ESAIM: PROCEEDINGS AND SURVEYS, June 2025, Vol. 81, p. 2-15
Didier Auroux, Martin Campos Pinto, Bruno Després, Victorita Dolean, Stéphane Lanteri, Victor Michel-Dansac

A DYNAMICAL NEURAL GALERKIN SCHEME FOR FILTERING PROBLEMS

Joubine Aghili[1,2], Joy Zialesi Atokple[3], Marie Billaud-Friess[4], Guillaume Garnier[5], Olga Mula[6] and Norbert Tognon[7]

Abstract. This paper considers the filtering problem which consists in reconstructing the state of a dynamical system with partial observations coming from sensor measurements, and the knowledge that the dynamics are governed by a physical PDE model with unknown parameters. We present a filtering algorithm where the reconstruction of the dynamics is done with neural network approximations whose weights are dynamically updated using observational data. In addition to the estimate of the state, we also obtain time-dependent parameter estimations of the PDE parameters governing the observed evolution. We illustrate the behavior of the method in a one-dimensional KdV equation involving the transport of solutions with local support. Our numerical investigation reveals the importance of the location and number of the observations. In particular, it suggests to consider dynamical sensor placement.

1. Introduction

Data assimilation of dynamical systems is a very active research topic, with many challenging open problems especially in cases where strong transport effects are present. This work presents a filtering algorithm based on neural network approximations whose weights dynamically evolve in time.

1.1. The filtering problem

We define filtering as the following task. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a spatial domain, and let V a Hilbert space of functions over $\mathcal{X}$. In this work, we assume that $V \subset \mathcal{C}(\mathcal{X})$ for simplicity (a more general theoretical framework will be developed in a subsequent contribution). The filtering task is to recover at each time $t \geq 0$ the unknown state $u \in \mathcal{C}^1(\mathbb{R}_+, V)$ from m observations

$$z_i(t) = u(t, x_i), \quad 1 \leq i \leq m,$$

where $x_i \in \mathcal{X}$ are observation points.

[1] IRMA, Université de Strasbourg, CNRS UMR 7501, 7 rue René Descartes, 67084 Strasbourg, France
[2] Université de Strasbourg, CNRS, Inria, IRMA, F-67000 Strasbourg, France
[3] Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
[4] Centrale Nantes, Nantes Université, LMJL UMR CNRS 6629, Nantes, France
[5] Sorbonne Université, Inria, Laboratoire Jacques-Louis Lions (LJLL), 75005 Paris, France
[6] Eindhoven University of Technology, Den Dolech 2, P.O. Box 513, 5600 Eindhoven, Netherlands
[7] INRIA-Paris Project Team ANGE, Sorbonne Université (LJLL), 2 Rue Simone Iff, 75012 Paris, France

Article published online by EDP Sciences and available at https://www.esaim-proc.org or https://doi.org/10.1051/proc/202581002

Recovering u only from the observations is a very ill-posed problem so we add an a priori assumption on the dynamics of the system: we assume that u is the solution of an evolution problem of the form

$$\left\{\begin{array}{rcll} \partial_t u(t,x,\xi) & = & f\left(t,x,u(t,x,\xi),\xi\right), & (t,x,\xi)\in\mathbb{R}_+^*\times\mathcal{X}\times\mathcal{P}, \\ u(0,x,\xi) & = & u_0(x,\xi) & x\in\mathcal{X}, \end{array}\right. \tag{1}$$

with initial condition $u_0 : \mathcal{X}\times\mathcal{P}\to\mathbb{R}$, and $f : \mathbb{R}_+^*\times\mathcal{X}\times\mathbb{R}\times\mathcal{P}\to\mathbb{R}$ being a nonlinear differential operator. The vector $\xi = (\xi_1,\dots,\xi_p)$ is a vector of scalar parameters ranging in a compact domain $\mathcal{P}\subset\mathbb{R}^p$. These parameters can represent physical quantities such as velocity, diffusivity, viscosity, source terms, or initial conditions. In full generality, ξ could depend on time. However, in the following we will rather think of it as a constant. We assume that problem (1) is well posed in the sense that, for every $\xi\in\mathcal{P}$ the problem admits a unique solution $u = u(\xi)\in\mathcal{C}^1(\mathbb{R}_+,V)$. In this form, we will regard u as a parameter to solution map $\xi\in\mathcal{P}\to u(\xi)\in\mathcal{C}^1(\mathbb{R}_+,V)$ so that $u(\xi)$ is a function of t and x. To simplify notation, we will sometimes write $u(\xi)(t,x) = u(t,x,\xi)$.

In a filtering problem, we assume that there is a parameter $\xi^\dagger$ corresponding to the observed evolution. This implies that we are searching to approximate $u(\xi^\dagger)(t,x)$ for all $t\geq 0$ and $x\in\mathcal{X}$. However, $\xi^\dagger$ is assumed to be unknown so the dynamics cannot be recovered with traditional forward solvers that compute a solution to problem (1) with $\xi = \xi^\dagger$. Instead, our data are the observations $z(t)$ or its first derivatives together with the assumption that the dynamics can be expressed in the form of eq. (1).

1.2. **Contribution: dynamical reconstruction with neural network approximations**

In this work, we build a filtering algorithm where, for each $t\geq 0$, $u(\xi^\dagger)(t,\cdot)$ is approximated by a feed-forward neural network model. Expressed in more concrete terms: viewing $u(\xi^\dagger)(t,\cdot)$ as a function from $\mathcal{X}$ to $\mathbb{R}$, we consider a set $\mathcal{U}$ of admissible functions from $\mathcal{X}$ to $\mathbb{R}$. The functions of $\mathcal{U}$ are generated by parametrized functions of the form

$$U : \mathcal{X}\times\Theta\to\mathbb{R} \tag{2}$$

$$(x,\theta)\mapsto U(x,\theta) \tag{3}$$

where θ is a set of parameters belonging to a space Θ. The set $\mathcal{U}$ is thus defined as

$$\mathcal{U} \coloneqq \{U(\cdot,\theta)\ :\ \theta\in\Theta\}.$$

In this work we focus on the particular case where $\mathcal{U}$ is generated by feed-forward neural networks with a given architecture. For given $n\in\mathbb{N}$ and $s\in\mathbb{N}$, we will consider $\theta = (\theta_1,\dots,\theta_n)\in\Theta = (\mathbb{R}^s)^n\cong\mathbb{R}^{ns}$ be our set of parameters with $\theta_i = (c_i,b_i,w_i)\in\mathbb{R}^s$. As a simple example, a shallow neural networks of width n with inputs in $\mathcal{X}\subset\mathbb{R}^d$ and outputs in $\mathbb{R}$ is the model $U : \mathcal{X}\times\Theta\to\mathbb{R}$ defined as

$$U(x,\theta) \coloneqq \sum_{i=1}^{n} c_i\,\phi(x,w_i,b_i), \tag{4}$$

where ϕ is a smooth activation function, and w and b represent weights and biases respectively. One example for ϕ is obtained by taking $w_i\in\mathbb{R}^2$ and $b_i\in\mathbb{R}$ and a ReLU activation unit, resulting to a real-valued mapping $\phi = \phi^{\mathrm{ReLU}}$ defined on $\mathcal{X}\times\mathbb{R}^d\times\mathbb{R}$ as

$$\phi^{\mathrm{ReLU}}(x,w,b) \coloneqq \max(0,w^T x + b),\quad \forall x\in\mathbb{R}^d.$$

Deeper networks could be defined following similar lines, see Section 4.2.

The rationale, and originality of our approach lies in the fact that the parameters θ of our reconstruction will dynamically evolve in time, that is, for every $t\geq 0$, we will find weights $\theta(t)\in\Theta$ such that $U(\cdot,\theta(t))$

approximates $u(\xi^\dagger)(t,\cdot)$, at best given our available information $z(t)$, and the form (1) of the dynamics. By allowing the parameter θ to be time dependent, our strategy allows for reconstruction as time advances.

One may wonder about the benefit of this dynamical, local in time approach compared to a more global, time-space point of view. In this other point of view, we would typically work with neural networks taking space $x \in \mathcal{X}$ and time $t \in \mathbb{R}_+$ as input parameters, and we would search for constant weights $\theta \in \Theta$ in a way that $u(\xi^\dagger)(t,x)$ is approximated well by $U((t,x),\theta)$ for all $(t,x) \in \mathbb{R}_+ \times \mathcal{X}$ (or a closed time interval $[0,T]$). This time-space approach is less adapted to the filtering problem for several reasons:

(1) We need to wait until the end of the time interval to train the network, and reconstruct in the full time interval.
(2) To reconstruct at a single time t, we expect to need fewer parameters $\theta(t)$ than in a global time-space approach. This point was confirmed in [3] in the case of forward reduced modeling with linear spaces (instead of nonlinear spaces such as neural networks). It was observed that the approximation with dynamically evolving linear spaces was more accurate than a time-space approach involving subspaces with larger dimensions.

1.3. **Connections with earlier works**

Our work connects with a relatively recent research trend aiming at solving inverse problems with neural networks. Probably the field of image analysis was the one of the first to start considering these approximation classes for inversion tasks. In this field, neural networks have produced state of the art results in countless applications and nowadays one could say that they have become the method of choice (see, e.g., [6]).

The use of neural networks to address inverse problems involving PDE physical models is more recent. It is currently attracting a lot of attention, and our work belongs to this trend. Among the relevant contributions that have recently appeared, we may mention [1,11,12,15]. Our method takes inspiration from the dynamical Neural Galerkin Scheme (NGS) introduced in [4], which uses the same idea for forward PDE simulation. The NGS is actually an internal building block of our strategy as we explain later on. Compared to most existing works, the main novelty of the NGS introduced [4] is that it can be seen as a dynamical (nonlinear) approximation approach. This means that NGS can be derived from some, local in time, Dirac-Frenkel variational principle (see, e.g., [9]), related to the set $\mathcal{U}$ of parametrized functions. In practice, it leads to a system of ODEs for dynamical update of the neural network's weights. Such setting for neural networks particularly extends previous several works done within the context of dynamical low-rank approximation in tensor subsets for forward simulation of PDEs (see, e.g., [7, 10, 14, 18]).

Using dynamical approximations for inverse problems such as filtering is a rather unexplored idea. To the best of our knowledge, only [5, 8, 16] and [19, Chapter 7] have considered such techniques for linear Kalman filtering. The approximation classes that were used in these works are linear subspaces that are also dynamically updated. The other only existing work using dynamical linear subspaces is [13]. It comes with additional ideas on structure preservation that are illustrated through the use of symplectic dynamical low rank spaces for Hamiltonian dynamics. [13] is also the only existing work that considers an algorithm to dynamically update the location of the sensors, and which is based on a rigorous analysis of reconstruction errors. The idea of dynamical sensor placement is important for filtering dynamics involving strong transport effects like the ones we aim to consider here. Compared to these works, the main novelty of this paper is the use of the class of feed-forward neural networks instead of dynamical linear subspaces. Since it is very nonlinear in nature, we expected it to be particularly useful for transport dominated dynamics.

1.4. **Organization of the paper**

Section 2 summarizes the NGS of [4] for forward PDE simulation. Section 3 presents our dynamical filtering algorithm for filtering. It involves the forward NGS as a building block, and provides both a reconstruction of the state u, and a time dependent estimation of the PDE parameters $\xi^\dagger$ associated to the observed trajectory. Finally, Section 4 illustrates the behavior of the method in a one-dimensional KdV equation. The study reveals that the method can provide good quality reconstructions provided that the sensors are located inside the

support of the solution. Since the location where the solution is supported evolves in time, this means that the locations of observations need to move, and follow the solution dynamics.

2. The Neural Galerkin Scheme for forward problems

Like for many algorithms for inverse problems and data assimilation, our filtering approach requires to evaluate solutions of eq. (1) for different parameter values $\xi \in \mathcal{P}$. The task of computing the parameter-to-solution map $\xi \mapsto u(\xi) \in \mathcal{C}^1(\mathbb{R}_+, V)$ is usually called the forward problem, as opposed to the inverse problem of estimating u from data observations, which corresponds to the mapping $A : z(t) \mapsto A(z(t)) \approx u(\xi^\dagger)(t, \cdot) \in V$. Therefore, as a preparatory step, we explain the scheme that we use for the forward problem, which will serve as a building block for our filtering algorithm. We rely on the NGS that was introduced in [4] and which involves a dynamical neural network approximation.

2.1. **The scheme**

We carry out the explanation for a fixed, given parameter $\xi \in \mathcal{P}$ so, to avoid overloading notation, we will omit it in this section. Consequently, we focus on the problem of approximating $u \in \mathcal{C}^1(\mathbb{R}_+, V)$ solution to

$$\left\{ \begin{array}{rcll} \partial_t u(t,x) & = & f\left(t, x, u(t,x)\right), & (t,x) \in \mathbb{R}_+^* \times \mathcal{X}, \\ u(0,x) & = & u_0(x) & x \in \mathcal{X}. \end{array} \right. \tag{5}$$

For every $t \geq 0$, we seek for an approximation of $u(t, \cdot) \in V$ of the form $\mathrm{U}(\cdot, \theta(t)) \in \mathcal{U} \subset V$.

The NGS from [4] consists in deriving an evolution equation for the weights θ by using the strong residual of eq. (5). The main steps are as follows. Assuming that U is differentiable w.r.t. θ, and θ differentiable w.r.t. t, by the chain rule we have

$$\partial_t \mathrm{U}(x, \theta(t)) = \nabla_\theta \mathrm{U}(x, \theta(t))^T \dot{\theta}(t), \tag{6}$$

where $\nabla_\theta = (\partial_{\theta_1}, \ldots, \partial_{\theta_n})^T \in \mathbb{R}^n$ stands for the gradient w.r.t. θ. Thus, for a given $t > 0$, the strong residual of the equation at a point $x \in \mathcal{X}$ is the function $r_t : \Theta \times \dot{\Theta} \times \mathcal{X} \to \mathbb{R}$ defined as

$$r_t(\theta, \eta, x) \coloneqq \nabla_\theta \mathrm{U}(x, \theta)^T \eta - f(t, x, \mathrm{U}(x, \theta)), \quad \forall (\theta, \eta, x) \in \Theta \times \dot{\Theta} \times \mathcal{X} \tag{7}$$

where $\dot{\Theta}$ is the space of time derivatives of Θ. The global residual $R_t : \Theta \times \dot{\Theta} \to \mathbb{R}$ is obtained by computing the squared $L^2_\nu(\mathcal{X})$ norm of $x \mapsto r_t(\theta, \eta, x)$, that is,

$$R_t(\theta, \eta) \coloneqq \frac{1}{2} \int_{\mathcal{X}} |\nabla_\theta \mathrm{U}(x, \theta)^T \eta - f(t, x, \mathrm{U}(x, \theta))|^2 \nu(\mathrm{d}x) \tag{8}$$

$$= \frac{1}{2} \|\nabla_\theta \mathrm{U}(\theta, \cdot)^T \eta - f(t, \cdot, \mathrm{U}(\cdot, \theta))\|^2, \quad \forall (\theta, \eta) \in \Theta \times \dot{\Theta}. \tag{9}$$

In the above formula, $\nu \in \mathcal{P}(\mathcal{X})$ is a measure over $\mathcal{X}$ and $\| \cdot \|$ is the standard $L^2_\nu(\mathcal{X})$ norm associated to the usual scalar product $\langle \cdot, \cdot \rangle$. In this work, we simply work with the Lebesgue measure $\nu(\mathrm{d}x) = \mathrm{d}x$, and we leave a more in-depth investigation on the optimal choice of ν for future work.

The NGS seeks to define the curve $\theta : \mathbb{R}_+ \mapsto \Theta$ in such a way that its velocity $\dot{\theta}(t)$ minimizes the global residual (9) at every $t \in \mathbb{R}_+^*$. This leads to the variational problem

$$\dot{\theta}(t) \in \underset{\eta \in \dot{\Theta}}{\arg\min}\, R_t(\theta(t), \eta), \quad \forall t > 0. \tag{10}$$

This local in time optimization problem corresponds to the so-called Dirac-Frenkel variational principle [9]. At the initial time $t = 0$, we find $\theta(0) := \theta_0$ by solving the least-squares problem

$$\theta_0 \in \arg\min_{\theta\in\Theta} \frac{1}{2}\int_{\mathcal{X}} |\mathrm{U}(x,\theta) - u_0(x)|^2 \mathrm{d}x. \tag{11}$$

The necessary optimality conditions of problem eq. (10) lead to the so-called Euler equations characterized by

$$\nabla_\eta R_t(\theta(t), \dot\theta(t)) = 0, \quad \forall t > 0. \tag{12}$$

Since

$$\nabla_\eta R_t(\theta,\eta) = \int_{\mathcal{X}} \nabla_\theta \mathrm{U}(x,\theta)\left(\nabla_\theta \mathrm{U}(x,\theta)^T\eta - f(t,x,\mathrm{U}(x,\theta))\right)\mathrm{d}x,$$

we can write (12) as

$$\int_{\mathcal{X}} \nabla_\theta \mathrm{U}(x,\theta(t))\left(\nabla_\theta \mathrm{U}(x,\theta(t))^T\dot\theta(t) - f(t,x,\mathrm{U}(x,\theta(t)))\right)\mathrm{d}x = 0$$

Note that the above relation can equivalently be written as

$$\left\langle \nabla_\theta \mathrm{U}(\cdot,\theta(t)), r_t(\theta(t),\dot\theta(t),\cdot)\right\rangle = 0 \tag{13}$$

The resulting equation (13) can be formally interpreted as $L^2(\mathcal{X})$-projection, for each time t, on the tangent space of the "differential manifold" $\mathcal{U}$ at $\mathrm{U}(\cdot,\theta(t))$, which is the linear space spanned by the partial derivatives with respect to the θ_i of the non linear map U (for more detailed discussion on this topic, see, e.g., [2, Section 2.1]).

Defining $M : \Theta \to \mathbb{R}^{n\times n}$ and $F : \mathbb{R}_+^* \times \Theta \to \mathbb{R}^n$ as

$$\begin{aligned} M(\theta) &\coloneqq \int_{\mathcal{X}} \nabla_\theta \mathrm{U}(x,\theta)\nabla_\theta \mathrm{U}(x,\theta)^T \mathrm{d}x, \\ F(t,\theta) &\coloneqq \int_{\mathcal{X}} \nabla_\theta \mathrm{U}(x,\theta) f(t,x,\mathrm{U}(x,\theta))\mathrm{d}x, \end{aligned} \tag{14}$$

then eq. (12) leads to the following system of Ordinary Differential Equations (ODEs) for the parameters θ

$$\left\{ \begin{array}{rcl} M(\theta(t))\dot\theta(t) &=& F(t,\theta(t)), \quad t > 0 \\ \theta(0) &=& \theta_0. \end{array}\right. \tag{15}$$

Remark 1. *If the initial problem given in eq.* (5) *comes with boundary conditions, there exists several possibilities to treat them at the discrete level in eq.* (15)*. One can enforce them in a strong way within the model* U *[4, Section 4.4], or weakly by adding conditions in the definition of* U*, see [4, Section 3.2].*

2.2. **Practical implementation**

To implement the ODE system (15) in practice, we approximate the integrals over $\mathcal{X}$ with Monte-Carlo quadrature (MC), and we use classical time integration schemes. For the sake of simplicity, we present in this section a simple forward Euler time integration, but any other time integration scheme could be considered. In fact, in our implementation, we have used an explicit Runge-Kutta scheme of order 4.

Let $0 = t_0 < t_1 < t_2 < \cdots < t_k < \ldots$ with (possibly non uniform) time steps $\delta t_k \coloneqq t_{k+1} - t_k > 0$, and suppose we compute at each t_k an approximation $\theta_k \approx \theta(t_k)$ using a forward Euler scheme. Thus, starting from θ_0 satisfying eq. (11), we are led to the scheme

$$\tilde M(\theta_k)\theta_{k+1} = \tilde M(\theta_k)\theta_k + \delta t_k \tilde F(t_{k+1},\theta_k), \quad k \geq 0, \tag{16}$$

where, for every $(t,\theta) \in \mathbb{R}_+ \times \Theta$, $\tilde{M}(\theta)$ and $\tilde{F}(t,\theta)$ are the approximations of $M(\theta)$ and $F(t,\theta)$ by MC integration involving J independent uniformly distributed random samples $\{\bar{x}_j\}_{j=1}^J$ over $\mathcal{X}$, we have

$$\begin{aligned} \tilde{M}(\theta) &\coloneqq \frac{1}{J}\sum_{j=1}^{J} \nabla_\theta \mathrm{U}(\bar{x}_j,\theta)\nabla_\theta \mathrm{U}(\bar{x}_j,\theta)^T, \\ \tilde{F}(t,\theta) &\coloneqq \frac{1}{J}\sum_{j=1}^{J} \nabla_\theta \mathrm{U}(\bar{x}_j,\theta) f(t,\bar{x}_j,\mathrm{U}(\bar{x}_j,\theta)). \end{aligned} \tag{17}$$

In practice, $\tilde{M}_k \coloneqq \tilde{M}(\theta_k)$ is a $n \times n$ matrix of the form $\tilde{M}_k = \frac{1}{J}\boldsymbol{V}\boldsymbol{V}^T$. Here, $\boldsymbol{V}$ is a matrix in $\mathbb{R}^{n\times J}$ whose j-th columns corresponds to $\nabla_\theta \mathrm{U}(\bar{x}_j,\theta)$. Similarly, $\tilde{F}_k \coloneqq \frac{1}{J}\boldsymbol{V}\boldsymbol{F}$ is a vector of size n where $\boldsymbol{F} = (F_j)_{1\le 1\le J} \in \mathbb{R}^J$ has entries $F_j = f(t_{k+1},\bar{x}_j,\mathrm{U}(\bar{x}_j,\theta_k))$.

In our numerical tests, the matrix $\tilde{M}_k$ quickly becomes singular due to possible linear dependence of the partial derivatives w.r.t. θ_i of U. In this work, we address this problem by regularizing $\tilde{M}_k$ by adding a small perturbation εI to $\tilde{M}_k$ to guarantee invertibility of the matrix. This extra term makes that $\lambda_{\min}(\tilde{M}_k + \varepsilon I) \ge \varepsilon > 0$, which ensures invertibility, and helps to prevent conditionning issues. Alternatively, one can consider the approach described in [2], where the idea is to update only randomly chosen entries of the neural network parameters θ at each step k of eq. (16) to tackle this problem.

Remark 2. *For some problems (e.g., arising from hyperbolic or transport dominated PDEs) uniformly distributed random samples $\{\bar{x}_j\}_{j=1}^J$ over $\mathcal{X}$ may not be efficient to capture dynamical and localized features of the approximation* U*. In that case, one can rather consider adapted sampling strategies to well capture* U *based on the spatial gradient of* U *(see [4] for details).*

We finish this section by recovering our dependence on a parameter ξ in the forward problem. Note that its presence simply implies that all the above quantities become dependent on it: the neural network parameters $\theta(t)$ will depend on it, and so will M, F and their sampled counterparts $\tilde{M}$ and $\tilde{F}$ (including $\tilde{M}_k$ and $\tilde{F}_k$). In what follows, we record this dependence explicitly in the notation when necessary by adding ξ as a superscript.

3. A dynamical filtering algorithm

In this section, we describe our algorithm for dynamical filtering. We recall that the setting here is that we are given observations of the form

$$z_i(t) = u(\xi^\dagger)(t,x_i), \quad 1 \le i \le m,$$

where the $x_i \in \mathcal{X}$ are observation points, and $\xi^\dagger$ is an unknown parameter. Note that we can leverage $z(t)$ to derive information about the velocity of u at these points since

$$\dot{z}_i(t) = \lim_{\delta t \to 0} (z_i(t+\delta t) - z_i(t))/\delta t = \partial_t u(\xi^\dagger)(t,x_i), \quad 1 \le i \le m. \tag{18}$$

In practice, we are given observations at fixed times such that $\dot{z}(t_k)$ can be estimated with finite differences, e.g., $\dot{z}(t_k) \approx (z(t_{k+1}) - z(t_k))/\delta t_k$ for $k > 0$.

Our filtering algorithm requires both $z(t_k)$ and $\dot{z}(t_k)$ (or an approximation of it). Since the parameter $\xi^\dagger \in \mathcal{P}$ is unknown, our approach is based on a joint parameter-state estimation for each time t_k. This procedure corresponds to algorithm 3.1, which proceeds in two steps: a state estimation step and a parameter estimation step, denoted by (**S**) and (**P**) respectively. First, we use the parameter ξ_k to compute θ_k by making one step of the forward problem with the NGS from section 2, namely eq. (16). Second, we use $\dot{z}(t_{k+1})$ and the state $\mathrm{U}(\cdot,\theta_{k+1})$ to estimate the parameter ξ_{k+1} in such a way that we minimize the residual of the equation at the observation points.

Algorithm 3.1 (Main algorithm)**.**

- ***Initial time*** *($k = 0, t_0 = 0$)**:***
  (**S**) *Given $z(0)$, we find*

$$\theta_0 \in \arg\min_{\theta\in\Theta} \left\{ \sum_{i=1}^{m} |z_i(0) - \mathrm{U}(x_i, \theta)|^2 \right\}. \tag{19}$$

  and approximate $u(\xi^\dagger)(0, \cdot) \approx U(\cdot, \theta_0)$. If the favorable case where the initial condition u_0 is fully known, we still need to compute its approximation in $\mathcal{U}$ so we compute

$$\theta_0 \in \arg\min_{\theta\in\Theta} \left\{ \sum_{j=1}^{J} |u_0(\bar{x}_j, \xi^\dagger) - \mathrm{U}(\bar{x}_j, \theta)|^2 \right\}, \tag{20}$$

  (**P**) *Given $\dot{z}(0)$ and θ_0, we estimate the parameter ξ_0 as*

$$\xi_0 \in \arg\min_{\xi\in\mathcal{P}} \left\{ \sum_{i=1}^{m} |f(t_1, x_i, \mathrm{U}(x_i, \theta_0), \xi) - \dot{z}_i(0)|^2 \right\}.$$

- ***Inductive step*** *($t_k \to t_{k+1}$ **for** $k \geq 0$)**:** Assume we have computed $\mathrm{U}(\cdot, \theta_k)$ and ξ_k for time t_k. We now proceed to time t_{k+1}. Using the observations $z(t_k)$ and $\dot{z}(t_k)$, we estimate θ_{k+1} and ξ_{k+1}:*
  (**S**) *We update θ_{k+1} as*

$$\theta_{k+1} = \theta_k + \delta t_k (\tilde{M}_k^{\xi_k})^{-1} \tilde{F}_k^{\xi_k}.$$

  We then approximate $u(\xi^\dagger)(t_{k+1}, \cdot) \approx \mathrm{U}(\cdot, \theta_{k+1})$.
  (**P**) *We find ξ_{k+1} as*

$$\xi_{k+1} \in \arg\min_{\xi\in\mathcal{P}} \left\{ \sum_{i=1}^{m} |f(t_{k+1}, x_i, \mathrm{U}(x_i, \theta_{k+1}), \xi) - \dot{z}_i(t_k)|^2 \right\}. \tag{21}$$

Remark 3. *One could consider more elaborate strategies where the observation locations x_i evolve in time. This feature is important to consider when working with dynamics involving strong transport effects. For these problems, appropriate locations for one time may become completely uninformative for other times due to transport as is shown in [13]. Even if a systematic strategy on how to place observation points goes beyond the scope of this paper, we explore this feature in our numerical experiments by moving the observations points according to "pre-scheduled" dynamics.*

Remark 4. *Instead of estimating θ_k and ξ_k in a two-stage approach, one could alternatively consider a joint estimation of (θ_k, ξ_k). This point is also left for future work.*

4. Numerical experiments

In what follows, we apply the algorithm 3.1 to treat a filtering problem involving the one-dimensional Korteweg de Vries (KdV) equation [17] presented in [4, Section 3.1]. For simplicity, we have assumed that u_0 was fully known in the experiments so we use eq. (20) in our initialization (instead of (19)). An important aspect which we investigate is the impact of the number and location of the observation points in the reconstruction. We explore the potential benefits of dynamically moving the observation points giving us the observations $z_i(t)$ compared to fixed locations.

4.1. **Problem setting**

We consider the KdV equation in one space dimension which reads

$$\begin{cases} \partial_t u(t,x,\xi) &= -\partial_x^3 u(t,x,\xi) - \xi u(t,x,\xi)\partial_x u(t,x,\xi), \\ u(0,x,\xi) &= u_0(x), \end{cases} \tag{22}$$

where the spatial domain and final time are set to $\mathcal{X} = [-10, 20]$ and $T = 4$, respectively. Here, the initial condition u_0 is assumed to be parameter independent. It is defined as in the second case considered in [17, pp. 233]. Since its definition is slightly convoluted, we omit giving the full details here and simply plot its shape in fig. 1 (left plot). The initial condition u_0 models the interaction of two solitons with different amplitudes, one located around $x = -5$ and the other around $x = 1$. Throughout the section, we consider the filtering problem which consists in recovering the solution of eq. (22) with fixed $\xi^\dagger = 6$. The map $(x,t) \mapsto u(\xi^\dagger)(t,x)$ is depicted in fig. 1 (right).

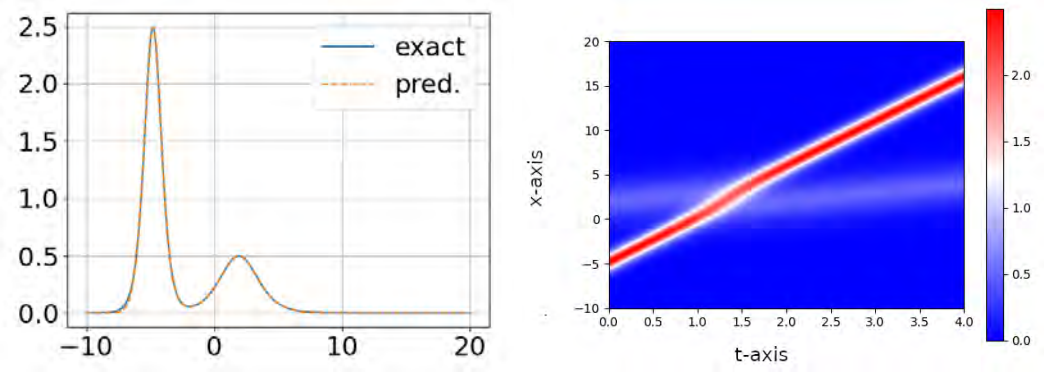

FIGURE 1. **Left:** Initial condition u_0 and its neural network approximation $\mathrm{U}(\cdot, \theta_0)$. **Right:** Solution to the KdV equation for $\xi^\dagger = 6$. This is the solution that we seek to approximate with the filtering algorithm.

4.2. **Implementation details**

Neural network class: We work with a family $\mathcal{U}$ of shallow neural networks where U is of the form (4). Our class consists of a sum of $n = 12$ parametrized Gaussian functions corresponding to the choice

$$\phi(x, w, b) := \exp(-w^2|x - b|^2).$$

We therefore have a total of $n(d + 2) = 36$ parameters to train (i.e., $\theta \in \mathbb{R}^{36}$).

Forward simulation: To solve forward problems, we apply the NGS of section 2 with $\mathcal{U}$. This leads to the nonlinear system of ODEs given in eq. (15), and in our case the matrices have size 36×36. We integrate this system with an explicit fourth-order Runge-Kutta (RK4) discretization to produce solutions to the forward problem. Our reference solution from fig. 1 was obtained in this way.

Filtering: The ODE system (15) is also solved during the inductive step (**S**) of our filtering algorithm 3.1. We apply the RK4 scheme over $K = 1000$ time steps with a fixed time step $\delta t = T/K$. Each step of the scheme requires the evaluations of the matrices $\tilde{M}$ and the right-hand side $\tilde{F}$ at intermediates time-steps. They are estimated using $J = 1000$ points $\{\bar{x}_j\}_{j=1}^J$, uniformly sampled in $\mathcal{X}$. In the step (**P**) of algorithm 3.1, we replace the continuous interval $\mathcal{P}$ with $\mathcal{P}^{\text{train}} \subset \mathcal{P}$ with $\#\mathcal{P}^{\text{train}} = 101$.

To study the conditioning of the matrices $\tilde{M}_k$, we present in fig. 2 the percentage of their nonzero eigenvalues for each time step t_k which are larger than 10^{-6} in absolute value. We observe that 20–25% of the eigenvalues

are above this treshold. This indicates that the matrix should not be considered to be invertible, and having full rank. To overcome the rank deficiency, we regularize $\tilde{M}_k$ by adding a regularization term εI, with $\varepsilon = 10^{-3}$.

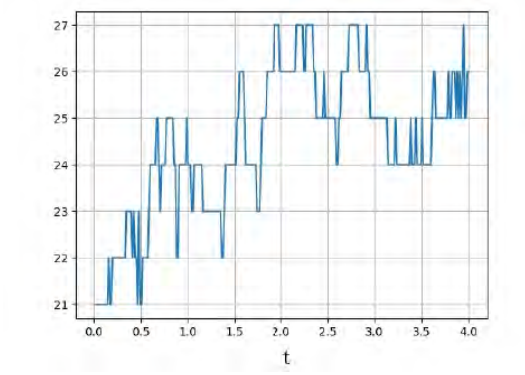

FIGURE 2. Percentage of eigenvalues of $\tilde{M}_k$ greater than 10^{-6} in absolute value, at each time $t \in [0, 4]$.

Details about the code: Our numerical results were obtained using a code written in the `Python` language. The computation of all derivatives and the minimization routines required for NGS are performed with the `pytorch` library.

4.3. **Quality of reconstruction**

In this section, we study the reconstruction quality of our filtering algorithm in terms of its ability to approximate $\xi^\dagger$, and $u(\xi^\dagger)$. Remember that $\xi^\dagger$ is constant in time ($\xi^\dagger = 6$) but our algorithm computes a time-dependent estimate of it through the (**P**) step (see eq. (21)).

To explore the impact of the amount and location of the observation points in the reconstruction, we start by considering in section 4.3.1 a favorable setting with many observation points ($m = 100$) uniformly spread in $\mathcal{X}$. In section 4.3.2, we consider a more challenging setting with fewer observations ($m = 10$).

4.3.1. *A favorable setting with many, well-located observations*

We first choose for $\{x_i\}_{i=1}^m$ a uniform grid of $m = 100$ of points in $\mathcal{X}$. For our simulations, the quantity $\dot{z}_i$ is estimated using eq. (18) with $\delta t = 10^{-3}$ from the true solution for $\xi^\dagger$. In fig. 3 we represent the estimated state with respect to time and space. We also plot the relative error between the true solution and the approximation at each time step t_k as $\texttt{err}_k := \frac{|\xi^\dagger - \xi_k|}{|\xi^\dagger|}$. The left figure shows that the estimated state is in very good agreement with the true solution displayed in fig. 1. This is confirmed when plotting the reconstructed state for given times $t \in \{0.5, 1, 1.51, 2, 2.5, 3.01\}$ on fig. 4. Similary, the parameter $\xi^\dagger$ is well estimated, up to an error around 13%. These first results demonstrate the ability of our approach to properly estimate both the state u and the parameter $\xi^\dagger$. The rather good reconstruction results are related to the location and the amount of observation points. The next section deals with the situation with very few number of observation points.

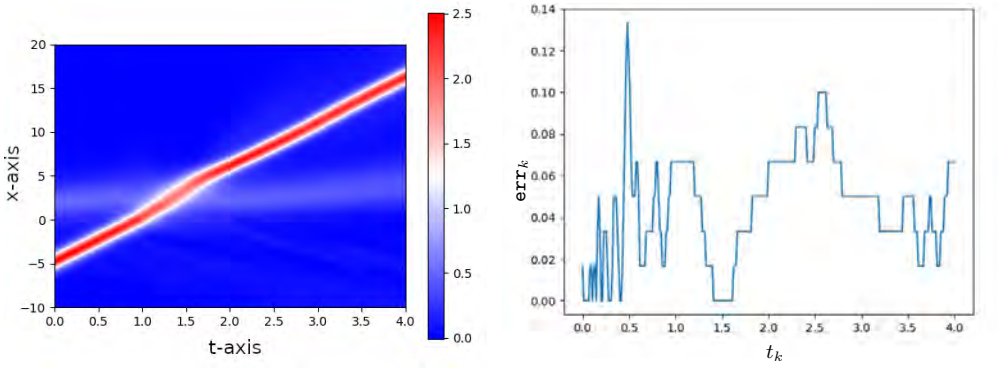

FIGURE 3. **Left:** Reconstructed solution using $m = 100$ sensors uniformly spread over $\mathcal{X}$. **Right:** Relative error $\mathtt{err}_k$ on the approximation of $\xi^\dagger$.

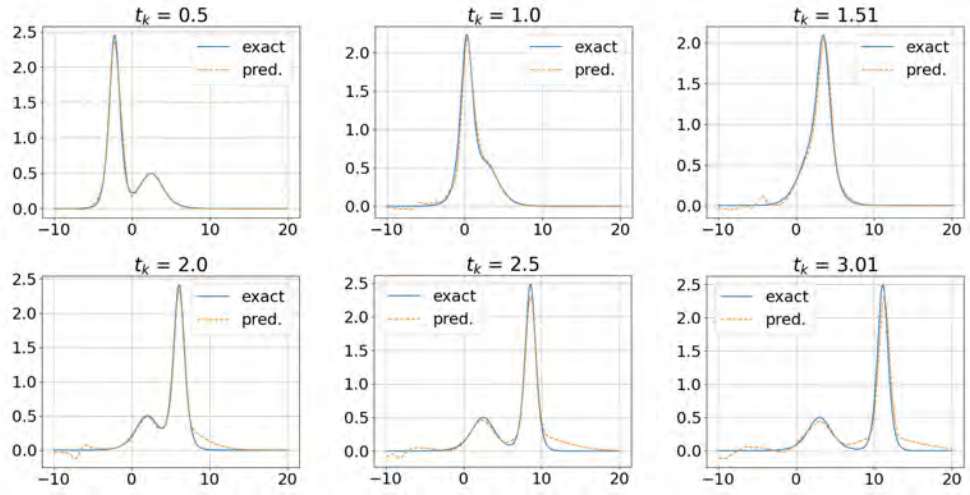

FIGURE 4. Reconstructed solution using $m = 100$ sensors uniformly spread over $\mathcal{X}$ for different times.

4.3.2. *A challenging setting with few observations*

We start by considering the same setting as above but now we only have $m = 10$ observation points uniformly spread over $\mathcal{X}$. The figs. 5 and 6 show the evolution of the reconstructed solution and $\mathtt{err}_k$ of the estimated parameter. In comparison to fig. 1, the dynamics of the solution is clearly not well estimated. This also holds for the parameter estimation for which the relative error can reach 100%. This behavior is not surprising given that the amount of measurements is rather limited, and their locations only allow to sense the solution very partially at every time.

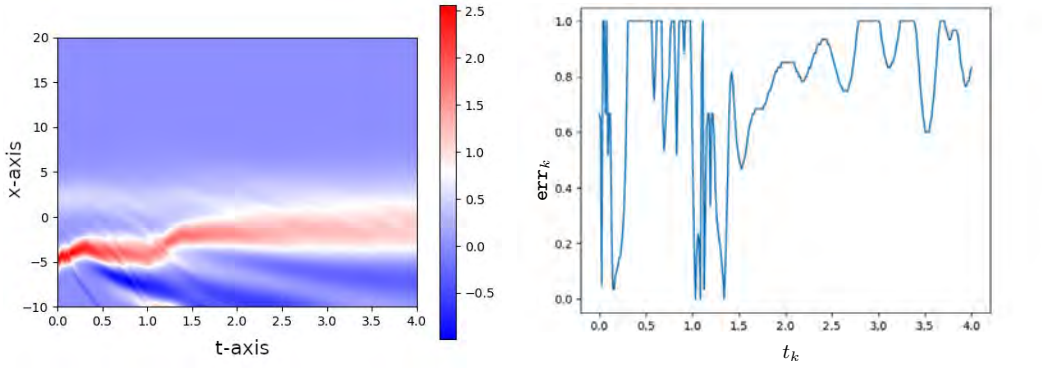

FIGURE 5. **Left:** Reconstructed solution using $m = 10$ sensors uniformly spread over $\mathcal{X}$. **Right:** Relative error $\mathtt{err}_k$ on the approximation of $\xi^\dagger$.

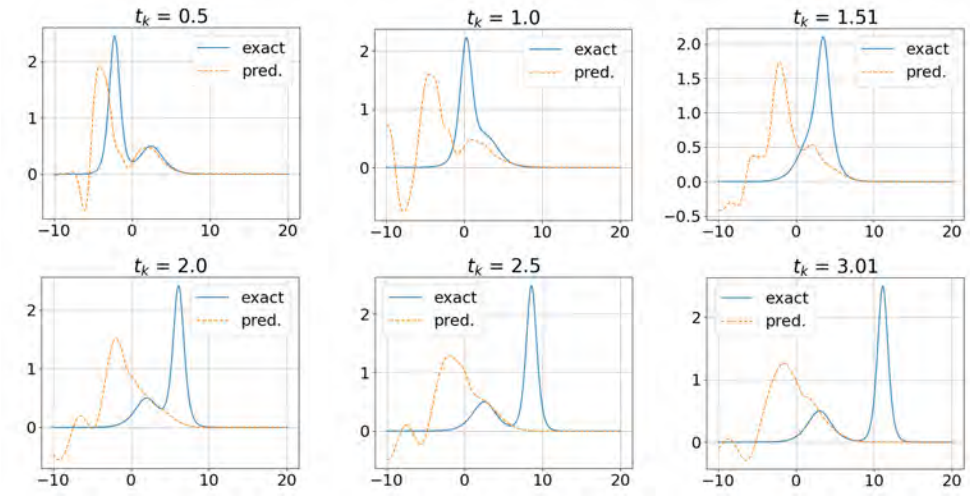

FIGURE 6. Reconstructed solution using $m = 10$ sensors uniformly spread over $\mathcal{X}$ for different times.

We next concentrate the $m = 10$ sensors around the support of the initial condition u_0: we place them uniformly in $[-5, 0] \subset \mathcal{X}$. One may expect that this location allows for a better reconstruction at early times, and degrades as time advances. The obtained behavior is reported on fig. 7, and confirms this intuition: for $t \leq 1$, we obtain fairly decent reconstructions with an error up to 30%. The quality degrades quickly at later times $t > 1$. This is also illustrated on fig. 8 showing that the solution is poorly reconstructed after $t = 1$.

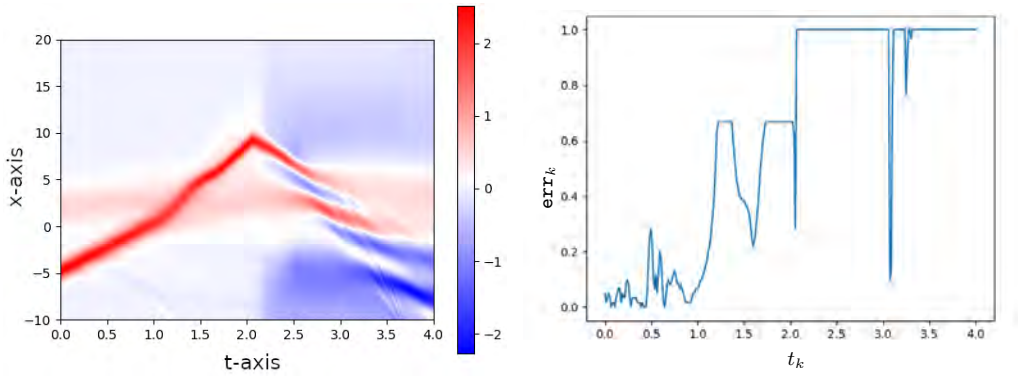

FIGURE 7. **Left:** Reconstructed solution using $m = 10$ sensors concentrated around the support of the initial condition. **Right:** Relative error $\mathtt{err}_k$ on the approximation of $\xi^\dagger$.

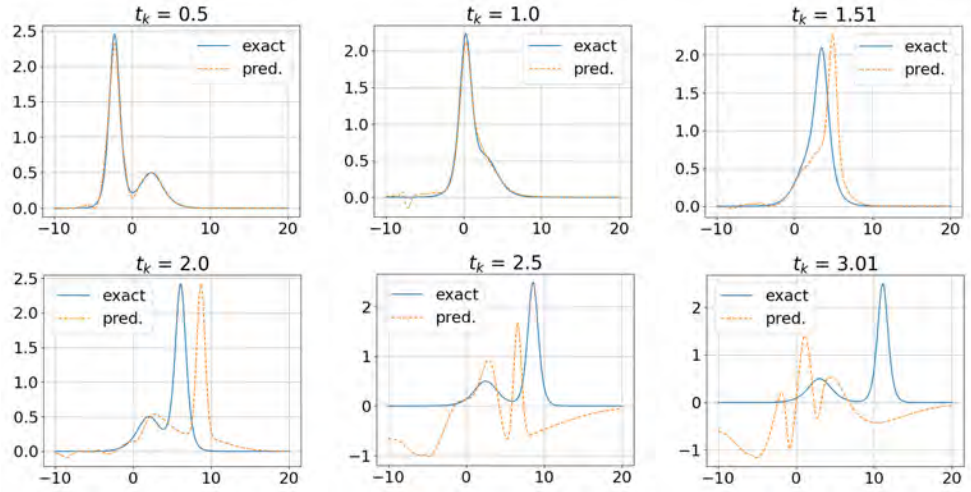

FIGURE 8. Reconstructed solution using $m = 10$ sensors concentrated around the support of the initial condition for different times.

The above experiments illustrate that, when measurements are scarce, the quality of the estimations highly depends on the localization of the observation points with respect to the support of the solution. To improve the quality of the estimates, we now consider $m = 10$ dynamical and equidistant observation points lying in the time dependent interval $[-5 + 4t, 0 + 6t]$. Although this choice is "hand-made", it is motivated by the idea of following the location of the main soliton. In full generality, the location should be given by an algorithm that gives the sensors' movement in the course of the evolution in a similar spirit as was done in [13] for a filtering algorithm with dynamically updated subspaces. The reconstructed solution over space and time is represented on fig. 9. We observe significant improvements of the estimates in a much longer period of time, i.e. over $[0, 3]$ in comparison to the previous case in figs. 5 and 7. In the state estimate, we observe that the second soliton (left bump) is badly estimated as shown on fig. 10, especially after $t > 1.5$. However, in comparison with fig. 1 the main soliton (right bump) is well captured until $t = 3.0$. We can notice a poor parameter estimation after $t > 3.0$, as we can see in the relative error plot in fig. 9. A better choice for the observation points would have been to track simultaneously both solitons according to their respective characteristics.

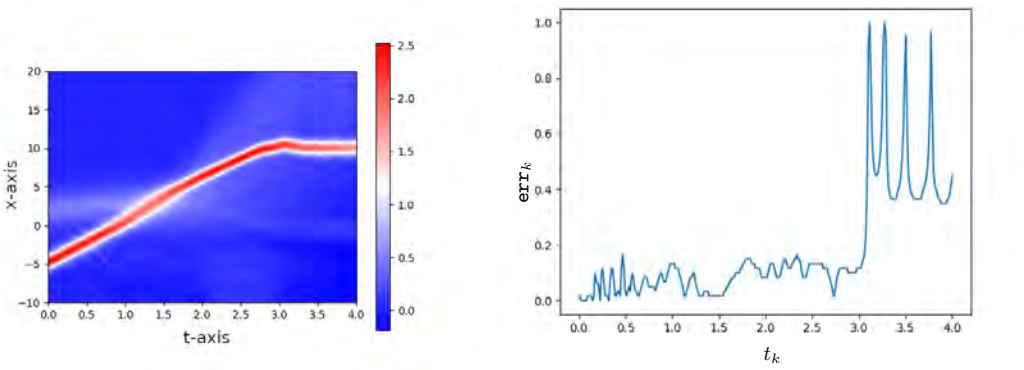

FIGURE 9. **Left:** Reconstructed solution using $m = 10$ sensors that follow the main soliton. **Right:** Relative error $\mathtt{err}_k$ on the approximation of $\xi^\dagger$.

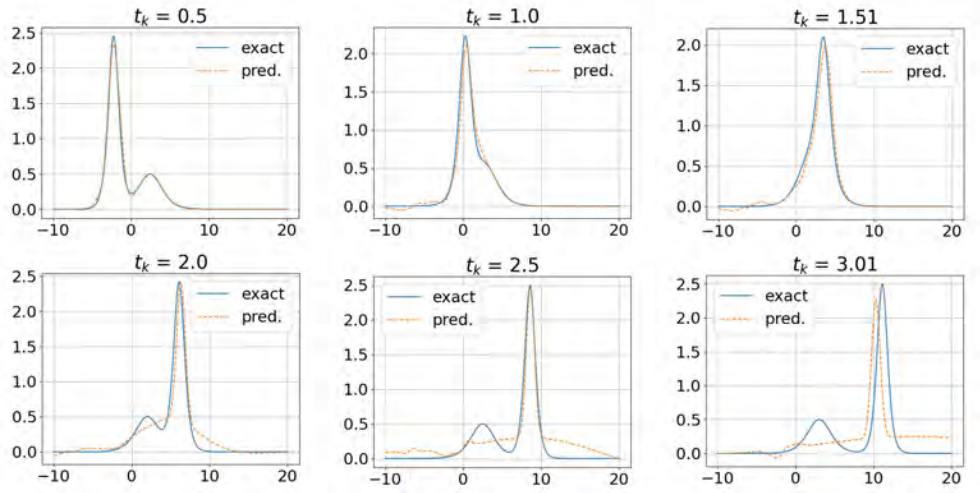

FIGURE 10. Reconstructed solution using $m = 10$ sensors that follow the main soliton for different times.

5. Conclusion

We have introduced an approach for filtering which involves dynamically updated neural networks. The method involves the NGS approach from [4] for forward simulation. We have validated the method for a problem where the underlying dynamics of the function to estimate are governed by a 1D parameter dependent KdV equation. The numerical experiments show that the position of the observation points is of high importance to capture good properties of the solution as well as recovering the latent parameter for the reconstruction with the neural network. When the amount of observations is limited (as is often the case in real applications), our results also indicate that we need to consider methods capable of capturing the dynamics of the function to estimate. There are so far no theoretical foundations for both the forward scheme and the filtering algorithm and this is a crucial point that will be addressed in a forthcoming work.

References

[1] Leah Bar and Nir Sochen. Unsupervised deep learning algorithm for pde-based forward and inverse problems. arXiv preprint arXiv:1904.05417, 2019.

[2] Jules Berman and Benjamin Peherstorfer. Randomized Sparse Neural Galerkin Schemes for Solving Evolution Equations with Deep Networks. arXiv, October 2023.

[3] Marie Billaud-Friess and Anthony Nouy. Dynamical Model Reduction Method for Solving Parameter-Dependent Dynamical Systems. SIAM J. Sci. Comput., August 2017.

[4] Joan Bruna, Benjamin Peherstorfer, and Eric Vanden-Eijnden. Neural Galerkin Scheme with Active Learning for High-Dimensional Evolution Equations. arXiv, March 2022.

[5] G. Donoghue and M. Yano. A multi-fidelity ensemble Kalman filter with hyperreduced reduced-order models. Comput. Methods Appl. Mech. Engrg., 398:Paper No. 115282, 18, 2022.

[6] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. Image processing with neural networks—a review. Pattern recognition, 35(10):2279–2301, 2002.

[7] Othmar Koch and Christian Lubich. Dynamical Low-Rank Approximation. SIAM J. Matrix Anal. Appl., April 2007.

[8] D. Lombardi. State estimation in nonlinear parametric time dependent systems using tensor train. Internat. J. Numer. Methods Engrg., 123(20):4935–4956, 2022.

[9] Christian Lubich. From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis, September 2008. [Online; accessed 29. Jan. 2024].

[10] Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken. Dynamical Approximation by Hierarchical Tucker and Tensor-Train Tensors. SIAM J. Matrix Anal. Appl., May 2013.

[11] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes. IMA Journal of Numerical Analysis, 42(2):981–1022, 2022.

[12] Roberto Molinaro, Yunan Yang, Björn Engquist, and Siddhartha Mishra. Neural inverse operators for solving pde inverse problems. arXiv preprint arXiv:2301.11167, 2023.

[13] Olga Mula, Cecilia Pagliantini, and Federico Vismara. Dynamical approximation and sensor placement for filtering problems. arXiv preprint arXiv:2312.12353, 2023.

[14] Achim Nonnenmacher and Christian Lubich. Dynamical low-rank approximation: applications and numerical experiments. Math. Comput. Simul., 79(4):1346–1357, December 2008.

[15] 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.

[16] J. Schmidt, P. Hennig, J. Nick, and F. Tronarp. The rank-reduced Kalman filter: Approximate dynamical-low-rank filtering in high dimensions. arXiv preprint arXiv:2306.07774, 2023.

[17] Thiab R Taha and Mark I Ablowitz. Analytical and numerical aspects of certain nonlinear evolution equations. III. Numerical, Korteweg de Vries equation. Journal of Computational Physics, 55(2):231–253, 1984.

[18] André Uschmajew and Bart Vandereycken. The geometry of algorithms using hierarchical tensors. Linear Algebra Appl., 439(1):133–166, July 2013.

[19] E. Vidličková. Dynamical low rank approximation for uncertainty quantification of time-dependent problems. PhD thesis, EPFL, 2022.