named *Transported Memory Networks*. The architecture draws inspiration from both traditional turbulence models and recurrent NNs and is compatible with generic discretizations. We demonstrate that it is point-wise and statistically comparable to, or improves upon, previous methods in terms of accuracy and efficiency.

1 Introduction

Computer Aided Engineering (CAE) has a long history in industrial engineering. Multi-physics simulations, including CFD, are among the most used simulation technologies in CAE [LLP22, KHJ22]. Although they have evolved exponentially in terms of speed and accuracy [RWMS18], they still take excessive amounts of time. With the rapid evolution of Machine Learning (ML) technologies, there is a unique potential to address this challenge [KKL$^+$21, Wei21].

Within this contribution, we focus on a hybrid method that follows the concepts introduced by Um et al. [UBF$^+$20] and Kochkov et al. [KSA$^+$21]. In these two approaches, traditional numerical solvers are augmented by ML building blocks to increase their speed and/or accuracy. Both approaches demonstrated remarkable performance, particularly in generalization capabilities. However, they employ CNNs which limits compatibility with industrial CFD codes.

*These authors contributed equally
†Corresponding author

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

Our contributions in this work are as follows: Investigate the impact of the CNN architecture (input stencil size) on performance; Propose an alternative approach that is more suitable for state-of-the-art industrial solvers; Benchmark and evaluate the new approach in comparison to previous approaches. We thereby introduce a new NN architecture, coined Transported Memory Network (TMN). It leverages the concept of long-term memory (similar to LSTM architectures [HS97]) that is effectively transported with the flow. It thus appropriately reflects the corresponding physics behavior in a Eulerian coordinate system that is fixed in space rather than advected with the flow.

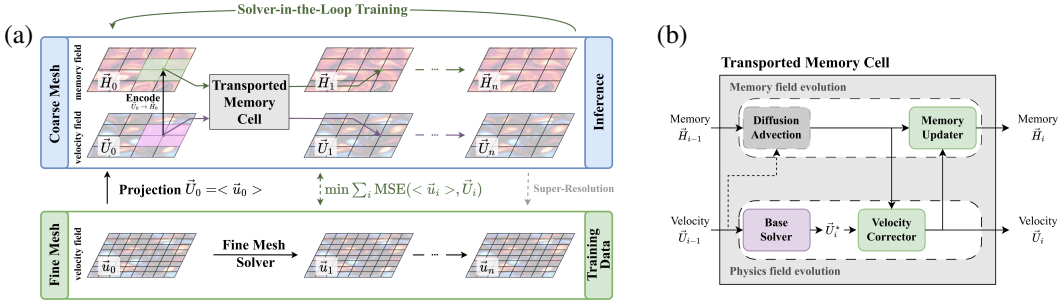

Figure 1: TMN architecture which additionally to physics variables introduces long term memory states that are effectively transported along the fluid flow. The corresponding evolution is handled by the Transported Memory Cell, which combines a base physics solver with a ML augmentation allowing coarse grain discretization without loss of prediction accuracy.

2 Background

2.1 Computational Fluid Dynamics

Within this contribution, we focus on the incompressible Navier-Stokes equations [Ach90]:

$$\rho\left(\partial_t \vec{u} + \vec{u}\cdot\nabla\vec{u}\right) = -\nabla p + \mu\Delta\vec{u} + \rho\vec{f} \qquad \text{and} \qquad \nabla\cdot\vec{u} = 0, \tag{1}$$

where $\vec{u}$ is the flow velocity, p the pressure, μ the dynamic viscosity, $\vec{f}$ an external body force and ρ the constant fluid density. A broad toolbox of numerical solvers is available [KHJ22]. Here, we aim to improve state-of-the-art industrial methods usually based on Finite Volume (FV) schemes. Without loss of generality, we will restrict ourselves to regular structured grids.

2.2 Hybrid CFD Solvers

Hybrid methods enrich a classical solver based on a coarse mesh with ML methods to achieve a comparable accuracy as obtained on a fine mesh [SSMA24]. That is, instead of computing an "exact" velocity $\vec{u}$, they resolve "only" a coarse-grained velocity $\vec{U}$, and introduce additional modifications to Equations (1) ensuring that the effects of unresolved scales are appropriately reflected. Here, we follow closely Um et al. [UBF$^+$20] and Kochkov et al. [KSA$^+$21] adopting a Learned Correction (LC) approach, where a corrective term, a CNN taking as input the current velocity field, is added to the output of a coarse solver prediction after each time step.

2.3 Limits of CNN architectures in industrial solvers

The non-local correction via CNNs[3] allows to indirectly resolve the Lagrangian nature of flows within Eulerian formulations by going back in space instead of going back in time. Our experiments have shown that the larger the non-locality of the correction, the more accurate predictions are (see Section D.2). However the application of CNNs in industrial high-end CFD solvers remains impractical. While replacing CNN with Graph Neural Networks may seem like a natural extension to handle unstructured meshes [SMV23] one challenge remains. Most solvers adopt a matrix-free approach, which allows efficient information access only for nearest neighbors.

[3]E.g., Kochkov et al. [KSA$^+$21] use 7 layers of convolutions with 3×3 kernels, implying that the network uses information from a 15×15 stencil around each computational cell.

3 Transported Memory Networks

Within this paper, we propose a new architecture, that is compatible with small computational stencils while still achieving the desired level of performance. Inspired by recurrent NN architectures [HS97, GS00, CVMG$^+$14] our approach takes into account a long term state memory encoded by a hidden state vector $\vec{H}(t)$, defined cell / discretization point-wise in addition to the velocity components carried by each cell (see Figure 1). Given the Lagrangian nature, the hidden state vector is effectively learned to be transported with the flow, as we show in Section D.3.

Concretely, we use the FV approach on a staggered grid [GDN98] based on the JAX-CFD[4] code of Kochkov et al. [KSA$^+$21] and follow the Algorithm 1 (where the subscripts θ denote the learnable functions). Please note, that we are not adding a physics prior by introducing an explicit transport of $\vec{H}$. For initializing the hidden state vector we use a separate encoder network which computes the initial condition for the hidden state vector from the initial condition of the velocity field, i.e, $\vec{H}_0 = \textbf{enc}_\theta(\vec{U}_0)$. At encoding we do not have access to any previous snapshots in time and thus use a CNN to encode the "history" of the flow[5]. For further details of the architecture see Appendix A.

Algorithm 1 Learned correction with hidden states

Set initial conditions $\vec{U}_0$
Encode initial hidden states $\vec{H}_0 = \textbf{enc}_\theta(\vec{U}_0)$
while $1 \leq i \leq N$ **do**
  Update velocity with base solver $\vec{U}_i^* = \textbf{solver}(\vec{U}_{i-1})$
  Correct velocity $\vec{U}_i = \vec{U}_i^* + \textbf{corr}_\theta(\vec{U}_i^*, \vec{H}_{i-1})$
  Update hidden states $\vec{H}_i = \textbf{up}_\theta(\vec{U}_i, \vec{H}_{i-1})$
end while

4 Experiments

4.1 Setting

All models are trained with a base solver operating on a 64×64 grid with reference data coming from a 2048×2048 discretization[6]. For comparison sake, we strictly follow Kochkov et al. [KSA$^+$21] for the training data, test cases and performance metrics. See Appendix C and corresponding reference for further details. All numbers reported are taken from an average over 16 different trajectories.

4.2 Autoregressive Training

The networks are trained by minimizing the loss function :

$$L_T = \sum_{i=1}^{T} \mathsf{MSE}\left(\vec{U}_i^{\text{exact}}, \vec{U}_i\right) + \sum_{j=0}^{T/N} \mathsf{MSE}\left(\textbf{enc}_\theta(\vec{U}_{1+jN}^{\text{exact}}), \vec{H}_{1+jN}\right) \quad \text{for } T \in \{16, 32, 64\}. \quad (2)$$

The first term is the standard mean square error between the down-sampled high resolution velocity field $\vec{U}_i^{\text{exact}}$ and the ML augmented solver velocity field $\vec{U}_i$ (see [KSA$^+$21] for more details). The second term computes the difference between the hidden state vector $\vec{H}_i$ after N time steps and the output of the encoder taking as input the velocity at this same N^{th} time step. This term ensures that the hidden state that is encoded at the end of a time batch will be similar to the one that is encoded at the beginning of the next (in the same trajectory).

As most hybrid approaches, we rely on an autoregressive training approach. That is, during training sets of velocity trajectories $(\vec{U}_i, \vec{U}_{i+1}, \dots \vec{U}_{i+N})$ are compared to the down-sampled outputs of a reference high resolution solver. The loss, measuring the mismatch, is then minimized through gradient descent. That is, the learned model is applied many times, autoregressively, for a single training step and thus a trajectory is fit [MCK$^+$23]. See Appendix B for further details on the training.

[4] https://github.com/google/jax-cfd

[5] The encoder is used only at the start of the simulation, leaving the locality during solver iterations intact.

[6] Data will be made available upon request.

4.3 Benchmarking the TMN approach

Point-wise Accuracy: As a metric for point-wise accuracy, we evaluate the Pearson correlation of the vorticity field $\nabla \times \vec{U}$ of the models with the one obtained from reference high resolution solvers. In particular, to facilitate comparison, we look at the time at which the correlation falls below 0.95.

In Figure 2a we report the results of the TMN model for different sizes of the hidden state vector and for the various cases: *forced turbulence* (same setting as the training data), *decaying turbulence*, *larger domain* and *more turbulent* (see Appendix C). Figure 2b shows the correlations over time for the *forced turbulence* case. While on average the model accuracy does improve with the number of hidden states, the trend is not as pronounced as in the case of the stencil size (see Section D.2). Overall, a hidden state vector of dimension 8 is enough to obtain comparable results to the full CNN architecture, but even models with 4 hidden states are competitive in most cases. For an evaluation of the statistical accuracy we refer to Appendix D.1.

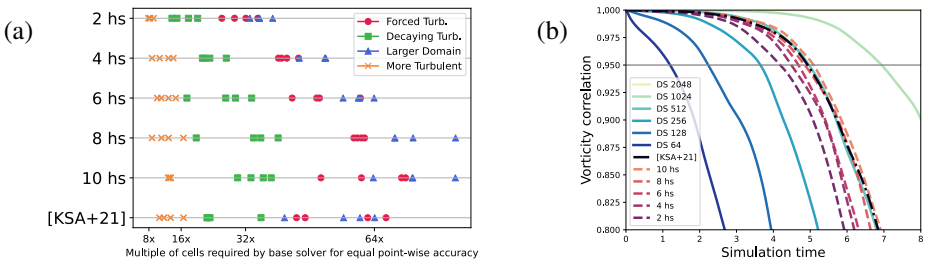

Figure 2: (a) Simulation time until correlation drops below 0.95 for models with different number of hidden states (hs) and 4 random initializations. (b) Correlation vs time for direct simulations (DS) with different discretizations and models with varying number of hidden states for *forced turbulence*.

Speed: In Table 1 we report the speedup compared to the reference solution on a 512×512 grid which has a comparable accuracy (c.f., Figure 2b) [7]. All models were run on a single Nvidia A10G GPU. These timings demonstrate that the TMN architecture is computationally faster for all models compared with the CNN model from Kochkov et al. [KSA$^+$21] without any performance optimization regarding the latency of the TMN models. We expect additional speedup going from 2D to 3D due to cubic scaling of compute effort in 3D versus quadratic scaling in 2D.

Table 1: Speedup of simulation time for different sizes of the hidden state vector.

# Hidden states	2	4	6	8	10	12	Kochkov et al. [KSA$^+$21]
Speedup	99×	95×	93×	90×	88×	85×	86×

5 Discussion and Limitations

In this paper we propose a novel architecture, coined TMN, for the hybrid CFD solvers introduced in [UBF$^+$20, KSA$^+$21] which rely on a CNN architecture. Different to CNN architectures, our architecture is well suited for integration into state-of-the-art industrial CFD solvers. Our results show that the TMN architecture compares very well with the CNN based architecture by Kochkov et al. [KSA$^+$21], yet shows advantages in terms of inference speed. Compared to classical CNN the approach relies only on direct neighbor information and thus is relatively geometry agnostic. This will likely improve generalization capabilities in realistic industrial scenarios.

While the proposed methodology is compatible with unstructured meshes, full integration of the approach into a general 3D unstructured solver remains to be demonstrated (work in progress). Handling of more general boundary conditions and variable time stepping also need to be addressed to be able to tackle industrial applications.

[7] Since we are concerned with industrial solvers which primarily use FV methods, we consider this a fair baseline, whereas [MH24] suggested a stronger baseline using Pseudo-Spectral methods [STW11]

References

[Ach90] David J Acheson. *Elementary fluid dynamics*. Oxford University Press, 1990.

[CVMG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[GDN98] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. *Numerical simulation in fluid dynamics: a practical introduction*. SIAM, 1998.

[GS00] Felix A Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE, 2000.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[KHJ22] David Kelsall, Steve Howell, and Uwe Janoske. CFD - a timeline - from the 17th century to the present day. In *BENCHMARK - 100 years of CFD*. NAFEMS, 10 2022.

[KKL+21] 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, 2021.

[KSA+21] 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.

[LLP22] Wing Kam Liu, Shaofan Li, and Harold S Park. Eighty years of the finite element method: Birth, evolution, and future. *Archives of Computational Methods in Engineering*, 29(6):4431–4453, 2022.

[MCK+23] Hugo Melchers, Daan Crommelin, Barry Koren, Vlado Menkovski, and Benjamin Sanderse. Comparison of neural closure models for discretised PDEs. *Computers & Mathematics with Applications*, 143:94–107, 2023.

[MH24] Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations, 2024.

[RWMS18] Ulrich Rüde, Karen Willcox, Lois Curfman McInnes, and Hans De Sterck. Research and education in computational science and engineering. *Siam Review*, 60(3):707–754, 2018.

[SMV23] Varun Shankar, Romit Maulik, and Venkatasubramanian Viswanathan. Differentiable turbulence ii, 2023.

[SSMA24] Benjamin Sanderse, Panos Stinis, Romit Maulik, and Shady E Ahmed. Scientific machine learning for closure models in multiscale problems: a review, 2024.

[STW11] Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral methods: algorithms, analysis and applications*, volume 41. Springer Science & Business Media, 2011.

[UBF+20] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[Wei21] E Weinan. The dawning of a new era in applied mathematics. *Notices of the American Mathematical Society*, 68(4):565–571, 2021.

A Transported Memory Network

The TMN model is comprised of three different learnable components that work on top of a base solver (see Algorithm 1). The hidden state encoder generates the initial condition for the hidden state vector based on the initial velocity field. The velocity corrector updates the velocity based on its current value and the hidden state vector, and finally the hidden state updater updates the hidden state vector to the next time step based on its current value and the corrected velocity.

The architectures of the three components are depicted in Figure 3[8]. Note that for both the velocity corrector and the hidden state updater, the first convolution layer employs a 3×3 kernel while all subsequent ones use 1×1 kernels. The sigmoid used as final output function for both the encoder and the hidden state updater bounds the hidden state vector components within $[-1, 1]$. Depending on the size of the hidden state vector, the total number of parameters for the 3 trainable networks range between $56,210$ (12 hidden states) to $42,430$ (2 hidden states).

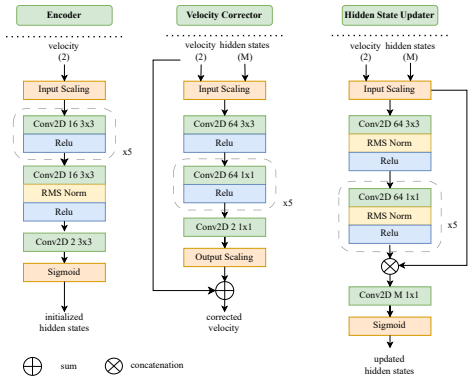

Figure 3: Local networks architecture, size of the velocity and hidden state vector in brackets.

B Training parameters

For all the models we used the Adam optimizer with an initial learning rate of 0.001 and exponential decay with rate of 0.5 and $50,000$ transition steps.

The length of the unrolled trajectories is progressively increased during training. In practice, we found that starting the training on trajectories composed of 16 time steps, and then increasing its length to 32 and finally to 64 time steps worked best for the TMN approach. These are obtained by splitting up longer trajectories of $4,800$ sequential time steps. At each time batch length, 400 training epochs are performed producing converging loss curves.

C Training data and test cases

As in [KSA+21], we consider a 2D Kolmogorov flow in a box of side-length 2π on a structured grid with periodic boundary conditions, Reynolds number $RN = 1,000$, $\rho = 1$, $\mu = 0.001$, $\Delta t = 7.0125 \cdot 10^{-3}$, and $f = \sin(4y)\vec{e}_1 - 0.1\vec{U}$ (*forced turbulence*), where $\vec{e}_1$ is the basis vector in x-direction. We generate 50 trajectories resulting from different random initial conditions. Two are kept for the validation set and 16 for the test set. We also test our model on the three generalization cases from Kochkov et al. [KSA+21]: a *larger domain* with increased side-length of 4π, a *more turbulent* Kolmogorov flow with $RN = 4,000$, and a *decaying turbulence* case where the forcing is removed.

[8] Even though those networks are implemented using CNNs, they are equivalent to a single Multi-Layer Perceptron (MLP) sliding over the grid with an input stencil of 3×3 (i.e., only using direct neighbors information).

D Further Experiments

D.1 Statistical Accuracy

For statistical accuracy, we compute the energy spectrum $E(k) = 0.5|\vec{U}(k)|^2$, where k is the wavenumber, and evaluate the spectral error defined as the average absolute error of $E(k)k^5$ between the reference solution and the respective model output.

Figure 4a shows the energy spectrum (scaled by k^5 where k is the wave number) for the various baselines, TMN models with 2 and 10 hidden states and for the model from [KSA$^+$21]. The learned models agree well with the high resolution baseline solution except at $k = 1$. Figure 4b depicts the spectral error for the 2 to 10 hidden states TMN models. When satisfied with statistical accuracy, 2 hidden states is enough to reach a comparable accuracy as the original LC model in Kochkov et al. [KSA$^+$21].

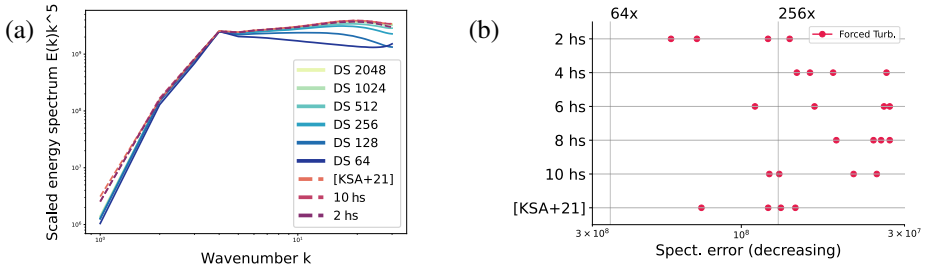

Figure 4: (a) the energy spectrum scaled by k^5 over wavelength is shown for direct simulations (DS) and learned models for *forced turbulence*. (b) the spectral error is depicted for different random initializations (dots), vertical lines indicate multiple of required cells by the base solver.

D.2 Effect of the Input Stencil Size

The original LC model in Kochkov et al. [KSA$^+$21] uses information coming from a 15×15 stencil around a given cell to compute the correction for that same cell. To identify the influence of the stencil size on model performance, we track the accuracy of a series of models with varying stencil size. The models consist of a variable number of convolution layers with 3×3 stencils, followed by two layers with 1×1 kernel that act as final, local MLP (see Figure 5a). Thus, the stencil size equals to $2N + 1$ where N is the number of convolution layers with 3×3 kernels. Figure 5 (Left) shows a diagram of the series of CNN models used. The only variable parameter between models is N, the number of convolution layers with 3×3 kernels, going from $N = 6$ for the 13×13 input stencil to $N = 1$ for the 3×3 input stencil. The number of trainable parameters for each networks range from $190,146$ to $5,506$.

As a metric for point-wise accuracy, we evaluate the Pearson correlation of the vorticity field $\nabla \times \vec{U}$ produced by the models with the one obtained from reference high resolution solvers. To facilitate comparison across models, we look in particular at the time at which the correlation falls below 0.95. The time until the correlation falls below 0.95 for the various stencil sizes are reported in Figure 5b for the *forced turbulence* case, with the performance of the model from Kochkov et al. [KSA$^+$21] added as reference. As expected the model performance improves with larger stencils.

D.3 Transport of Hidden State Vector

Figure 6 shows the evolution of the first two components of the long term state $\vec{H}$ for one of the trajectories of the test set in the forced turbulence case. Also, the vorticity of the velocity field is pictured. As can be seen from the plots, the hidden state components are highly correlated with the vorticity. This indicates that the update

$$\vec{H}_i = \mathbf{up}_\theta(\vec{U}_i, \vec{H}_{i-1}) \tag{3}$$

effectively learns a transport equation (the vorticity itself is transported with the flow and any transported quantity would follow a similar pattern). We stress that during training the model never sees the vorticity directly, but only the velocity components, see Equation (2).

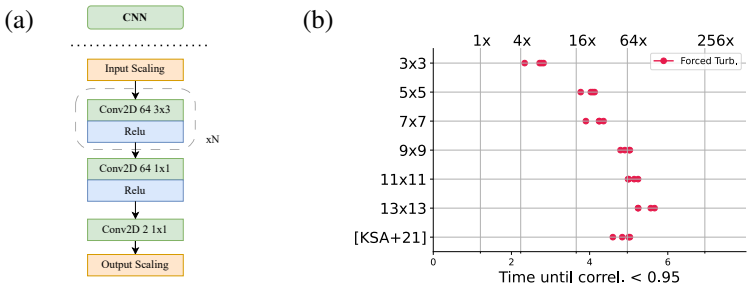

Figure 5: (a) Architectures for the stencil size study (b) Simulation time until correlation drops below 0.95 for models with different input stencil sizes and 4 random initializations, vertical lines indicate the multiples of cells required by base solver to achieve the respective timings.

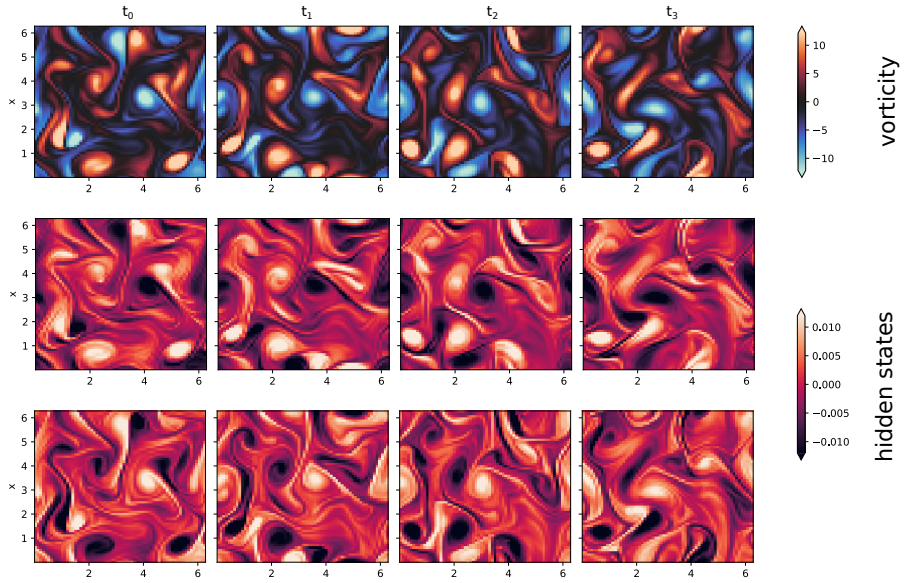

Figure 6: Vorticity (row 1) and first 2 hidden states (rows 2-3) show remarkable resemblance.

As an alternative to the direct update of Equation (3), and taking inspiration from classical turbulence model, we also tried to explicitly transport the hidden states and only learn a source term. That is,

$$\vec{H}_i = \textbf{trsprt}\left(\vec{U}_i, \vec{H}_{i-1}\right) + \mathbf{s}_\theta(\vec{U}_i, \vec{H}_{i-1}) \tag{4}$$

with a learnable source component $\mathbf{s}_\theta$ and **trsprt** modelling the transport of quantity $\vec{H}_{i-1}$ by the velocity field $\vec{U}_i$ given by the base solver, i.e.,

$$\textbf{trsprt}(\vec{U}, H^i) = -\nabla \cdot (H^i \vec{U}) + \nabla \cdot (\xi(\vec{U}, \vec{H}) \nabla H^i)$$

with H^i the i-th component of $\vec{H}$. We have investigated

- advection only, i.e. $\xi \equiv 0$,
- advection-diffusion with learnable constant viscosity ξ,
- advection-diffusion with learnable non constant viscosity with $\xi(\vec{U}, \vec{H})$ parametrized by a simple NN.

However, we did not observe a performance improvement adding a physics prior and in some cases even observed a negative impact on performance.