A Trainable Approach to Zero-Delay Smoothing Spline Interpolation

Emilio Ruiz-Moreno ⓘ, Luis Miguel López-Ramos ⓘ, *Member, IEEE*, and
Baltasar Beferull-Lozano ⓘ, *Senior Member, IEEE*

Abstract—The task of reconstructing smooth signals from streamed data in the form of signal samples arises in various applications. This work addresses such a task subject to a zero-delay response; that is, the smooth signal must be reconstructed sequentially as soon as a data sample is available and without having access to subsequent data. State-of-the-art approaches solve this problem by interpolating consecutive data samples using splines. Here, each interpolation step yields a piece that ensures a smooth signal reconstruction while minimizing a cost metric, typically a weighted sum between the squared residual and a derivative-based measure of smoothness. As a result, a zero-delay interpolation is achieved in exchange for an almost certainly higher cumulative cost as compared to interpolating all data samples together. This paper presents a novel approach to further reduce this cumulative cost on average. First, we formulate a zero-delay smoothing spline interpolation problem from a sequential decision-making perspective, allowing us to model the future impact of each interpolated piece on the average cumulative cost. Then, an interpolation method is proposed to exploit the temporal dependencies between the streamed data samples. Our method is assisted by a recurrent neural network and accordingly trained to reduce the accumulated cost on average over a set of example data samples collected from the same signal source generating the signal to be reconstructed. Finally, we present extensive experimental results for synthetic and real data showing how our approach outperforms the abovementioned state-of-the-art.

Index Terms—Smoothing spline interpolation, stream learning, sequential decision making, recurrent neural network.

I. INTRODUCTION

ONLINE learning has been studied and applied in a broad range of research fields, including optimization theory [5], [6], [7], signal processing [8], and machine learning

Fig. 1. A signal reconstructed from stream data points by different methods and models. The symbols x_t and f_t represent the current time and signal estimate, respectively. (a) Recursive least squares [1], [2] with a linear model. (b) NORMA [3] with Gaussian kernels. (c) Smoothing myopic interpolation with cubic Hermite splines [4].

[9], [10], [11]. Within these fields, online methods generating a series of estimates from sequentially streamed data are especially useful to reduce complexity in large-scale problems [12], to dynamically adapt to new patterns in the data [13], and to enable acting under real-time requirements [14].

This work addresses the last one of the previous use cases in the context of signal reconstruction. Specifically, it investigates the use of online methods with *zero-delay* response for *smooth* signal reconstruction. First, most physical signals are bounded and smooth due to energy conservation [15]; hence it is beneficial to maintain smoothness as a property during signal reconstruction. Second, the zero-delay requirement demands new portions of the smooth signal to be reconstructed as soon as a new data sample is available. Consequently, a reduced constant complexity per iteration [16] is required so that the online method is executed at a higher speed than the transmission rate at which the data samples are received. These requisites are well-motivated since they appear in many practical problems, such as online trajectory planning [17], [18], real-time control systems [19], [20], and high-speed digital to analog conversion [21], among others. Although the tasks of estimating smooth signals or delivering a zero-delay response are separately managed by most online methods, handling them together becomes challenging, as we expound next.

Some popular online methods that can be used for smooth signal estimation are online kernel methods [22], [23] and online Gaussian processes [24], [25]. They aim at yielding a sequence of signal estimates with convergence guarantees or sublinear regret [26]. To this end, they initially propose a signal estimate which is updated (or modified) possibly globally as new data samples arrive. Their goal is to refine the signal estimate rather than reconstruct new portions of the smooth signal. Therefore, neither smoothness nor even continuity of the sequentially reconstructed signal is guaranteed. In fact, any online method not ensuring the smoothness of the reconstruction during the signal estimate update (even when the signal estimate is modeled by smooth functions) suffers from this issue, as illustrated in Fig. 1(a) and 1(b). On the other hand, online interpolation methods can be suitable candidates for the task of smooth signal reconstruction with a zero-delay response. These methods use piecewise-defined functions to model a sequence of local signal estimates. Some of these functions allow shaping piecewise-modeled signal estimates that can be updated by assembling a new section (or piece) while guaranteeing the smoothness of the overall sequentially reconstructed signal, as shown in Fig. 1(c). Among them, piecewise polynomial functions, also known as *splines*, are arguably the most representative ones [27], [28]. Actually, splines have been used since ancient times [29], long before their mathematical foundations were even established [30], presumably because of their approximation capabilities over functions of arbitrary complexity and ease of use.

It should be noted that most recursive signal estimation methods modeling function estimates by splines as a basis expansion [31], [32], [33], suffer from the same issues exposed before. This is mainly because the smoothness of their signal estimate is directly incorporated into the basis representation and not treated as a set of continuity constraints. On the contrary, some works [34], [35] have explored the task of interpolating sequentially streamed data under real-time requirements by means of splines subject to continuity constraints. However, the online methods they use involve a multi-step lookahead or shifting window mechanism, which introduces a delay. Indeed, most online methods for spline interpolation work with local information, for instance, a subset constituted by the last sequentially received data samples. In this case, a delayed response allows them to use a larger subset of sequentially received data samples and correct the signal estimates as long as they are updated within the delay limits. In brief, they can expand the extent of available information at the expense of some delay. On the other hand, and to the best of our knowledge, the only zero-delay spline interpolation method in the literature is the *myopic* approach, referred to as the "classical greedy approach" in [35], which reduces the delay response to zero by totally ignoring any source of forthcoming information, i.e., a purely local method. Clearly, there is a research gap on zero-delay spline interpolation methods exploiting additional nonlocal information to achieve a better reconstruction. This motivates us to work on the research question of whether it is possible to maintain the zero-delay requirement while efficiently using more information than the myopic approach.

In this paper, we answer affirmatively to the above research question by introducing a novel method for zero-delay smoothing[1] spline interpolation that incorporates a priori information about the dynamics of the signal being reconstructed. To this end, we identify the elements of a state space-based sequential decision-making process [36] in the context of zero-delay smoothing spline interpolation. The proposed method relies on a *policy*, i.e., a strategy, that yields a section of the spline (*action*) as a function of the current condition of the so far reconstructed signal and the last received data sample (*state*). Such a policy consists of a differentiable convex optimization layer (DCOL) [37] on top of a recurrent neural network (RNN) [38], [39]. The DCOL allows managing continuity constraints (for any differentiability class) at each interpolation step, thus guaranteeing the smoothness of the signal reconstruction. The RNN assists the signal estimate update mechanism when appending a new spline section by taking into account the effect of each interpolated section on future interpolation steps. This aid comes in the form of global data-driven knowledge, and it is tailored to minimize the global cost of the smoothing interpolation problem, on average. The cost is, in this case, the residual sum of squares plus a weighted derivative-based measure of smoothness. Lastly, our method is *trainable* in the sense that it uses example time series, i.e., time series sampled from the same signal source generating the signal to be reconstructed, to customize the policy to the temporal dependencies (dynamics) of the signal at hand.

The main contributions of this paper can be summarized as follows:

- We rigorously formulate the problem of smoothing spline interpolation from sequentially streamed data, where each spline section has to be determined as soon as a data sample is available and without having access to subsequent data (zero-delay requirement). Due to its nature, it is formulated as a sequential decision-making problem.
- As opposed to previously proposed (myopic and not trainable) zero-delay methods, our method trains a policy that aims at minimizing the smoothing interpolation cost metric on average. In order to capture the temporal, possibly long-term, dependencies between the streamed data samples and exploit them to reduce further the average cost metric, an RNN able to capture the signal dynamics is incorporated.
- The proposed policy guarantees that the reconstructed signal is smooth (a certain number of derivatives are continuous over the interior of the signal domain). This is achieved by adding a DCOL at the output of the RNN and imposing a set of continuity constraints at each interpolation step. In addition, such a layer admits a closed-form evaluation, resulting in improved computational efficiency with respect to off-the-shelf DCOL libraries.
- We present extensive experimental results that validate our approach over synthetic and real data. Additionally, we show how our approach outperforms the state-of-the-art (namely, myopic) zero-delay methods in terms of the smoothing interpolation average cost metric.

---

[1]Here, the term smoothing refers to a controlled trade-off between fitting the data samples and proposing a smooth signal estimate.

The rest of the paper is structured as follows: Sec. II introduces the notation and presents some basic concepts and definitions. Then, in Sec. III, we provide our problem formulation. Next, in Sec. IV and Sec. V, we respectively provide a solution, a benchmark, and a baseline. Thereafter we experimentally validate our solution in Sec. VI. Finally, Sec. VII concludes the paper.

II. PRELIMINARIES

In this section, we present the notation and introduce the type of data used in the paper. Afterward, we address the description of spline-based signal estimates as well as related concepts recurrently appearing in this work. Finally, we formally describe the smoothing spline interpolation problem, which will be used as a starting point for the formulation of our problem.

A. Notation

Vectors and matrices are denoted by bold lowercase and capital letters, respectively. Given a vector $\boldsymbol{v} = [v_1, \ldots, v_C]^\top$, its cth component is indicated as $[\boldsymbol{v}]_c \triangleq v_c$. Similarly, given a matrix $\boldsymbol{M} \in \mathbb{R}^{R \times C}$, the element in the rth row and cth column is indicated as $[\boldsymbol{M}]_{r,c}$. The notation $[\boldsymbol{v}]_{i:j}$ refers to the sliced vector $[v_i, \ldots, v_j]^\top \in \mathbb{R}^{j-i+1}$. We use Euler's notation for the derivative operator; thus, D_x^k denotes the kth derivative over the variable x.

B. Problem Data

The data considered in this paper consist of discrete time series, or *series* for short, of T terms each. We interchangeably refer to the terms of the series as *observations*. Each tth observation $\boldsymbol{o}_t \in \mathbb{R}^2$ is described by its time stamp $x_t \in \mathbb{R}$ and its value $y_t \in \mathbb{R}$, i.e., $\boldsymbol{o}_t = [x_t, y_t]^\top$. The observation-associated time stamps are set in strictly monotonically increasing order, i.e., $x_{t-1} < x_t$ for all terms in the series. Any two consecutive time stamps define a time section $\mathcal{T}_t = (x_{t-1}, x_t]$. Finally, the initial time stamp x_0 is set by the user.

C. Spline-Based Signal Estimates

A spline is defined as a piecewise polynomial function. We denote any spline composed of T piecewise-defined functions, or *function sections*, as

$$f_T(x) = \begin{cases} g_1(x), & \text{if } x_0 < x \leq x_1 \\ g_2(x), & \text{if } x_1 < x \leq x_2 \\ \vdots \\ g_T(x), & \text{if } x_{T-1} < x \leq x_T \end{cases} \quad (1)$$

where every tth function section $g_t : \mathcal{T}_t \to \mathbb{R}$ is a linear combination of polynomials of the form

$$g_t(x) = \boldsymbol{a}_t^\top \boldsymbol{p}_t(x), \quad (2)$$

with combination coefficients $\boldsymbol{a}_t \in \mathbb{R}^{d+1}$ and basis vector function $\boldsymbol{p}_t : \mathcal{T}_t \to \mathbb{R}^{d+1}$ defined as

$$\boldsymbol{p}_t(x) = \left[1, (x - x_{t-1}), \ldots, (x - x_{t-1})^d\right]^\top. \quad (3)$$

The integer d denotes the *order* of the spline. A spline f_T is said to have a *degree of smoothness* φ if it has φ continuous derivatives over the interior of its domain $\text{dom}(f_T) = \bigcup_{t=1}^T \mathcal{T}_t$. Next, **Proposition 1** shows how to enforce continuity up to degree $\varphi \leq d$ in a spline-based signal estimate of order d.

Proposition 1: Given a spline expressed as in (1), we can enforce its degree of smoothness to be $\varphi \leq d$ by imposing the following equality constraint

$$[\boldsymbol{a}_t]_{1:\varphi+1} = \boldsymbol{e}_{t-1}, \quad (4)$$

for every $t \in \mathbb{N}^{[1,T]}$, where $\boldsymbol{e}_t \in \mathbb{R}^{\varphi+1}$ is a vector such that each of its elements is computed as

$$[\boldsymbol{e}_t]_i = \frac{1}{(i-1)!} \sum_{j=1}^{d+1} [\boldsymbol{a}_t]_j \, u_t^{j-i} \prod_{k=1}^{i-1} (j-k), \quad (5)$$

with $u_t \triangleq x_t - x_{t-1}$, and with the exception of $\boldsymbol{e}_0$, which determines the initial conditions of the reconstruction and can be either calculated or set by the user.

Proof: see Section A in the Appendix.

D. Smoothing Spline Interpolation

Consider the space $\mathcal{W}_\rho$ of functions defined over the domain $(x_0, x_T) \subseteq \mathbb{R}$ with $\rho - 1$ absolutely continuous derivatives and with the ρth derivative square integrable. Then, given a whole series of observations $\{\boldsymbol{o}_t\}_{t=1}^T$ with $T \geq \rho$ and a positive hyperparameter η, we can formulate the following batch optimization problem

$$\min_{f \in \mathcal{W}_\rho} \sum_{t=1}^T (f(x_t) - y_t)^2 + \eta \int_{x_0}^{x_T} (D_x^\rho f(x))^2 \, dx \quad (6)$$

known as smoothing spline interpolation [40], [41]. The name is due to the unique solution to the optimization problem (6) being a spline conformed of T function sections, as in (1). More specifically, the solution of (6) is a spline of order $2\rho - 1$ with $2\rho - 2$ continuous derivatives and natural boundary conditions [31]. The hyperparameters η and ρ control the smoothness of such a solution. Particularly, the integer ρ dictates the minimum required degree of smoothness of the search function space $\mathcal{W}_\rho$ and the type of regularization[2] (second term in (6)). Regarding η, it controls the trade-off between the squared sum of vertical deviations of the signal estimate from the data and the regularization term. Notice that as $\eta \to 0$, the solution of (6) approaches the interpolation spline while as $\eta \to \infty$, it tends to the polynomial of order $\rho - 1$ that best fits the observations in the least-squares sense.

On the other hand, note that the structure of the solution of the problem (6), being a natural spline, arises organically rather than being imposed in advance. This is a direct consequence of its batch formulation allowing us to delimit the search space $\mathcal{W}_\rho$ to splines of order d and degree of smoothness φ satisfying $2\rho - 1 \leq d$ and $\rho - 1 \leq \varphi \leq 2\rho - 2$ without loss of optimality.

---

[2]Our experimental setup focuses on $\rho = 2$, a common choice in practice, which penalizes excessive curvature in the spline. Applications with $\rho > 2$ can also be found, e.g., trajectory planning tasks [42]. However, they are out of the scope of this paper, as we justify in the ensuing Sec. VI-B.

From a practical perspective, it is sufficient to choose the minimum required order and degree of smoothness, thus reducing the model's complexity. However, this trait is not necessarily present in online settings. That is, the smoothness of the solution does not arise naturally using online methods, and it has to be enforced. So here, the choice of the spline order and degree of smoothness is rather user-defined or task-oriented.

III. PROBLEM FORMULATION

Once the problem data, the description of spline-based signal estimates, and the smoothing spline interpolation problem have been introduced in Sec. II, we are ready to formalize the main task of this paper, namely the trainable zero-delay smoothing spline interpolation problem. This section fully describes the aforementioned task from a data-driven sequential decision-making perspective by introducing a suitable dynamic programming (DP) [43] framework. To this end, we first model the *environment*, define the state space and action space, and delimit a suitable family of candidate policies. Then, we introduce the *total cost* and formulate the above task as the problem of finding the policy incurring the lowest total cost on average.

A. Characterization of the Problem Data

In our problem, the data described in Sec. II-B are observed sequentially. Before every tth time step, the observation about to be received o_t remains undetermined but still governed by the dynamics of the environment. In this work, we model the dynamics of the environment as a random process $Y(\omega, x)$, where ω is a sample point from a sample space Ω, and x is a value within an index set $\mathcal{X} \subseteq \mathbb{R}$, in this case, time. At time x_t, all possible outcomes form a random variable $Y(\omega, x_t)$ or y_t for short. If the mth sample is considered at time x_t, the outcome has a value denoted by $Y(\omega_m, x_t)$ or simply $y_{m,t}$. Consequently, if a discrete set of time stamps is chosen, i.e., $\mathcal{X} = \{x_1, \ldots, x_T\}$, T random variables can be formed, and all the information about the discrete random process $Y_{\mathcal{X}}$ is contained in the joint probability density function $P_{Y_{\mathcal{X}}}$.

B. State Space

At every time step t, we encode a snapshot of the observable environment and the condition of the so-far reconstructed signal in a vector-valued variable called state. With $\mathcal{S}$ denoting the state space, each tth state $s_t \in \mathcal{S}$ is constituted by the corresponding observation o_t, and the condition at which the reconstruction was left, which is specified by the vector e_{t-1} whose components are given as in (5), and the time instant x_{t-1}. Formally, every tth state s_t is expressed as $s_t = [o_t^\top, e_{t-1}^\top, x_{t-1}]^\top$. Since every state s_t is uniquely determined once the spline coefficients a_{t-1} are fixed, we can explicitly describe the state update mechanism, by means of a deterministic mapping, as

$$s_{t+1} = F(s_t, a_t, o_{t+1}). \qquad (7)$$

Formalizing the state update mechanism in (7) allows us to identify all visitable states seamlessly.

C. Action Space

Immediately after receiving the tth observation, we propose a function section as in (2), and we implicitly select the spline coefficients a_t. This is because the function section is determined as soon as a_t is chosen (the basis vector defined in (3) is given). From this point of view, selecting the spline coefficients of a function section can be understood as an action. Any valid action generates a function section of the same order d as the spline reconstruction. Formally, $a_t \in \mathcal{A} \subseteq \mathbb{R}^{d+1}$ for all the tth terms, where $\mathcal{A}$ denotes the action space. However, if we want a reconstructed spline that is continuous up to the φth derivative, not all valid actions are appropriate. In our context, for any tth action to be deemed *admissible* (or feasible), it must satisfy the constraint in (4). Notice that the set of admissible actions depends on the current state. Therefore, we accordingly denote the admissible action space as $\mathcal{A}(s_t)$.

D. Policy Space

A policy $\pi = \{\mu_t : \mathcal{S} \to \mathcal{A}\}_{t \in \{1, 2, \ldots\}}$ consists of a sequence of functions that map states into actions. Policies are more general than actions because they incorporate the knowledge of the state. However, notice that not all policies return admissible actions. Only the policies that satisfy $\pi(s_t) \in \mathcal{A}(s_t)$ for all time steps are termed *admissible policies*. Separately, *stationary policies* are policies that do not change over time, i.e., $\mu \equiv \mu_t = \mu_{t+1}$ for all time steps. Hence, a stationary policy is unequivocally defined by the mapping μ. Stationary policies are suitable for making decisions in problems with a varying horizon (varying number of time steps), assuming usually stationary environments.

These arguments motivate the use of admissible stationary policies. However, the space of admissible stationary policies is huge, and therefore, the problem of finding the most adequate policy within it can be overwhelmingly complex. Policy approximation techniques help reduce the pool of candidate policies by restricting them to a certain family of policies. These techniques tend to work best (in the sense of providing an adequate policy) when the problem has a clear structure that can be accommodated into the policy. In our case, we aim to incorporate the temporal dependencies across the observations into the policy, as well as the notion of smoothness discussed in Sec. II-D. To this end, we resort to parametric policy approximation [10] denoting any approximated stationary policy as μ_θ, where the vector $\theta \in \mathbb{R}^P$ contains the P parameters constituting the aforenamed policy. The set Π of parametric stationary policies that return admissible actions is, therefore, the space of policies of interest to this work.

E. Total Expected Cost

The following **Proposition 2** shows that the smoothing spline interpolation objective introduced in Sec. II-D, equation (6), can be expressed as a summation where each term depends on a single action, resembling the sequence of instantaneous costs in a typical DP formulation.

Proposition 2: The objective of the optimization problem (6) can be equivalently computed additively as

$$\sum_{t=1}^{T} \left(\boldsymbol{a}_t^\top \boldsymbol{p}_t(x_t) - y_t \right)^2 + \eta \, \boldsymbol{a}_t^\top \boldsymbol{M}_t \boldsymbol{a}_t, \qquad (8)$$

where $\boldsymbol{M}_t \in \mathbf{S}_+^{d+1}$ with elements given by

$$[\boldsymbol{M}_t]_{i,j} = \begin{cases} 0 & \text{if } i \leq \rho \text{ or } j \leq \rho \\ \frac{u_t^{i+j-2\rho-1}}{i+j-2\rho-1} \prod_{k=1}^{\rho} (i-k)(j-k) & \text{otherwise,} \end{cases} \qquad (9)$$

being $u_t \triangleq x_t - x_{t-1}$.

Proof: See Section B in the Appendix.

Based on **Proposition 2**, we can express the objective of the smoothing spline interpolation problem (6), as the total cost $\sum_{t=1}^{T} \kappa(\boldsymbol{s}_t, \boldsymbol{a}_t)$, with $cost \kappa : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ given by

$$\kappa(\boldsymbol{s}_t, \boldsymbol{a}_t) = \left(\boldsymbol{a}_t^\top \boldsymbol{p}_t(x_t) - y_t \right)^2 + \eta \, \boldsymbol{a}_t^\top \boldsymbol{M}_t \boldsymbol{a}_t, \qquad (10)$$

where $\boldsymbol{M}_t$ is constructed as in (9). This is because each tth state-action pair contains all necessary information. From here and under a given policy of interest $\boldsymbol{\mu_\theta}$, as described in Sec. III-D, the metric

$$\mathbb{E}_{y_t \sim P_{Y_{\mathcal{X}}}} \left[\sum_{t=1}^{T} \kappa \left(\boldsymbol{s}_t, \boldsymbol{\mu_\theta}(\boldsymbol{s}_t) \right) \right], \qquad (11)$$

denotes the *total expected cost* incurred by following such a policy from a given initial state $\boldsymbol{s}_0$, and traveling all the remaining states $\boldsymbol{s}_t \in \mathcal{S}_t$ via (7). The expectation in (11) is performed over the random process modeling the dynamics of the environment through the observations within the states.

F. Policy Search by Cost Optimization

Computing the expectation in (11) is computationally expensive or even intractable when the underlying random process generating the series of observations is unknown. Instead, we can rely on sample average approximation of example series collected from past realizations of the process. The sample average approaches the expectation as the number of examples grows. In this way, we can determine a data-driven policy by solving the following optimization problem

$$\arg \min_{\boldsymbol{\mu_\theta} \in \Pi} \sum_{m=1}^{M} \sum_{t=1}^{T} \kappa \left(\boldsymbol{s}_{m,t}, \boldsymbol{\mu_\theta}(\boldsymbol{s}_{m,t}) \right) \qquad (12a)$$

$$\text{s. to: } \boldsymbol{s}_{m,t} = F \left(\boldsymbol{s}_{m,t-1}, \boldsymbol{\mu_\theta}(\boldsymbol{s}_{m,t-1}), \boldsymbol{o}_{m,t} \right), \forall m, t, \qquad (12b)$$

$$\boldsymbol{\mu_\theta}(\boldsymbol{s}_{m,t}) \in \mathcal{A}(\boldsymbol{s}_{m,t}), \forall m, t, \qquad (12c)$$

where the integer M denotes the number of example series, indexed by m, and where all the initial states $\boldsymbol{s}_{m,0}$ as well as all observations $\boldsymbol{o}_{m,t}$ are given.

IV. PROPOSED SOLUTION

The previous Sec. III has provided the necessary definitions and considerations to arrive at a rigorous problem formulation. An exact solution to the problem (12) is probably impossible to obtain in practice, mainly due to the complexity of the search space Π. There are multiple possibilities regarding the policy approximation and optimization techniques that can be taken towards obtaining a near-optimal solution to (12). This section presents a specific set of design choices based on the current state-of-the-art. In particular, we rely on a policy parametrization through cost parametrization technique, borrowed from the DP literature, in synergy with an RNN architecture. Then we make use of backpropagation through time (BPTT) [44], a gradient computation technique borrowed from the deep learning literature [45]. Our proposed solution can effectively solve the problem formulated in Sec. III-F for $\rho \leq 2$. The remaining configurations manifest instability issues, and even though they may be solvable, they lie outside of the scope of the current paper as further discussed in the following Sec. VI-B.

Future developments in the DP or deep learning areas, such as new policy approximation approaches, neural architectures, or optimizers, can possibly render the techniques proposed in this section obsolete but will not affect the validity of the problem formulated in Sec. III-F.

A. Policy Form

Parametric policy approximation via parametric cost function approximation (CFA) [10] is a method that seeks through the policy space, in our case Π, among those policies defined as an optimization problem with parametrized objectives. In this work, we are interested in CFA-based policies of the form

$$\boldsymbol{\mu_\theta}(\boldsymbol{s}_{m,t}) = \arg \min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s}_{m,t})} \left\{ \kappa(\boldsymbol{s}_{m,t}, \boldsymbol{a}) + J_{\boldsymbol{\theta}}(\boldsymbol{s}_{m,t}, \boldsymbol{a}; \boldsymbol{h}_{m,t}) \right\}, \qquad (13)$$

where the map κ denotes the cost described in (10), and the mapping $J_{\boldsymbol{\theta}} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^H \to \mathbb{R}$ is a parametric *cost-to-go* approximation involving P parameters contained in the vector $\boldsymbol{\theta}$. Regarding the vector $\boldsymbol{h}_{m,t} \in \mathbb{R}^H$, it represents a *latent state* value at the tth time step of an mth example series. The latent state may encode relevant information from past observations and can be viewed as a policy memory [46], [47], [48].

We aim for a cost-to-go approximation $J_{\boldsymbol{\theta}}$, which penalizes those actions that are distant from the output of a certain RNN. The main reason behind this approach is that an RNN that successfully captures the temporal dynamics of the environment has the potential to pull towards actions that yield a low expected total cost. So, it is constructed as follows

$$J_{\boldsymbol{\theta}}(\boldsymbol{s}_{m,t}, \boldsymbol{a}; \boldsymbol{h}_{m,t}) = \lambda \left\| \boldsymbol{a} - \begin{bmatrix} \boldsymbol{0}_{\varphi+1} \\ \boldsymbol{r}_{m,t} \end{bmatrix} \right\|_2^2, \qquad (14)$$

where $\lambda \in \mathbb{R}_+$. The vectors $\boldsymbol{r}_{m,t} \in \mathbb{R}^{d-\varphi}$, and $\boldsymbol{h}_{m,t} \in \mathbb{R}^H$ represent the outputs and latent state of an RNN, $R_{\boldsymbol{\theta}'} : \mathcal{S}_t \times \mathbb{R}^H \to \mathbb{R}^{d-\varphi} \times \mathbb{R}^H$, respectively. They are obtained from the following relation

$$R_{\boldsymbol{\theta}'}(\boldsymbol{s}_{m,t}; \boldsymbol{h}_{m,t}) = \begin{bmatrix} \boldsymbol{r}_{m,t} \\ \boldsymbol{h}_{m,t+1} \end{bmatrix}, \qquad (15)$$

with $\boldsymbol{\theta} = [\lambda, \boldsymbol{\theta}'^\top]^\top$ and $\boldsymbol{\theta}' \in \mathbb{R}^{P-1}$, exemplified in Fig. 2. From now on, we refer to the policy in (13) with cost-to-go as in (14) as the RNN-based policy. Finally, notice that besides being
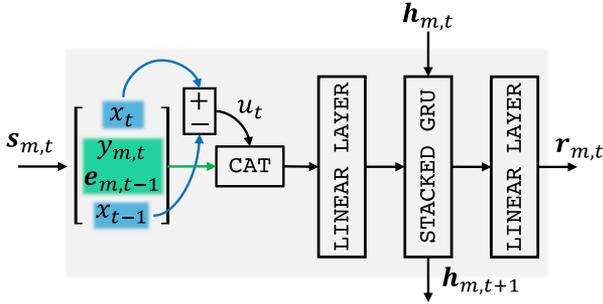
Fig. 2.   Visual representation of our RNN architecture satisfying the relation in (15). The elements onto the gray shaded area constitute the mapping $R_{\boldsymbol{\theta}'}$. The CAT cell stands for a concatenation operation.

parametric, the RNN-based policy is admissible and stationary by design.

B. Policy Evaluation

Evaluating the proposed policy involves solving the optimization problem stated in (13). Notice that both the cost κ built as in (10), and the cost-to-go approximation $J_{\boldsymbol{\theta}}$ described in (14), are convex with respect to the actions and hence, the objective in (13) is convex too. Moreover, the admissible action set, described in Sec. III-C, is convex. Therefore, the optimization problem in (13) is convex thus, any locally optimal action is globally optimal [49].

Additionally, the optimization problem in (13) has been designed to admit a closed-form solution. Closed-form evaluations can usually be computed faster and more precisely than solutions obtained from numerical methods, and thus, they are more suitable under zero-delay requirements. See Section C in the Appendix for the derivation of the closed-form evaluation.

C. Policy Training

As explained in Sec. IV-A, we have reduced the search space of problem (12) by restricting the policy space to a family of policies of the form given in (13). Specifically, from searching a function $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ in the function space Π, we have narrowed the problem down to that of finding a vector $\boldsymbol{\theta}$ in the vector space $\mathbb{R}^P$. In fact, tuning the proposed policy parameters by solving the optimization problem (12) is commonly referred to as policy training. Unfortunately, the objective (12a) is non-convex with respect to the parameters in $\boldsymbol{\theta}$. As a reasonable solution, we rely on a gradient-based optimizer aiming to converge to a high-performance local minimum.

From a deep learning perspective, the policy evaluation presented in Sec. IV-B can be understood as a forward pass of a DCOL on top of an RNN, and hence, it is trained using BPTT via automatic differentiation [50]. This point of view is schematized in Fig. 3, where traveling the given mth series, by following a policy $\boldsymbol{\mu}_{\boldsymbol{\theta}}$, allows to construct the cumulative objective in (12a) used for training. Additionally, and thanks to the closed-form policy evaluation discussed in Sec. IV-B, computing and propagating the gradient of the tth action $\boldsymbol{a}_t$ with respect to the parameters contained in $\boldsymbol{\theta}$ is done avoiding the need of unrolling numerical optimizers [51] or using specific numerical tools for DCOLs such as CVXPY Layers [37].
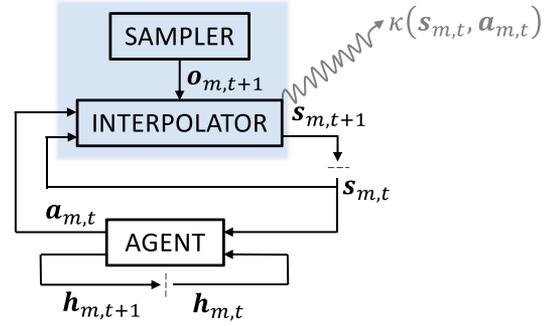
Fig. 3.   Rolled representation of the environment-agent system. The sampler cell samples the random process modeling the dynamics of the environment. The interpolator cell performs the reconstruction, evaluates the cost, and updates the states. The agent cell contains the policy and the RNN. The blue-shaded area encompasses the environment. The cost (in gray) is used during the training phase but not for evaluation.

V. BENCHMARK AND BASELINE METHODS

Recall from Sec. II-D that the batch formulation provides the optimal reconstruction with hindsight. The batch solution can be found by solving the optimization problem (6), but only once all time-series data are available. Thus, it cannot be used for zero-delay interpolation. Conceptually, online methods achieve a zero-delay response at the expense of incurring higher or equal loss than the batch solution. For this reason, the batch solution is used here as a baseline.

On the other hand, as stated in the Introduction and to the best of our knowledge, there is no related work to our trainable zero-delay smoothing interpolation approach in the literature. One could consider that the closest approach is the interpolation method known as myopic. This is a local method in the sense that it only focuses on the last received data sample while completely ignoring the distribution of future arriving data. For this reason, the myopic method is used here as a benchmark. In this sense, our proposed method must outperform the myopic method to be deemed acceptable.

A. Myopic Benchmark

A policy that chooses the action that minimizes the current or instantaneous cost is commonly referred to as myopic. It can be constructed as

$$\boldsymbol{\mu}(\boldsymbol{s}_t) = \arg \min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s}_t)} \left\{ \kappa(\boldsymbol{s}_t, \boldsymbol{a}) \right\}, \qquad (16)$$

with cost κ as in (10) and admissible action set as described in Sec. III-C. Notice that since the myopic policy does not contain trainable parameters, it does not need to be trained. Moreover, the myopic approach is carried out as a parameterless CFA-based policy, hence, becoming a particular case of (13). For this reason, it also admits a unique and closed-form evaluation. See Section C in the Appendix for more details.

VI. EXPERIMENTS

In this section, we experimentally validate the effectiveness of the proposed RNN-based policy, introduced in Sec. IV. To this end, we first describe the time-series datasets used. Then,

we outline the possible policy configurations, i.e., the possible types of splines as well as the RNN architecture. Afterward, we report how the experiments have been carried out. Finally, we present and comment on the experimental results.

A. Problem Data Description

For these experiments, we use a synthetic dataset and five real datasets. Each dataset consists of a time series of 28800 signal samples which has been split into 288 series of 100 samples each, except for the first real dataset which contains 57600 samples split into 576 series.

The synthetic dataset is first generated as a uniformly arranged realization of a given autoregressive process AR(2) with white Gaussian noise $\mathcal{N}(0, 0.1)$. Then, the resulting series is compressed via PI [52] with CompDev = 0.1, CompMax = ∞ and CompMin = 0. As a result, the series time stamps are not uniformly distributed anymore.

The first real dataset (R1) consists of a series of household minute-averaged active power consumption (in Kilowatts) [53]. The second real dataset (R2) is a quantized and PI-compressed (and hence not uniformly sampled) time series measuring an oil separation deposit pressure[3] (in Bar). For the third real dataset (R3) [54], a cooling fan with weights on its blades is used to generate vibrations which are recorded by an attached accelerometer. The vibration samples are recorded every 20 milliseconds. We use the accelerometer recorded x-values (which are standardized) for the rotation speeds ranging from 5 to 40 rpm. The fourth real dataset (R4) [55] monitors the skin temperature (in Celsius degrees) of a volunteer subject through a wearable device every 4 minutes. The fifth and last real dataset (R5) [56] consists of a sensor within a sensor network deployed in a lab, collecting the temperature-corrected relative humidity in percentage. The sampling rate is non-uniform and ranges from deciseconds to tens of seconds. Finally, it is worth mentioning that the datasets R4 and R5 contain gaps (several orders of magnitude wider than the average sampling period) of missing data that we have shortened to avoid instability in the reconstruction. In similar cases where the available raw data is of low quality, thorough and task-specific data preprocessing techniques are assumed. This can improve the performance results as described in the ensuing Sec. VI-D.

B. Policy Configuration

We experimentally observe that the myopic policy described in Sec. V-A is not stable for values of $\rho > 2$. Recall that the value of ρ affects the policy cost, set as in (10), and delimits the order and degree of smoothness of the spline signal estimate, as explained in Sec. II-C. We also observe instability under the myopic policy for $\rho = 2$ with a spline signal estimate of order $d = 3$ and degree of smoothness $\varphi = 2$. Consequently, our proposed RNN-based policy is unstable for the same ρ values and spline configurations since it implicitly uses the myopic policy as a guided starting point. This can be seen by comparing (16) and (13) with a near-zero initial value of λ. Although further

3Data collected from Lundin's offshore oil and gas platform Edvard-Grieg.

theoretical instability studies, alternative policy architectures, or low-delay approaches can contribute to solving the instability issue, they lie outside of the scope of this paper. Nonetheless, we have maintained the general problem formulation as a starting point for future works to take over. On the other hand, the interpolation problem with $\rho = 1$ is not interesting since it leads to linear interpolation. Therefore, in the present work, we focus on the smoothing interpolation problem with $\rho = 2$ and with the remaining stable spline configurations, within the search function space described in Sec. II-D, which can lead to optimal reconstructions. Those spline configurations, hereinafter specified by the shorthand notation (d, φ) of the order and degree of smoothness of the spline, correspond to (3,1) and (4,2). Accordingly, the notation Myopic(d, φ) or RNN(d, φ) refers to the type of policy besides the spline configuration.

Regarding the RNN architecture shaping the approximated cost-to-go within the RNN-based policy, introduced in Sec. IV-A and illustrated in Fig. 2, we set a preprocessing step that forwards the time length, i.e. $u_t = x_t - x_{t-1}$, of the tth time section $\mathcal{T}_t$, instead of directly using the time stamps. This preprocessing step makes the architecture invariant to time shifts in the set of time stamps. In our experiments, the recurrent unit consists of two stacked gated recurrent unit (GRU) layers [57], with a latent state (hidden state) of size 16 and an input of size 16. The input and output layers are set as linear layers to match the required dimensionality, i.e., to match the input size after the preprocessing step and to match the order of the spline minus the number of constrained coefficients as output size.

C. Experimental Setup

The datasets are randomly divided into 192 series for training, 64 for validation, and 32 for testing. Except for the R1 dataset, which has been divided in the same proportion but in relation to its data size. The benefits of this train-validation-test partition are two-fold: i) the policy becomes more robust against unknown initial conditions, and ii) we can validate the reconstruction against an optimal batch solution (shorter sequences are computationally tractable using batch optimization). All series within a dataset are standardized for implementation convenience. To avoid data leaking, the mean and standard deviation of their respective training partition are used for the standardization. In other words, we compute the mean and variance of the training partition and assume them to be the moments of the true data distribution. The standardization of series is useful to enforce the RNN unit to focus on the fluctuations of the signal values rather than on their magnitude. Finally, the RNN-based policy has been trained using the adaptive moments (Adam) optimizer [58], with $\beta_1 = 0.9$, $\beta_2 = 0.999$, without weight decay, and a learning rate of 0.001 over mini-batches of 32 time series each (double mini-batch size in the case of R1).

D. Results and Discussion

Some of the training-validation curves are presented in Fig. 4. As expected, we observe that randomly initialized
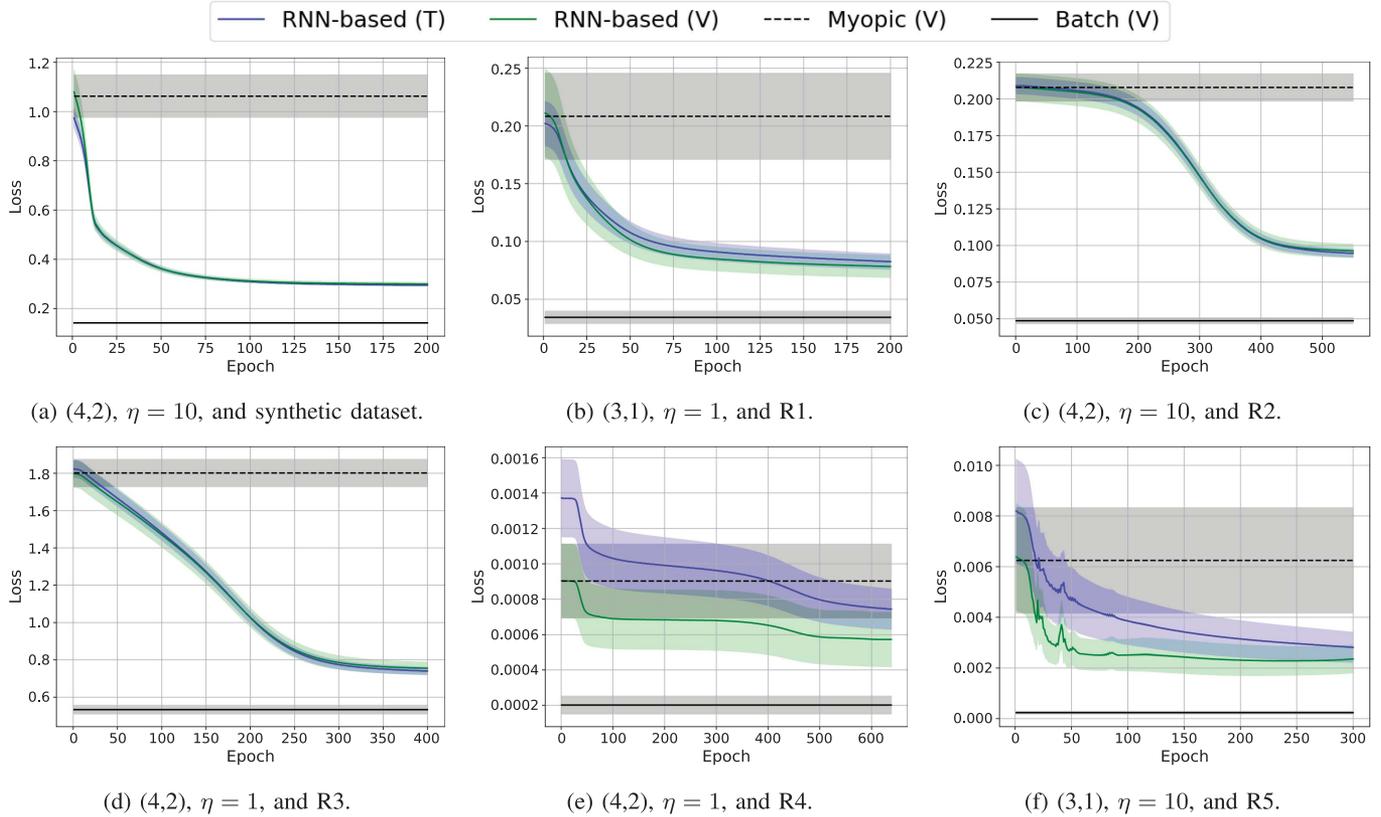
Fig. 4. Some of the training-validation curves for the considered RNN-based policy configurations (d, φ) and η values for each dataset. The legends (T) and (V) refer to the training and validation partitions, respectively. The loss metric is the average total cost per function section (see Sec. III-E). The shaded areas represent one standard deviation.

RNN-based policies (except for the parameter λ, which controls the length of the initial performance gap, as discussed in Sec. VI-B, and is manually initialized) only outperform the myopic policy after training. We also observe wider (in relative terms) standard deviations in those datasets with more abrupt changes, either from the nature of the data, as in R1, or due to missing data and posterior preprocessing, as in the case of R4 and R5. This phenomenon appears also to be caused by highly non-uniform sampling rates, as in R5. But in this case, the width seems to decrease as the policy yields more accurate estimates. This implies that the RNN-based policy is able to learn how to adapt under non-uniform sampling rates properly.

Once the RNN-based policy has been trained, we measure its performance with respect to the myopic policy (as the benchmark) and the batch reconstruction (as the baseline) through an *improvement* metric defined as

$$I = \frac{\ell_M - \ell_R}{\ell_M - \ell_B}, \quad (17)$$

where ℓ_M, ℓ_R, ℓ_B denote the loss metric displayed in Fig. 4 but over the test partition for the myopic, the RNN-based policies and the batch solution, respectively. Acceptable performances yield improvement values in $(0, 1]$, being $I = 1$ the best possible value, whereas nonpositive improvement values indicate a deficient performance. The standard deviation

of the improvement metric is then estimated through error propagation, i.e.,

$$\sigma_I = \sqrt{\left(\frac{\partial I}{\partial \ell_M}\right)^2 \sigma_M^2 + \left(\frac{\partial I}{\partial \ell_R}\right)^2 \sigma_R^2 + \left(\frac{\partial I}{\partial \ell_B}\right)^2 \sigma_B^2}, \quad (18)$$

with σ_M, σ_R, and σ_B denoting the standard deviation of the respective loss metrics over the test partition. The improvement results are summarized in Table I. From Fig. 4 and Table I, it can be observed that the policy configurations with the highest improvement scores over each of the considered dataset test partitions are in agreement with their corresponding validation curves. Table I also shows standard performance descriptors such as the mean squared error (MSE) and mean absolute error (MAE). See Section D in the Appendix for their computation. Note that for most of the experiments that we have carried out, the RNN-based policy outperforms, in terms of the MSE and MAE metrics, the myopic policy while it falls behind the batch policy. This observation experimentally justifies the smoothness assumption in our formulation.

Regarding the parameter $\lambda \in \mathbb{R}_+$ introduced in (14), it can be understood as the confidence of the RNN-based policy in its ability to foresee incoming data samples. In this way, it also quantifies the importance of the RNN architecture (detailed in Fig. 2) in the reconstruction task. As an illustration, Fig. 5 shows

TABLE I
PERFORMANCE METRICS AVERAGED OVER THE TEST PARTITIONS, FOR DIFFERENT η VALUES AND POLICY CONFIGURATIONS WITH $\rho = 2$. RECALL THAT THE IMPROVEMENT METRIC (17) REPORTS THE GAIN OF ANY RNN(d, φ) CONFIGURATION OVER ITS CORRESPONDING MYOPIC(d, φ) (AS A BENCHMARK) AND BATCH(d, φ) (AS THE BASELINE) CONFIGURATIONS

Dataset	Configuration	$\eta = 0.1$			$\eta = 1$			$\eta = 10$		
		MSE	MAE	Improvement	MSE	MAE	Improvement	MSE	MAE	Improvement
Synthetic	Myopic(3, 1)	0.49	0.60	71.5% ± 2.8%	0.31	0.46	51.1% ± 3.9%	0.64	0.63	78.1% ± 2.5%
	Batch(3, 1)	0.41	0.56		0.26	0.44		0.22	0.39	
	RNN(3, 1)	0.29	0.47		0.25	0.42		0.28	0.43	
	Myopic(4, 2)	0.50	0.60	64.8% ± 2.6%	0.38	0.50	53.0% ± 3.6%	0.83	0.71	81.0% ± 2.6%
	Batch(4, 2)	0.41	0.56		0.26	0.44		0.22	0.39	
	RNN(4, 2)	0.28	0.44		0.27	0.43		0.30	0.44	
R1	Myopic(3, 1)	0.09	0.13	59.4% ± 20.3%	0.20	0.22	81.3% ± 8.2%	0.44	0.40	75.5% ± 8.2%
	Batch(3, 1)	0.06	0.11		0.07	0.12		0.10	0.16	
	RNN(3, 1)	0.07	0.12		0.10	0.15		0.19	0.24	
	Myopic(4, 2)	0.13	0.16	59.6% ± 17.0%	0.29	0.27	78.6% ± 8.6%	0.51	0.43	72.9% ± 8.5%
	Batch(4, 2)	0.06	0.11		0.07	0.12		0.10	0.16	
	RNN(4, 2)	0.08	0.13		0.12	0.17		0.23	0.27	
R2	Myopic(3, 1)	0.42	0.50	67.3% ± 6.4%	0.40	0.48	52.2% ± 7.1%	0.32	0.42	68.8% ± 5.7%
	Batch(3, 1)	0.27	0.40		0.26	0.39		0.23	0.37	
	RNN(3, 1)	0.23	0.36		0.23	0.37		0.20	0.34	
	Myopic(4, 2)	0.65	0.62	70.1% ± 3.9%	0.58	0.59	52.0% ± 5.4%	0.39	0.47	73.5% ± 4.0%
	Batch(4, 2)	0.27	0.40		0.26	0.39		0.23	0.37	
	RNN(4, 2)	0.21	0.35		0.22	0.36		0.21	0.35	
R3	Myopic(3, 1)	4.60	1.80	76.3% ± 4.4%	2.90	1.40	81.3% ± 5.3%	1.80	1.09	65.6% ± 11.6%
	Batch(3, 1)	3.10	1.40		2.35	1.30		1.66	1.06	
	RNN(3, 1)	2.56	1.30		1.56	1.02		1.40	0.96	
	Myopic(4, 2)	6.30	2.10	69.7% ± 4.4%	3.02	1.43	88.9% ± 3.9%	1.89	1.10	80.8% ± 9.0%
	Batch(4, 2)	3.10	1.40		2.36	1.26		1.66	1.06	
	RNN(4, 2)	1.80	1.00		1.43	0.95		1.52	1.00	
R4	Myopic(3, 1)	4.5e-3	2.8e-2	13.7% ± 57.3%	4.1e-3	2.7e-2	−61.8% ± 94.4%	3.6e-3	2.5e-2	16.5% ± 54.5%
	Batch(3, 1)	2.7e-3	2.3e-2		2.6e-3	2.2e-2		2.4e-3	2.0e-2	
	RNN(3, 1)	7.1e-3	6.8e-2		6.0e-3	5.6e-2		3.2e-3	3.1e-2	
	Myopic(4, 2)	6.9e-3	3.6e-2	−35.5% ± 78.5%	5.7e-3	3.2e-2	65.7% ± 25.5%	4.9e-3	3.0e-2	19.5% ± 51.5%
	Batch(4, 2)	2.7e-3	2.3e-2		2.6e-3	2.2e-2		2.4e-2	2.0e-2	
	RNN(4, 2)	3.7e-3	3.3e-2		4.1e-3	3.1e-2		3.5e-3	2.7e-2	
R5	Myopic(3, 1)	2.7e-4	9.3e-3	−55.4% ± 101.6%	6.0e-4	1.3e-2	26.3% ± 59.8%	7.1e-3	3.8e-2	88.2% ± 7.1%
	Batch(3, 1)	1.8e-4	7.8e-3		1.7e-4	7.7e-3		2.6e-4	9.1e-3	
	RNN(3, 1)	2.3e-4	8.7e-3		3.1e-4	1.0e-2		3.2e-3	3.0e-2	
	Myopic(4, 2)	4.7e-4	1.1e-2	−13.3% ± 68.2%	9.7e-4	1.7e-2	39.8% ± 41.9%	1.1e-2	4.8e-2	85.1% ± 7.5%
	Batch(4, 2)	1.8e-4	7.8e-3		1.7e-4	7.7e-3		2.6e-4	9.1e-3	
	RNN(4, 2)	2.9e-4	9.7e-3		6.1e-4	1.4e-2		5.4e-3	3.6e-2	
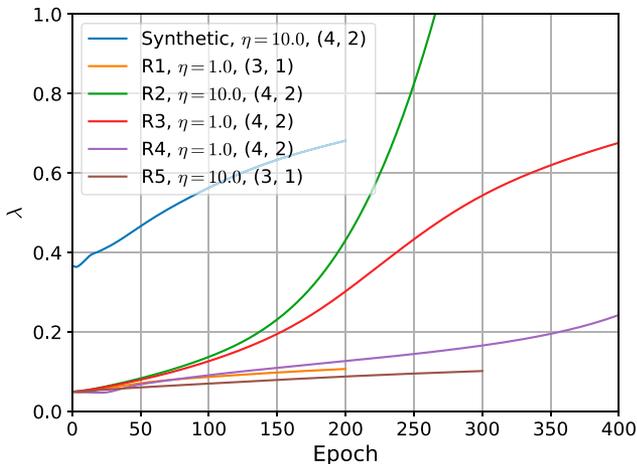
Fig. 5. Training curves of the parameter $\lambda \in \mathbb{R}_+$ introduced in (14). The elements displayed coincide with those shown in Fig. 4.

the training curves corresponding to the parameter λ for the policy configurations presented in Fig. 4.

On the other hand, we observe a competitive performance in terms of the execution time of the RNN-based policy evaluation (forward pass) as compared to their myopic counterpart. Our evaluation time results are summarized in Fig. 6, where the policies are implemented in Python 3.8.8. and the experiment is done in a 2018 laptop with a 2.7 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory. Regarding memory complexity, the myopic policy is parameterless (see Sec. V-A), and our configuration of the RNN-based policy (see Sec. VI-B) contains approximately 3400 trainable parameters, which is arguably a reduced model size for most tasks.

Finally, and for the sake of completeness, Fig. 7 shows a snapshot of a zero-delay smooth signal reconstruction alongside its two first derivatives using our proposed method.
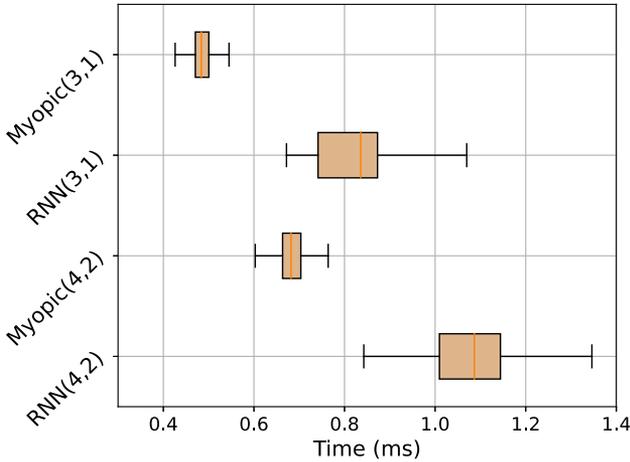
Fig. 6. Box plot of the policy execution time per interpolation step over the test partitions and η values $\{0.1, 1, 10\}$. It illustrates (excluding outliers) the minimum, first quartile, median, third quartile, and maximum.
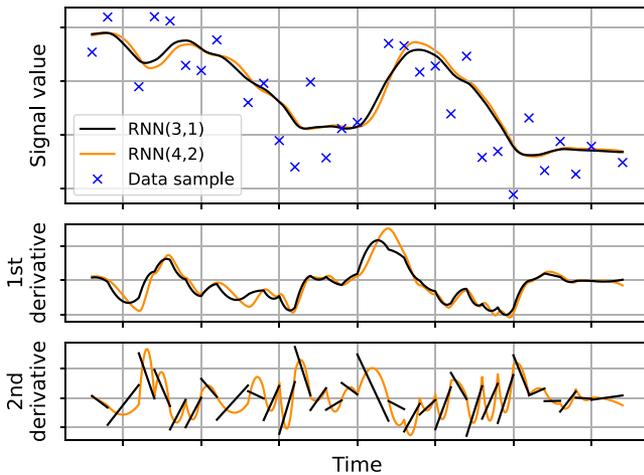
Fig. 7. Snapshot of a reconstructed time series from the test partition of the synthetic dataset. Both of the presented policies have been trained over the training partition before reconstruction.

VII. CONCLUSION

In this paper, we propose a method for zero-delay smoothing spline interpolation. Our method relies on a parametric policy, named the RNN-based policy, specifically engineered for the zero-delay interpolation task. As new data samples arrive, this policy yields piecewise polynomial functions used for smooth signal reconstruction. Our experiments show that the RNN-based policy can learn the dynamics of the target signal and efficiently incorporate them (in terms of improved accuracy and reduced response time) into the reconstruction task.

This work can be seen as a proof of concept with several immediate follow-ups. The flexibility in our policy design allows extending this work to multivariate time series with a moderate increase in complexity. It is also possible to generalize the problem data, e.g., quantization intervals instead of data points, as well as to accommodate additional constraints as long as the convexity of the policy evaluation problem is preserved. Lastly, we notice that our work provides the foundation and can

be tailored effectively for reconstructing non-stationary signals by borrowing reinforcement learning techniques.

APPENDIX

A. Proof of **Proposition 1**

Recall from Sec. II-C that every spline f_T, as in (1), is composed of T function sections and $T - 1$ contact points. We say that two consecutive function sections have a contact of order φ if they have φ equal derivatives at the contact point. Then, guaranteeing a degree of smoothness φ for a given spline f_T is equivalent to ensuring that all its contact points are at least of order φ since every tth function section g_t, as in (2), is already smooth over the interior of its domain $\mathcal{T}_t$. In practice, this can be ensured by imposing the following equality constraints

$$\lim_{x \to x_{t-1}^-} D_x^k\, g_{t-1}(x) = \lim_{x \to x_{t-1}^+} D_x^k\, g_t(x), \tag{19}$$

for every $k \in \mathbb{N}^{[0,\varphi]}$ and $t \in \mathbb{N}^{[2,T]}$. From here, notice that the kth derivative of every tth function section g_t can be computed as

$$D_x^k\, g_t(x) = \boldsymbol{a}_t^\top \left[D_x^k\, [\boldsymbol{p}_t(x)]_1, \dots, D_x^k\, [\boldsymbol{p}_t(x)]_{d+1} \right]^\top. \tag{20}$$

Also, notice from the definition in (3) that the ith component of the tth basis vector function $\boldsymbol{p}_t$ equals

$$[\boldsymbol{p}_t(x)]_i = (x - x_{t-1})^{i-1}, \tag{21}$$

for all $i \in [1, d+1]$. From this point, the kth derivative of each ith component of the basis vector function $\boldsymbol{p}_t$ can be straight-forwardly computed as

$$D_x^k\, [\boldsymbol{p}_t(x)]_i = (x - x_{t-1})^{i-1-k} \prod_{j=1}^{k} (i - j). \tag{22}$$

Now observe that

$$\lim_{x \to x_{t-1}^+} D_x^k\, [\boldsymbol{p}_t(x)]_i = \begin{cases} k! & \text{if } i = k+1, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Therefore, from the relations in (20) and (23), the right hand term in (19) can be equivalently computed as

$$\lim_{x \to x_{t-1}^+} D_x^k\, g_t(x) = \sum_{i=1}^{d+1} [\boldsymbol{a}_t]_i \lim_{x \to x_{t-1}^+} D_x^k\, [\boldsymbol{p}_t(x)]_i \tag{24a}$$

$$= k!\, [\boldsymbol{a}_t]_{k+1}. \tag{24b}$$

Separately, we can define a vector $\boldsymbol{e}_t \in \mathbb{R}^{\varphi+1}$ whose components are constructed as

$$[\boldsymbol{e}_t]_{k+1} \triangleq \frac{1}{k!} \lim_{x \to x_t^-} D_x^k\, g_t(x) \tag{25a}$$

$$= \frac{1}{k!} \sum_{i=1}^{d+1} [\boldsymbol{a}_t]_i\, u_t^{i-1-k} \prod_{j=1}^{k} (i - j), \tag{25b}$$

for every $k \in \mathbb{N}^{[0,\varphi]}$ and $t \in \mathbb{N}^{[1,T]}$ with $u_t \triangleq x_t - x_{t-1}$, and where the step (25b) uses the relations described in (20) and (22). On the other hand, $\boldsymbol{e}_0$ encodes the initial boundary conditions of the reconstruction and can be set by the user in advance

or calculated. Finally, by dividing both sides of the equality constraint in (19) by $k!$, using the relations derived in (24) and (25), and appropriately renaming the indices we obtain the **Proposition 1**.

B. Proof of Proposition 2

Recall from Sec. II-D that the solution to the optimization problem stated in (6) is a spline function in $\mathcal{W}_\rho$. This fact allows us to reduce the search function space without loss of optimality. In fact, we can incorporate the spline form of the solution into the objective functional as far as we ensure the required minimum degree of smoothness of the solution, for example, via (19). From here, we can equivalently compute the regularization term in the objective in (6) (second term) as

$$\int_{\bigcup_{t=1}^{T}\mathcal{T}_t} (D_x^\rho f_T(x))^2\, dx = \sum_{t=1}^{T} \int_{\mathcal{T}_t} (D_x^\rho g_t(x))^2\, dx. \quad (26)$$

Separately, and making use of the definition of function section in (2), we obtain the following relation

$$\int_{\mathcal{T}_t} (D_x^\rho g_t(x))^2\, dx = \int_{\mathcal{T}_t} \left(D_x^\rho \boldsymbol{a}_t^\top \boldsymbol{p}_t(x) \right)^2 dx \quad (27a)$$
$$= \boldsymbol{a}_t^\top \boldsymbol{M}_t \boldsymbol{a}_t, \quad (27b)$$

with

$$[\boldsymbol{M}_t]_{i,j} = \int_{\mathcal{T}_t} D_x^\rho [\boldsymbol{p}_t(x)]_i\, D_x^\rho [\boldsymbol{p}_t(x)]_j\, dx. \quad (28)$$

From the relation in (22), it is clear that the first ρ rows and columns of the matrix defined in (28) are zero valued. Then, we can compute the rest of the elements in the matrix $\boldsymbol{M}_t \in \boldsymbol{S}_+^{d+1}$ as follows

$$[\boldsymbol{M}_t]_{i,j} = \prod_{k=1}^{\rho} (i-k)(j-k) \int_{x_{t-1}}^{x_t} (x - x_{t-1})^{i+j-2(\rho+1)}\, dx \quad (29a)$$
$$= \frac{(x_t - x_{t-1})^{i+j-2\rho-1}}{i+j-2\rho-1} \prod_{k=1}^{\rho} (i-k)(j-k). \quad (29b)$$

On the other hand, the sum of squared residuals in the objective in (6) (first term) can be equivalently computed as

$$\sum_{t=1}^{T} (f_T(x_t) - y_t)^2 = \sum_{t=1}^{T} (g_t(x_t) - y_t)^2, \quad (30)$$

from the definition of spline, see relation (1).

Summing up, the result stated in **Proposition 2** can be reached starting from the objective in (6) then following the relations in (26), (27) alongside (29) and (30).

C. Closed-Form Policy Evaluation

Notice that both the proposed policy in (13) and the myopic policy in (16) can be equivalently evaluated by solving the following quadratic convex problem

$$\boldsymbol{\mu}(\boldsymbol{s}_t) = \arg\min_{\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s}_t)} \left\{ \boldsymbol{a}^\top \boldsymbol{A}_t \boldsymbol{a} + \boldsymbol{b}_t^\top \boldsymbol{a} \right\}, \quad (31)$$

TABLE II
TERMS IN (31). THE NOTATION IS SHARED WITH THE REST OF THE PAPER WITH THE INCORPORATION OF $\boldsymbol{P}_t \triangleq \boldsymbol{p}_t(x_t)\boldsymbol{p}_t(x_t)^\top$ AND $\boldsymbol{v}_t \triangleq [\boldsymbol{0}_{\varphi+1}^\top, \boldsymbol{r}_t^\top]^\top$

	$\boldsymbol{A}_t$	$\boldsymbol{b}_t$
Myopic	$\boldsymbol{P}_t + \eta \boldsymbol{M}_t$	$-2y_t \boldsymbol{p}_t(x_t)$
RNN	$\boldsymbol{P}_t + \eta \boldsymbol{M}_t + \lambda \boldsymbol{I}_{d+1}$	$-2(y_t \boldsymbol{p}_t(x_t) + \lambda \boldsymbol{v}_t)$

where the terms $\boldsymbol{A}_t \in \boldsymbol{S}_+^{d+1}$ and $\boldsymbol{b}_t \in \mathbb{R}^{d+1}$ take different values for the different policy variations as described in the Table II. The form in (31) is displayed as an intermediate step for the sake of clarity, and the dependencies with example time series (indexed by m) and the policy parameters (contained in $\boldsymbol{\theta}$) have been omitted for the sake of notation. Then, we relocate the equality constraints (presented in (4) and satisfied by the actions in the admissible set $\mathcal{A}(\boldsymbol{s}_t)$) in the objective of (31), by restating

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{e}_{t-1} \\ \boldsymbol{0}_{d-\varphi} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{\varphi+1} \\ \boldsymbol{\alpha} \end{bmatrix}, \quad (32)$$

or equivalently, by setting $\boldsymbol{a} = \boldsymbol{B}_1 \boldsymbol{e}_{t-1} + \boldsymbol{B}_2 \boldsymbol{\alpha}$ where the components of $\boldsymbol{e}_{t-1}$ are described in (5), the matrices $\boldsymbol{B}_1 \triangleq [\boldsymbol{I}_{\varphi+1}, \boldsymbol{0}_{(\varphi+1)\times(d-\varphi)}]^\top \in \mathbb{R}^{(d+1)\times(\varphi+1)}$ and $\boldsymbol{B}_2 \triangleq [\boldsymbol{0}_{(d-\varphi)\times(\varphi+1)}, \boldsymbol{I}_{d-\varphi}]^\top \in \mathbb{R}^{(d+1)\times(d-\varphi)}$ are defined for the sake of notation, and where $\boldsymbol{\alpha} \in \mathbb{R}^{d-\varphi}$. After some algebraic steps, both policies can be equivalently evaluated as

$$\boldsymbol{\mu}(\boldsymbol{s}_t) = \begin{bmatrix} \boldsymbol{e}_{t-1} \\ \boldsymbol{\alpha}_t \end{bmatrix}, \quad (33)$$

where the vector $\boldsymbol{\alpha}_t \in \mathbb{R}^{d-\varphi}$ is obtained from

$$\boldsymbol{\alpha}_t = \arg\min_{\boldsymbol{a}\in\mathbb{R}^{d-\varphi}} \left\{ \boldsymbol{\alpha}^\top \boldsymbol{B}_2^\top \boldsymbol{A}_t \boldsymbol{B}_2 \boldsymbol{\alpha} \right. \quad (34a)$$
$$\left. + 2 \left(\boldsymbol{e}_{t-1}^\top \boldsymbol{B}_1^\top \boldsymbol{A}_t \boldsymbol{B}_2 + \boldsymbol{b}_t^\top \boldsymbol{B}_2 \right) \boldsymbol{\alpha} \right\}, \quad (34b)$$

with closed-form solution given by

$$\boldsymbol{\alpha}_t = - \left(\boldsymbol{B}_2^\top \boldsymbol{A}_t \boldsymbol{B}_2 \right)^{-1} \left(\boldsymbol{B}_2 \boldsymbol{A}_t \boldsymbol{B}_1 \boldsymbol{e}_{t-1} + \frac{1}{2} \boldsymbol{B}_2^\top \boldsymbol{b}_t \right), \quad (35)$$

being $\boldsymbol{B}_2^\top \boldsymbol{A}_t \boldsymbol{B}_2 \in \boldsymbol{S}_{++}^{d-\varphi}$.

D. Boostrap Method for Estimating the MSE and MAE

When the data distribution, or in our case, the underlying (assumed) smooth process, is unknown, we cannot follow the standard MSE and MAE computation procedure because the original function $\psi \in \mathcal{W}_\rho$ is also unknown. Instead, we only have access to a certain dataset of test samples, e.g., $\mathcal{D} = \{(x_t, \psi(x_t))\}_{t=1}^{T}$. Following a bootstrap-inspired method [59], we choose a subset $\mathcal{B} \subseteq \mathcal{D}$ for the signal reconstruction and use the complementary set $\overline{\mathcal{B}}$, i.e., $\mathcal{B} \cup \overline{\mathcal{B}} = \mathcal{D}$ and $\mathcal{B} \cap \overline{\mathcal{B}} = \emptyset$ to estimate the MSE and MAE performance metrics. Mathematically, this can be expressed as

$$\text{MSE}(f_\mathcal{B}) = \frac{1}{|\overline{\mathcal{B}}|} \sum_{i\in\overline{\mathcal{B}}} (f_\mathcal{B}(x_i) - \psi(x_i))^2, \quad (36a)$$
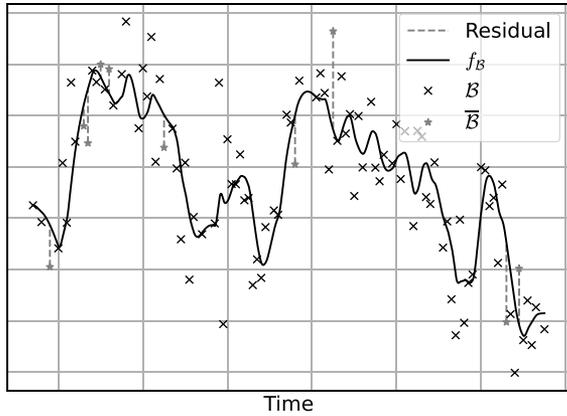
Fig. 8. Illustration of the residuals used to estimate the MSE via bootstrapping. The test partition comes from the synthetic data and has been reconstructed with a Myopic(3,1) policy with $\eta = 1$ and $\rho = 2$. $\mathcal{B}$ contains 90% of the test partition and $\overline{\mathcal{B}}$ the remaining 10%.

$$\text{MAE}\,(f_{\mathcal{B}}) = \frac{1}{|\overline{\mathcal{B}}|} \sum_{i \in \overline{\mathcal{B}}} |f_{\mathcal{B}}(x_i) - \psi(x_i)|, \qquad (36b)$$

where $f_{\mathcal{B}}$ is the signal estimate constructed from the test data subset $\mathcal{B}$. This procedure is illustrated in Fig. 8.

Notice that it is important to partition the test data because any test data sample used for the signal reconstruction cannot be used to compute the performance metrics. Otherwise, this results in data leakage. On the other hand, due to the lack of data samples, the performance metrics estimated in this way may not be as accurate as if we had larger test sets or, more specifically, large test sets with higher temporal resolution. Thus, to reduce the variance of the MSE and MAE estimators, we repeat the procedure for several randomly chosen partitions $\mathcal{B}, \overline{\mathcal{B}}$ with replacement (i.e., they may repeat) and average the result. Particularly, we perform 10 repetitions.

REFERENCES

[1] R. L. Plackett, "Some theorems in least squares," *Biometrika*, vol. 37, no. 1/2, pp. 149–157, 1950.

[2] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. Hoboken, NJ, USA: Wiley, 1996.

[3] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[4] C. De Boor and C. De Boor, *A Practical Guide to Splines*, vol. 27. New York, NY, USA: Springer-Verlag, 1978.

[5] S. Shalev-Shwartz et al., "Online learning and online convex optimization," *Found. Trends® Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.

[6] E. Hazan et al., "Introduction to online convex optimization," *Found. Trends® Optim.*, vol. 2, no. 3–4, pp. 157–325, 2016.

[7] F. Orabona, "A modern introduction to online learning," 2019, *arXiv:1912.13213*.

[8] A. Uncini, *Fundamentals of Adaptive Signal Processing*. Cham, Switzerland: Springer, 2015, pp. 287–347.

[9] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4. New York, NY, USA: Springer, 2006, p. 738.

[10] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Belmont, MA, USA: Athena Scientific, 2012.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[12] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[13] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, 2016, pp. 7195–7201.

[14] R. Nishihara et al., "Real-time machine learning: The missing pieces," in *Proc. 16th Workshop Hot Topics Operating Syst.*, 2017, pp. 106–110.

[15] O. Kosheleva and V. Kreinovich, "Why physical processes are smooth or almost smooth: A possible physical explanation based on intuitive ideas behind energy conservation," Univ. Texas, El Paso, TX, USA, Technical Report, 2021. [Online]. Available: https://scholarworks.utep.edu/cs_techrep/1513/

[16] H. S. Wilf, *Algorithms and Complexity*. New York, NY, USA: AK Peters/CRC Press, 2002.

[17] S. A. Bazaz and B. Tondu, "Minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators," *Robot. Auton. Syst.*, vol. 29, no. 4, pp. 257–268, 1999.

[18] T. Kröger, *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*, vol. 58. Berlin, Germany: Springer, 2010.

[19] J. NILSSON, "Real-time control systems with delay," Ph.D. thesis, Lund Inst. Technol., Uppsala, Sweden, 1998.

[20] A. Gambier, "Real-time control systems: A tutorial," in *Proc. 5th Asian Control Conf. (IEEE Cat. No. 04EX904)*, vol. 2, Piscataway, NJ, USA: IEEE, 2004, pp. 1024–1031.

[21] C. Schmidt, C. Kottke, V. Jungnickel, and R. Freund, "High-speed digital-to-analog converter concepts," in *Proc. Next-Gener. Opt. Commun.: Compon., Sub-Syst., Syst. VI*, vol. 10130, Bellingham, WA, USA: SPIE, 2017, pp. 133–141.

[22] B. Schölkopf et al., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[23] E. Ruiz-Moreno and B. Beferull-Lozano, "Tracking of quantized signals based on online kernel regression," in *Proc. IEEE 31st Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Piscataway, NJ, USA: IEEE, 2021, pp. 1–6.

[24] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Berlin, Germany: Springer, 2004, pp. 63–71.

[25] Q. Lu, G. V. Karanikolas, and G. B. Giannakis, "Incremental ensemble Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1876–1893, Feb. 2023.

[26] A. Blum and Y. Monsour, "Learning, regret minimization, and equilibria," in *Algorithmic Game Theory*. Cambridge, UK: Cambridge University Press, 2007.

[27] I. J. Schoenberg, "Contributions to the problem of approximation of equidistant data by analytic functions," in *IJ Schoenberg Selected Papers, Contemporary Mathematicians*, C. de Boor, Ed. Boston, MA, USA: Springer, 1988, pp. 3–57.

[28] L. Schumaker, *Spline Functions: Basic Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[29] E. Meijering, "A chronology of interpolation: From ancient astronomy to modern signal and image processing," *Proc. IEEE*, vol. 90, no. 3, pp. 319–342, Mar. 2002.

[30] E. Waring, "Vii. problems concerning interpolations," *Philos. Trans. R. Soc. London*, no. 69, pp. 59–67, 1779.

[31] G. Wahba, *Spline Models for Observational Data*. Philadelphia, PA, USA: SIAM, 1990.

[32] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Process. Mag.*, vol. 16, no. 6, pp. 22–38, Nov. 1999.

[33] M. Unser and T. Blu, "Generalized smoothing splines and the optimal discretization of the wiener filter," *IEEE Trans. Signal Process.*, vol. 53, no. 6, pp. 2146–2159, Jun. 2005.

[34] J. M. de Carvalho and J. V. Hanson, "Real-time interpolation with cubic splines and polyphase networks," *Can. Elect. Eng. J.*, vol. 11, no. 2, pp. 64–72, 1986.

[35] R. Debski, "Real-time interpolation of streaming data," *Comput. Sci.*, vol. 21, no. 4, pp. 513–532, 2020.

[36] K. Frankish and W. M. Ramsey, *The Cambridge Handbook of Artificial Intelligence*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[37] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[38] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[39] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2017, *arXiv:1801.01078*.

[40] C. H. Reinsch, "Smoothing by spline functions," *Numer. Math.*, vol. 10, no. 3, pp. 177–183, 1967.

[41] S. N. Wood, *Generalized Additive Models: An Introduction With R*. Boca Raton, FL, USA: Chapman and Hall/CRC, 2006.

[42] W. Fan, C.-H. Lee, and J.-H. Chen, "A realtime curvature-smooth interpolation scheme and motion planning for CNC machining of short line segments," *Int. J. Mach. Tools Manuf.*, vol. 96, pp. 27–46, 2015.

[43] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[44] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[46] L. Peshkin, N. Meuleau, and L. Kaelbling, "Learning policies with external memory," 2001, *arXiv:cs/0103003*.

[47] A. M. Schäfer, "Reinforcement learning with recurrent neural networks," Ph.D. thesis, Osnabrück, Univ., Diss., Osnabrück, 2008.

[48] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural network policies with continuous memory states," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Piscataway, NJ, USA: IEEE, 2016, pp. 520–527.

[49] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[50] A. Paszke et al., "Automatic differentiation in PyTorch," 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[51] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, 2021.

[52] OSIsoft, "Pi system," 1980. [Online]. Available: https://www.osisoft.com/pi-system

[53] G. Hebrail and A. Berard, "Individual household electric power consumption," UCI Mach. Learn. Repository, Irvine, CA, USA, 2012.

[54] G. S. Sampaio, A. R. de Aguiar Vallim Filho, L. S. da Silva, and L. A. da Silva, "Prediction of motor failure time using an artificial neural network," *Sensors*, vol. 19, Oct. 2019, Art. no. 4342.

[55] J. Huan et al., "A wearable skin temperature monitoring system for early detection of infections," *IEEE Sensors J.*, vol. 22, no. 2, pp. 1670–1679, Jan. 2022.

[56] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel berkeley research lab data," 2004. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

[57] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proc. 8th Workshop Syntax, Semantics Struct. Statistical Transl. (SSST-8)*, 2014, pp. 103–111.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[59] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Boca Raton, FL, USA: CRC Press, 1994.

Emilio Ruiz-Moreno received the bachelor's degree in physics from Universidad de Valencia, Valencia, Spain, in 2016, and the master's degree in mechatronics (top-ranked student) from Universidad Politécnica de Valencia, Valencia, Spain, in 2018. Since August 2018, he has been pursuing the Ph.D. degree in artificial intelligence with the WISENET Research Center, Universitetet i Agder, Grimstad, Norway. His research interests include optimization and machine learning.

Luis Miguel López-Ramos (Member, IEEE) received the M.Sc. and Ph.D. (Extraordinary Doctoral Award) degrees in multimedia and communications jointly from the King Juan Carlos University and Carlos III University of Madrid, Spain, in 2012 and 2016, respectively. Between 2017 and 2021, he was with the Department of Information and Communication Technology, University of Agder, Grimstad, Norway, as a Postdoctoral Research Fellow with the WISENET Center. Since 2022, he has been with the Simula Metropolitan Center for Digital Engineering, Oslo, Norway, as a Postdoctoral Researcher with the Holistic Systems Department. His general research interests include the broad areas of signal processing, machine learning, optimization, and explainable AI. His current research topics are online algorithms for inference with low delay, interpretable modeling of multiple time series, and explanation methods.

Baltasar Beferull-Lozano (Senior Member, IEEE) received the M.Sc. degree (First-in-Class Honours) in physics from Universidad de Valencia, Spain, in 1995, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Southern California (USC), Los Angeles, in 1999 and 2002, respectively. In October 2002, he joined the Audio-Visual Communications Laboratory, Department of Communication Systems, at EPFL, as a Research Associate, where he spent around three years. In January 2006, he joined the School of Engineering at the University of Valencia as an Associate Professor. Since August 2014, he has been a Professor with the Department of Information and Communication Technology and (by courtesy) the Department of Engineering, University of Agder, Norway, where he leads the Center Intelligent Signal Processing and Wireless Networks (WISENET). Since April 2021, he has been also a Chief Research Scientist and Research Professor with the Simula Metropolitan Center for Digital Engineering (SimulaMet), Norway, where he leads the Department of Signal and Information Processing for Intelligent Systems (SIGIPRO). He has been serving as a Senior Area Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING since 2016. He has also served as a member of the Technical Program Committees for several ACM & IEEE international conferences. His research interests are in the general areas of data science and machine learning, graph signal processing, in-network signal processing and collective intelligence, optimization, networked cyber-physical systems, and artificial intelligence for next generation wireless networks. At USC, he received several awards, including the Best Ph.D. Thesis Paper Award in 2002 and the Outstanding Academic Achievement Award in 1999. He has received the Best Paper Awards at several international conferences, such as IEEE DCOSS and IEEE DSLW. He has also received a TOPPFORSK Grant Award from the Research Council of Norway, 2015. He is a member of the Norwegian Academy of Science and Technology.