# Bridging Dynamical Models and Deep Networks to Solve Forward and Inverse Problems

**Marie Déchelle** *
Sorbonne Université, MLIA, LIP6,
marie.dechelle@lip6.fr

**Jérémie Donà** *
Sorbonne Université, MLIA, LIP6
jeremie.dona@lip6.fr

**Kévin Plessis-Fraissard** *
Sorbonne Université, MLIA, LIP6,
plessis@poleia.lip6.fr

**Patrick Gallinari**
Sorbonne Université, MLIA, LIP6
Criteo AI Labs
patrick.gallinari@lip6.fr

**Marina Levy**
Sorbonne Université, LOCEAN, IPSL
marina.levy@locean.ipsl.fr

## Abstract

Modeling the dynamics of physical systems recently gained attention in the machine learning community. Most recent works rely on complete observations of the physical state, whereas only partial observations are available in practice. Estimating the full state dynamics is important for the understanding of the underlying phenomenon and for model based prediction. Largely unexplored from an ML viewpoint, we address in this work the estimation and forecast of a partially observed spatio-temporal system, leveraging prior dynamical knowledge. To solve both forward (forecasting) and inverse (identification) problems, we bridge numerical models of partial differential equations and deep learning and introduce a dynamical regularization on the unobserved states. This constrains our estimation and improves estimation performances. The approach is validated on two simulated datasets where the dynamics is controlled and fully known.

## 1 Introduction

Modeling the dynamics of physical phenomena classically relies on ordinary or partial differential equations (ODE and PDE), involving two practical issues: prediction and estimation of the unobserved states and dynamics, respectively the forward and inverse problem. Solving both problems can be critically enabling for example for transport equations, key in climate and ocean circulation models [10]. Both tasks raise major issues: for instance, forward modelling through data assimilation [20] is computationally expensive while inverse problems are intrinsically ill-posed [28].

Recent trends consider Machine Learning (ML) as an alternative or complementary approach to traditional physical models, allowing the integration of observations and potentially faster computations through model reduction. To solve forward modeling, machine learning leverages complex recurrent architectures able to deal with high-dimensional partially observed systems [29, 7, 32]. This pure ML approach is however not sufficient when dealing with complex physical dynamics. Large size inverse problem resolution with deep models recently gained attention [2, 6, 3], but they do not provide interpretable estimation of the unobserved state. [18, 21] use prior knowledge in the forward model

*Equal contribution

1st NeurIPS workshop on Interpretable Inductive Biases and Physically Structured Learning (2020), virtual.

and fully observed state to recover unknown parameters, but are limited to low dimensional settings. To cope with the ill-posedness in inverse problem resolution and retrieve meaningful estimation of the unobserved state, further regularization is needed [23]. In that perspective promising works [6, 27, 19, 25] consider strong inductive priors.

We propose a framework bridging numerical schemes of PDE and neural networks to approximate the forward model. Incorporating physical knowledge, our framework considers the dynamics of the observed variable and its known dependency to unknown variables. Thanks to a dynamical regularization of the hidden states, the model also solves the inverse problem. We illustrate the ability of our approach to tackle high dimensional partially observed systems and show its effectiveness on simplified dynamics of ocean circulation.

## 2 Related Work: Neural Representation of Differential Equations

Our work derives from the neural representation of PDE. Initiated by the similarity between forward Euler scheme for ODEs and Residual Networks (ResNets) [14], a flourishing literature has thrived developing links between ODE/PDE and neural networks, ranging from analysis involving the continuous limit of ResNets [5] to the neural implementation of sophisticated numerical schemes [13]. Recently, some works leveraged data specific knowledge to shape the prediction function, for example imposing specific fluid dynamic [26] or Hamiltonian constraints [11, 31]. Considering dynamics driven by PDEs, [6] estimates a velocity field from data, under the constraint of an advection-diffusion PDE to derive a forecast scheme. [3] goes further and proposes a general scheme for modeling dynamical systems governed by unknown PDEs from partial observations. However, in both settings, the estimated field lacks physical interpretability. The forward model can be constrained as in [19] where convolutional filters of residual networks are forced to approximate euclidean differential operators. Very recently, [12, 21] proposed to combine physics informed solvers and deep networks for both forward and inverse problems, requiring however full state observation. Because we only use partial information about the true PDE and partial observations of the dynamics, our work extends hybrid models [12, 6, 21]. It enables physically plausible estimates of the unobserved states, opening the way to well-built combination between neural estimation and numerical scheme of partial differential equations.

## 3 Model

We denote by $X_t \in \Omega \subseteq \mathbb{R}^{p+q}$ a partially observed physical state, written as $X_t = (X_t^{obs}, Y_t) \in \mathcal{X} \times \mathcal{Y}$ where $X_t^{obs} \in \mathcal{X} \subseteq \mathbb{R}^p$ is observed and $Y_t \in \mathcal{Y} \subseteq \mathbb{R}^q$ is unobserved. Our primary goal is to estimate the full physical state $X_t$ from partial observations, i.e from a sequence of $X_t^{obs}$.

### 3.1 Solving an Inverse Problem via Forward Modeling

We assume that the evolution of $X_t$ is governed by a differential equation with a dynamic $f$:

$$X_t = \begin{pmatrix} X_t^{obs} \\ Y_t \end{pmatrix} \tag{1a}$$

$$\frac{dX_t}{dt} = f(X_t) \tag{1b}$$

We formulate our problem as retrieving the full state $X_t$ given a sequence of observed variables $X_t^{obs}$. This problem is ill-posed since $f$ is not known and no observation of $Y$ is available. Unlike [6] who has access to a closed form solution to estimate $f$, we only rely on partial information on the forward model. Our contribution involves two interlinked dimensions: 1) learning jointly estimates of $Y$ and 2) learning the whole dynamics of $X$ (both observed and unobserved) by incorporating partial physical knowledge. We give below a formulation of these two steps.

**State Estimation Formulation** Relying on adjoint state method [24], we learn a neural network $G_\theta$ estimating the unobserved $Y_t$ from $k$ consecutive measurements $(X^{obs})_{t-k:t} = (X_{t-k+1}^{obs}, ..., X_t^{obs})$:

$$\begin{aligned} G_\theta : \mathcal{X}^k &\to \mathcal{X} \times \mathcal{Y} \\ (X^{obs})_{t-k:t} &\to \widehat{X}_t = (X_t^{obs}, \widehat{Y}_t) \end{aligned} \tag{2}$$

However, having no supervision for the learning of $G_\theta$, we leverage the dynamics of the observed variable to retrieve a physical estimate of the full state through data assimilation principles [1, 22].

**Dynamical Model Formulation** We rewrite $f$ from Equation (1b) as $f = (f_1, f_2)$ acting respectively on $X^{obs}$ and $Y$:

$$\frac{dX_t}{dt} = \frac{d}{dt}\begin{pmatrix} X_t^{obs} \\ Y_t \end{pmatrix} = \begin{pmatrix} f_1(X_t) \\ f_2(X_t) \end{pmatrix} \tag{3}$$

Observing physical variables provides information about their dynamics. For example when studying transport motions, we know that advection is involved. It leads us to assume that $f_1$ is partially known, allowing us to write without loss of generality $f_1$ as $f_1(X_t) = f_1^k(X_t) + f_1^u(X_t)$, where $f_1^k$ is known a priori and $f_1^u$ is to be estimated. This leads to the following decomposition:

$$\frac{dX_t}{dt} = \begin{pmatrix} f_1^k(X_t) + f_1^u(X_t) \\ f_2(X_t) \end{pmatrix}, \tag{4}$$

Note that despite $f_1^k$ is known, it involves the unobserved quantity $Y$. To sum up, to accurately predict the dynamics of $X^{obs}$, we aim at learning $f_1^u$ while accurately estimating $Y$.

There is usually no guarantee that $\hat{Y}_t$ is coherent temporally nor physically interpretable [3]. Therefore, our proposition is to estimate $f_2$, i.e. to force the unobserved $Y$ to obey a PDE and to make the trajectory of $Y$ well defined from an initial datum estimated thanks to $G_\theta$. One key insight in our work is that we learn the dynamics of $Y$ using a PDE, regularizing the estimation. In our work, $f_2$ is implemented with a ResNet, which can be viewed as approximating a transport equation (see for example [17, 16]). Also, the method of characteristics (see Section 6.3) provides existence and uniqueness of the solution to the Cauchy problem associated to the transport equation under mild assumptions.

We formulate our optimization problem accordingly and verify experimentally the effect of our regularization. Note that [12] proposes to inform the forward model and to solve ill-posedness by minimizing the $\ell_2$-norm of $f_1^u$, which will be investigated as a baseline (denoted with $\|f_1^u\|$).

### 3.2 Control of the Dynamic and Training Objective

As previously stated, we want to accurately estimate the dynamics of the observed variable, but also to model the intrinsic dynamics of the unobserved variable $Y$. We have access to partial observations up to $t_0$ and want to forecast the full state from $t_0$ to $T$. We consider the following objective:

$$\min_{f_1^u, f_2} \left\| Y_{t_0+T} - \left( Y_{t_0} + \int_{t_0}^{t_0+T} f_2(X_t)dt \right) \right\|_2 \texttt{ subject to } \frac{dX_t^{obs}}{dt} = f_1(X_t), \tag{5}$$

Unfortunately, having no access to the true $Y$ we only rely on estimates given by $G_\theta$. In order to solve Equation (5), we introduce two losses: we penalize the forecasts errors in the observed state, and force the unobserved variable $Y$ to obey a learned dynamics $f_2$.

Let $F_1^k, F_1^u, F_2$ be the numerical integration of $f_1^k$, $f_1^u$, $f_2$ over one time step. $F_2^n$ is $F_2$ iterated $n$ times. In practice, having no priors on the dynamics $f_1^u$ and $f_2$, we directly estimate their integrated counterpart $F_1^u$ and $F_2$ using a neural network. $F_1^u$ shares the same inputs as $G_\theta$.

**Forecasting loss on $X^{obs}$** For $Y$ to be estimated properly, it must lead to accurate predictions of $X^{obs}$. Thus, we penalise the discrepancy between forecasts of $X^{obs}$ and their true value:

$$\mathcal{J}_{X^{obs}} = \sum_{t=t_0}^{T} \left\| F_1^k\left(\widehat{X}_t^{obs}, F_2^{t-t_0}(G_\theta(X_{t_0-k:t_0}^{obs}))\right) + F_1^u\left(\widehat{X}_{t-k:t}^{obs}\right) - X_{t+1}^{obs} \right\|_2, \tag{6}$$

Note that for $t < t_0$, $\widehat{X}_t^{obs}$ refers to actual observations, while for $t \geq t_0$, $\widehat{X}_t^{obs}$ is the prediction done using former time steps estimations such that $\widehat{X}_{t+1}^{obs} = F_1^k\left(\widehat{X}_t^{obs}, F_2^{t-t_0}(G_\theta(X_{t_0-k:t_0}^{obs})\right) + F_1^u\left(\widehat{X}_{t-k:t}^{obs}\right)$.

Table 1: MSE Predictions (x100) scores of compared models across 10 time steps.

| Models | Advection | | | Source Dataset | | |
|---|---|---|---|---|---|---|
| | $\psi$ | $w$ | $S$ | $\psi$ | $w$ | $S$ |
| Ours (u,v known) | 0.00 | n/a | 0.00 | 0.19 | n/a | 0.12 |
| Ours | 2.11 | 4.90 | 0.23 | 4.97 | 4.96 | 0.89 |
| Ours ($\|f_1^u\|$) | 2.04 | 4.49 | 0.07 | 3.11 | 10.10 | 1.11 |
| Ours (no $F_2$) | 0.95 | 8.28 | 1.07 | 2.98 | 15.71 | 9.06 |
| Ours (no $F_2$, $\|f_1^u\|$) | 1.07 | 9.07 | 1.09 | 1.00 | 11.74 | 4.48 |
| NeuralODE | 3.17 | n/a | n/a | 5.24 | n/a | n/a |

**Evolution of $Y$** Besides, we are interested in the dynamics of the unobserved $Y$ and constrain it to obey a partial differential equation defined by $F_2$:

$$\mathcal{J}_{Y,t} = \left\| F_2^{t-t_0+1} \circ G_\theta(X^{obs}_{t_0-k:t_0}) - G_\theta(X^{obs}_{t-k+1:t+1}) \right\|_2, \tag{7}$$

The optimization of Equation (5) consists in learning the parameters of ($G_\theta$, $F_2$, $F_1^u$) and by minimizing our overall cost function $\mathcal{J}$ defined by:

$$\mathcal{J} = \mathcal{J}_{Y,t=T} + \lambda_{X^{obs}} \mathcal{J}_{X^{obs}} \tag{8}$$

Note that our model can be adapted to various data specific scheme such as fully Lagrangian or symplectic integration and more general all-purpose integrator such as `Rk4`, as long as differentiability is maintained while computing $F_1^k$.

## 4 Experiments

**Physical Dataset** We validate our approach on simplified simulations of ocean surface variables. Our dataset is composed of simulated sea surface temperature (SST) dynamics $\psi$ advected by a velocity field $w$ plus a forcing term, denoted $S$, as:

$$\frac{\partial \psi(x,y,t)}{\partial t} + \nabla.(\psi w(x,y,t)) = S(w,x,y,t), \tag{9}$$

We take as initial condition real ocean temperatures from [20]. In our experiments $X^{obs} = \psi$ and $Y = w$. Several types of velocity fields $w$, representing the Gulf-Stream, are simulated following [4]. From this distribution of $w$, we investigate two experimental settings: no source term ($S = 0$) and a non null source term $S$ inspired by [9]. Thorough experimental details and hyperparameters are available in the appendix Section 6.2 and Section 6.4. Note that the unobserved $w = (u, v)$ is an high-dimensional vector field, contributing to the difficulty of the task.

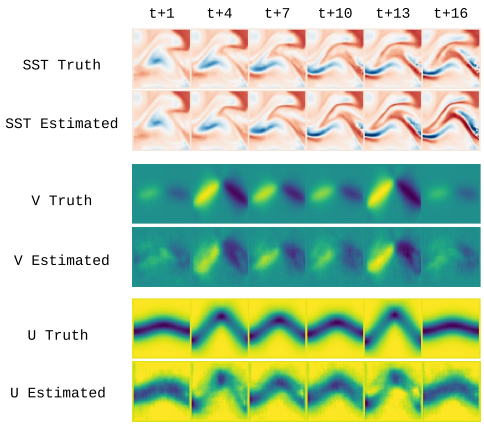


Figure 1: Sequence of estimations on $\psi$, $w = (u, v)$ for the Dataset with non null physical source

We consider as physical prior on the dynamics $F^k(\psi, w) = \nabla.(\psi w)$, representing the advection of the observed quantity $\psi$, implemented as a semi-Lagrangian scheme (see Section 6.3), known for its stability in time.

**Inverse Problem** Figure 1 shows examples of predicted hidden states and columns labeled $w$ give the MSE between estimation and target hidden state $w$. Both Figure 1 and Table 1 show that $G_\theta$ truthfully estimates the hidden state using our framework. Minimizing $\|f_1^u\|$ as in [12] does not help the estimation of the unobserved variables, as confirmed by the conducted ablation study. Therefore, our dynamical prior on $w$ helps solving the inverse problem.

**Forward Problem** Constraining temporally the hidden states harms prediction accuracy despite truthful estimates in both SST (Figure 1) and source term $S$. We observe in Table 1 that models without $F_2$ fail at estimating $w$, forcing $F_1^u$ to capture the whole dynamics. Unlike our estimations of $S$ showed in appendix Figure 2, they thus also fail at estimating an interpretable source term $S$. Finally, data agnostic algorithms are less performing for long term forecasts than informed neural models, confirming that providing knowledge in a data-driven forward model brings stability in the forecasts.

## 5 Conclusion

We propose to bridge PDE-specific numerical scheme with deep networks to solve forward and inverse problem for partially observed dynamics. We empirically show that regularizing time varying unobserved states helps solving both the forward problem and the inverse estimation. Further studies should include more thorough experiments evidencing different dynamics. Theoretical considerations on the proposed regularization will also be investigated.

## Broader Impact

The cooperation between physical models and deep learning is an emerging topic with interesting applications in the climate modeling community. For example, this cooperation could be coupled with data assimilation techniques allowing for better understanding of Earth's climate, crucial for the estimation of the impact of climate change.

## Acknowledgments and Disclosure of Funding

We would like to thank all members of the MLIA team from the LIP6 laboratory of Sorbonne Université for helpful discussions and comments. We acknowledge financial support from the LOCUST ANR project (ANR-15-CE23-0027) and the European Union's Horizon 2020 research and innovation program under grant agreement 825619 (AI4EU). This study has been conducted thanks to Natl60 data provided by MEOM research team.

## References

[1] David L. T. Anderson and Jürgen Willebrand, editors. *Tracer Inverse Problems*, pages 1–77. Springer Netherlands, Dordrecht, 1989.

[2] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019.

[3] Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Learning the spatio-temporal dynamics of physical processes from partial observations. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3232–3236, 2020.

[4] Guido Boffetta, G Lacorata, G Redaelli, and A Vulpiani. Detecting barriers to transport: a review of different techniques. *Physica D: Nonlinear Phenomena*, 159(1-2):58–70, 2001.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[6] Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124009, 2019.

[7] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1174–1183, Stockholmsmässan, Stockholm, Sweden, July 2018. PMLR.

[8] Michail Diamantakis. The semi-lagrangian technique in atmospheric modelling: current status and future challenges. In *Seminar on Recent Developments in Numerical Methods for Atmosphere and Ocean Modelling, 2-5 September 2013*, pages 183–200, Shinfield Park, Reading, 2014. ECMWF, ECMWF.

[9] Claude Frankignoul. Sea surface temperature anomalies, planetary waves, and air-sea feedback in the middle latitudes. *Reviews of geophysics*, 23(4):357–390, 1985.

[10] Lucile Gaultier, Jacques Verron, Jean-Michel Brankart, Olivier Titaud, and Pierre Brasseur. On the inversion of submesoscale tracer fields to estimate the surface ocean circulation. *Journal of Marine Systems*, 126:33–42, 2013.

[11] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15353–15363, 2019.

[12] Vincent Le Guen, Yuan Yin, Jérémie Dona, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting, 2020.

[13] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, dec 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[16] Skander Karkar, Ibrahhim Ayed, Emmanuel de Bézenac, and Patrick Gallinari. A principle of least action for the training of neural networks, 2020.

[17] Zhen Li and Zuoqiang Shi. Deep residual learning and pdes on manifold. *CoRR*, abs/1708.05115, 2017.

[18] Ori Linial and Uri Shalit. Generative {ode} modeling with known unknowns. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[19] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216, 2018.

[20] Gurvan Madec. *NEMO ocean engine*. Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, 2008.

[21] Viraj Mehta, Ian Char, Willie Neiswanger, Youngseog Chung, Andrew Oakleigh Nelson, Mark D Boyer, Egemen Kolemen, and Jeff Schneider. Neural dynamical systems: Balancing structure and flexibility in physical prediction, 2020.

[22] Andrew M Moore. Data assimilation in a quasi-geostrophic open-ocean model of the gulf stream region using the adjoint method. *Journal of Physical Oceanography*, 21(3):398–427, 1991.

[23] Finbarr O'Sullivan. A statistical perspective on ill-posed inverse problems. *Statistical science*, pages 502–518, 1986.

[24] R.-E. Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 11 2006.

[25] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[26] Maziar Raissi, Hessam Babaee, and Peyman Givi. Deep learning of turbulent scalar mixing. *Phys. Rev. Fluids*, 4:124501, Dec 2019.

[27] Maziar Raissi, Paris Perdikaris, and George 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.

[28] Pierre C Sabatier. Past and future of inverse problems. *Journal of Mathematical Physics*, 41(6):4082–4124, 2000.

[29] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting.

In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 802–810. Curran Associates, Inc., 2015.

[30] Andrew Staniforth and Jean Côté. Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review. *Monthly Weather Review*, 119(9):2206–2223, 09 1990.

[31] Peter Toth, Danilo J. Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2020.

[32] Yunbo Wang, Jianjin Zhang, Hongyu Zhu, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

# 6 Appendix

## 6.1 Introduction To Ocean Modeling

Primitive equation models such as NEMO [20] can realistically model the ocean circulation, the current velocity fields and their advection. These models represent the tri-dimensional structure of the ocean but are very costly in computational time and in memory. Simplified models can represent the ocean surface circulation with good accuracy in most cases. For instance, shallow water equations simplifies classical Navier-Stokes equations, integrating over depth, and can be written as:

$$\text{Momentum equation:} \quad \frac{\partial w}{\partial t} + (w.\nabla)w - f \wedge w = -g'\nabla h + D^w + F^w \tag{10}$$

$$\text{Tracer advection:} \quad \frac{\partial \psi}{\partial t} + \nabla.(\psi w) = D^\psi + F^\psi \tag{11}$$

where $w$ is the horizontal velocity vector, $f$ the Coriolis parameter, $h$ the depth of the surface layer obtained from sea surface height observations, $g'$ the reduced gravity which takes the stratification in density of the ocean into account such that $g' \approx g10^{-3}$, $\psi$ a tracer concentration. The mixture terms, referred to as $D^{\psi/w}$ and $F^{\psi/w}$, are not known.

Note that in the context of the presented work: $\psi = X^{obs}$ and represents the sea surface temperature.

## 6.2 Simulation Details

Now we have introduced elements of ocean dynamics, we describe our datasets.

We access monthly data over a year of sea ocean surface temperature of the North Atlantic observations model denoted NATL60 resulting from NEMO model [20] [2]. The dataset covers a 2300km $\times$ 2560km zone at 60°(1.5km) resolution, in the North Atlantic Ocean (Gulf Stream).

Both datasets considered in the paper follow the same equations approximating the tracer equation (Equation (11)):

$$\frac{\partial \psi}{\partial t} + \nabla.(\psi w) = S(w, l, t) \tag{12}$$

The first case corresponds to $S = 0$, i.e a pure advection, while our second investigated case explores the case where $S$ is derived from [9].

### 6.2.1 Building a Velocity Field

To simulate a transport setting represented by Equation (12), we first build a velocity field $w$.

**Intuition from Navier Stokes** For slow movements (that is of characteristic time superior to a day and of spatial dimension superior to 20km) forcings are omitted and incompressibility is assumed, i.e the turbulent terms are null: $(w.\nabla)w = 0$. The momentum equation Equation (10) can be simplified:

$$\text{Geostrophic equation:} \quad f \wedge w = -g'\nabla h \tag{13}$$

When projected onto x and y axis, 13 becomes

$$-fv = -g'\frac{\partial h}{\partial x}, \qquad fu = -g'\frac{\partial h}{\partial y} \tag{14}$$

**Simulation** As introduced above in Equation (14), we restrict our attention to incompressible velocity field, automatically satisfied by introducing a stream function $\mathcal{H} = \mathcal{H}(l, t)$ and defining $w$ through partial derivatives of $\mathcal{H}$ as $w = (\frac{\partial \mathcal{H}}{\partial y}, -\frac{\partial \mathcal{H}}{\partial x})$. Following [4], we define $\mathcal{H}$ as:

$$\mathcal{H}(x, y, t) = -\tanh[\frac{y - B(t) \times \cos kx}{\sqrt{1 + k^2 B(t)^2 \times sin^2 kx}}] + cy, \tag{15}$$

Note that $B$ varies periodically with time according to $B = B_0 + \epsilon \cos(\omega t + \phi)$. We compute 10 different velocity fields with parameters $B_0 = 1.2, k = 0.84, c = 0.03, \omega = 0.4, \epsilon = 0.8, \phi = 0.5$.

[2] Details available at : https://meom-group.github.io/swot-natl60/access-data.html

### 6.2.2 Datasets

To generate our data, we sample randomly 800 images of size $64 \times 64$ in NATL60 dataset, each image representing an initial condition. Among these 800 images, 80 are reserved for validation. 200 other $64 \times 64$ images are sampled for test. The images will be used as initial conditions for our numerical scheme to integrate Equation (12) using the previously computed $w$ across 65 timesteps thanks to a semi-Lagrangian scheme (see Section 6.3).

**Advection Dataset** For the purely advective dataset, $S$ is set to 0. Details on the semi-Lagrangian advection in this case can be found in [30].

Finally, to make our numerical integration well posed, we need border conditions. We impose symmetrical border conditions implying that what comes out the left part re-enters at the right, and reciprocally. We also impose velocity to be null on both top and bottom parts of the image.

**Source Dataset** For this dataset $S$ is non-null, and its precise description can be found in [9]. Simply put, this source term is a non linear transformation of $(u, v)$ multiplied by the difference between the ocean temperature and a reference temperature:

$$S(w, \psi, t) = w_e \times (\psi(t) - T_e),$$

where $w_e$ is 0 where $\frac{\partial \mathcal{H}}{\partial t}$ is below a small value ($\epsilon = 10^{-4}$) and 1 otherwise, and for each sequence $T_e$ is the sequence mean image (computed without source).

Same border conditions as before are assumed.

## 6.3 Semi-Lagrangian Scheme

Unlike Eulerian scheme, relying on time discretization of the derivative, the semi Lagrangian scheme relies on the constancy of the solution of a PDE along *a characteristic curve*. Consider a solution to the advection equation, i.e. Equation (12) with $S = 0$. The method of characteristics consists in exhibiting curves $(x(s), t(s))$ along which the derivative of the solution $\psi$ is simple, i.e $\frac{\partial \psi}{\partial s}(x(s), t(s)) = 0$. For a 1D constant advection scheme, computations leads to:

$$\begin{aligned} \frac{dt}{ds} &= 1 \implies s = t \\ \frac{dx}{ds} &= w \implies x = x_0 + wt \end{aligned}$$

giving therefore, $\psi(x, t) = \psi_0(x - wt)$, linking the value of the solution at all time to its initial condition. Therefore from a single observation at $t_0$, it suffices to estimate the original departure points $x_0 - wt$ to infer the prediction at $t$.

However, when $w$ is not constant in time, the method remains doable, not along characteristic *lines* : $(x_0 + wt)$, but along characteristic *curves* which are given by:

$$\begin{aligned} \frac{dt}{ds} &= 1 \implies s = t \\ \frac{dx}{ds} &= w(x, t) \end{aligned} \tag{16}$$

Note that a great deal in the semi-Lagrangian literature involves solving correctly Equation (16). The mid-point integration rule is a classical method for handling this problem leading to a fixed point algorithm. In our case, we used a mid-point integration scheme rule with one iteration that we found sufficient in practice. Further developments can be found for example in [8].

## 6.4 Model Specifications

We used Python 3.8 and Pytorch 1.5 to implement our model trained on Nvidia GPU with CUDA 10.1.

### 6.4.1 Prior Knowledge and Assumptions

For both datasets: we use as $F_1^k$ the same numerical semi-Lagrangian advection scheme than used for data simulation. It remains differentiable with relation to $w$ allowing optimization of $G_\theta$.

Finally, relying on the momentum equation of Equation (10), we infer that the evolution of $w$ is independent from the evolution of $\psi$, thus we make $F_2$ takes as input only $\hat{w}$, previously estimated from a sequence of $\psi$. Using notations from section 3, we consider $F_2$ such that: $F_2(X_t) = F_2(Y_t)$.

### 6.4.2 Implementation Details for Null Source

**Architectures** $F_1^u$ is a convolutional Residual Network with 2 residual blocks. The input are first downscaled using two layers of strided convolutions. Each residual block has 128 channels, following the implementation of [15].

$F_2$ is a convolutional Residual Network with 2 residual blocks. The input are first downscaled using two layers of strided convolutions. Each residual block has 128 channels, following the implementation of [15].

$G_\theta$ is a U-net with at most 512 latent channels also following the implementation of [15].

**HyperParameters** The learning rate for all algorithms and baselines is $lr = 10^{-5}$ using Adam optimizer with $\beta = (0.9, 0.999)$, with batch size 32.

The number of input frames for $G$ is 4, i.e in Equation (2) $k = 4$. The number of predicted time steps $T$, see Equation (6), is 6.

In practice we set $\lambda_{X^{obs}} = 1$, and specify another multiplicative hyperparameter $\lambda_Y$ so that $\lambda_Y = 0.01$

**Baselines** In this setting, for Neural ODE baseline, $G$ has similar architectures, estimating 2 hidden channels (2 for $w$). The dynamical network, trained using a 3-layer convolutional networks, with 64 hidden channels, is integrated using RK4 scheme available from `https://github.com/rtqichen/torchdiffeq`. The optimization hyperparameters $\lambda_Y$, $\lambda_{X^{obs}}$ are the same.

For experiments with minimisation of $\|f_1^u\|$, a simple cost is added to the original cost function Equation (8): $\mathcal{L}_{f_1^u} = 0.01 \times ||F_1^u(X^{obs}_{(t-5:t)})||_2$

### 6.4.3 Implementation Details for Frankignoul Source

**Architectures** $F_1^u$ is a U-net with at most 512 latent channels, following the implementation of [15].

$F_2$ is a convolutional Residual Network with 2 residual blocks. The input are first downscaled using two layers of strided convolutions. Each residual block has 128 channels, following the implementation of [15].

$G_\theta$ is a U-net with at most 512 latent channels also following the implementation of [15].

For experiments with minimisation of $\|f_1^u\|$, a simple cost is added to the original cost function Equation (8): $\mathcal{J}_{f_1^u} = 0.01 \times ||F_1^u(X^{obs}_{(t-k:t)})||_2$

**HyperParameters** The learning rate for all algorithms and baselines is $10^{-4}$ using Adam optimizer with $\beta = (0.9, 0.999)$, with batch size 32.

The number of input frames for $G$ and $F_1^u$ is 4, i.e in Equation (2) $k = 4$. The number of predicted time steps $T$, see Equation (6), is 6.

In practice we set $\lambda_{X^{obs}} = 1$, and specify another multiplicative hyperparameter $\lambda_Y$ so that $\lambda_Y = 0.1$

**Baselines** In this setting for Neural ODE baseline, $G$ has similar architectures, estimating 3 hidden channels (2 for $w$, one for the source $S$ and one for the source). The dynamical network, trained using a 3-layer convolutional networks, with 64 hidden channels, is integrated using RK4 scheme. The optimization hyperparameters $\lambda_Y$, $\lambda_{X^{obs}}$ are the same.

For experiments with minimisation of $\|f_1^u\|$, a simple cost is added to the original cost function Equation (8): $\mathcal{J}_{f_1^u} = 0.01 \times ||F_1^u(X^{obs}_{(t-k:t)})||_2$

### 6.5 Additional Results

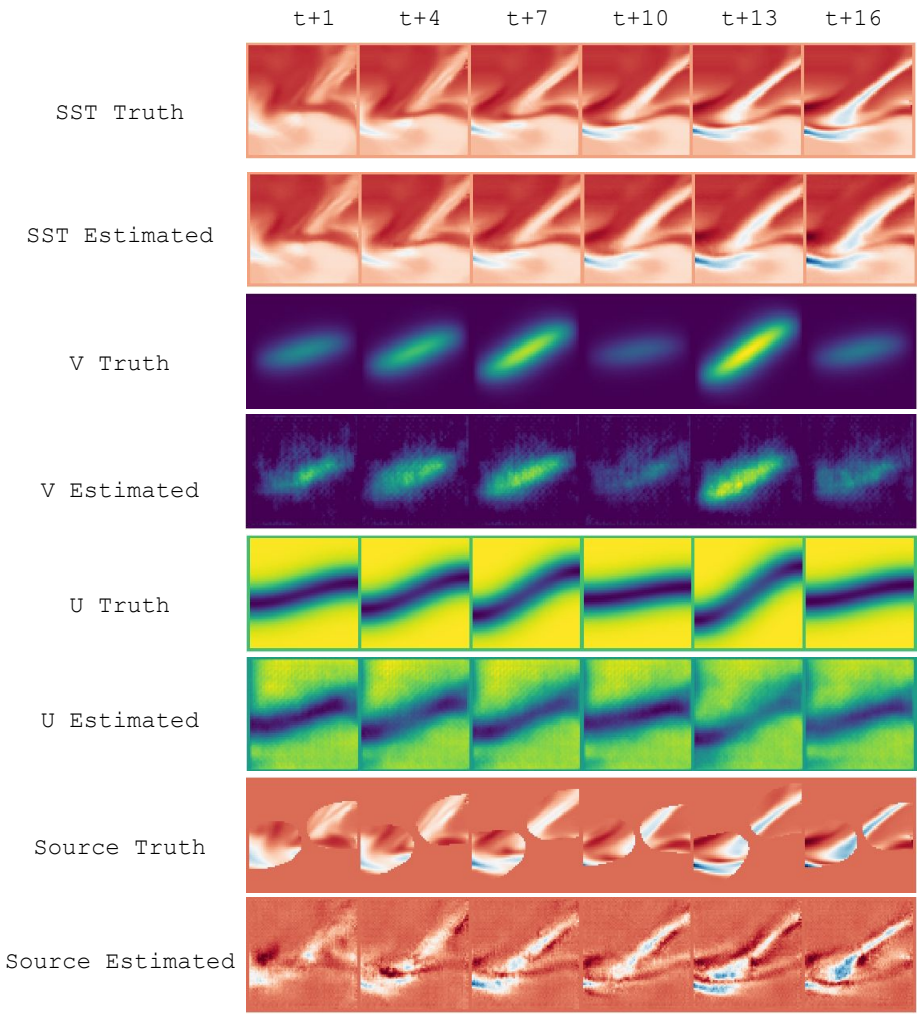


Figure 2: Sequence of estimations on $\psi$, $w = (u, v)$, $S$ for the Dataset with non null physical source