Machine Learning for Signal Reconstruction from Streaming Time-series Data

Emilio Ruiz-Moreno

UiA University of Agder

Doctoral Dissertations at the University of Agder 569

Emilio Ruiz-Moreno

Machine Learning for Signal Reconstruction from Streaming Time-series Data

Doctoral Dissertation for the Degree *Philosophiae Doctor (Ph.D.)*
at the Faculty of Engineering and Science, Specialisation in ICT

University of Agder
Faculty of Engineering and Science
2024

Doctoral Dissertations at the University of Agder 456
ISSN: 1504-9272
ISBN: 978-82-8427-173-6

©Emilio Ruiz-Moreno, 2024

Printed by Make!Graphics
Kristiansand

Preface

This Ph.D. thesis results from the research conducted at the the WISENET Center, Department of Information and Communication Technology, Universitetet i Agder (UiA), Grimstad, Norway, from August 2018 to September 2023. The corresponding Ph.D. degree has been supervised by Professor Baltasar Beferull-Lozano, and co-supervised by Dr. Luis Miguel López-Ramos, and Professor Jing Zhou.

During my Ph.D. program, I have combined research work at the SFI Offshore Mechatronics Centre for Research-based Innovation, from September 2018 to August 2022, under grant 237896/O30, with teaching duties at UiA from January 2020 to August 2022. Moreover, in addition to the Ph.D. coursework components, I attended the machine learning crash course (MLCC) at Università di Genova, Italy, in 2019; the DeepLearn 2021 Summer at Las Palmas de Gran Canaria, Spain, in 2021; and the finance track of the Oxford Machine Learning Summer School (OxML) at the University of Oxford, England, in 2022.

Acknowledgments

My doctoral journey has been nothing short of a thrilling quest, filled with new experiences and challenges that have enriched my understanding of the world and field of study. Along this path, I have been fortunate to encounter mentors, colleagues, and friends who have aided me in my academic endeavors and have played pivotal roles in shaping the essence of this thesis. So, let me take this opportunity to express my heartfelt gratitude to all those who have journeyed with me.

First and foremost, I would like to extend my gratitude to my supervisor Baltasar for his exceptional guidance and unwavering support. His mentorship has been a cornerstone of my academic growth, and I am truly fortunate to have had the privilege of working with him. I would also like to thank my co-supervisor Luismi, whose profound domain knowledge and trust in my academic skills have been instrumental in the research presented in this thesis. Beyond the academic realm, Baltasar and Luismi's kindness has even made this stiff scholarly pursuit enjoyable.

Thanks to everyone at the WISENET Center, old and new colleagues, for providing the warmest working environment I could have asked for. Especially to my core office mates Siavash, Ravi, and Juan Diego for all the shared moments. I will keep filling whiteboards with souvenir magnets for old times' sake.

Many thanks to all the friends I have made along the way. All the board games, nights at one of the three pubs in Grimstad, and moments lived together hold a special place in my memories and have made these past years unforgettable.

I want to thank my friend Alfredo; I am sure I could not have had a better housemate.

I want also to thank my family and friends back home. To my parents and sisters for their immeasurable love and support, to Cheto and Bambi for their dog affection, and to my lifelong friends for always being there for me despite time and distance.

Last but not least, to my partner Eleni. Thank you for your patience, continuous encouragement, and help throughout the last few years.

Emilio Ruiz-Moreno
Oslo, Norway
January 24, 2024

Abstract

Nowadays, deploying cyber-physical networked systems generates tremendous streams of data, with data rates increasing as time goes by. This trend is especially noticeable in several fairly automated sectors, such as energy or telecommunications. Compared to the last decades, this represents not only an additional large volume of data to explore and the need for more efficient and scalable data analysis methods but also raises additional challenges in the design and analysis of real-time streaming data processing algorithms. In many applications of interest, it is required to process a sequence of samples from multiple, possibly correlated, data time series that are acquired at different sampling rates and which may be quantized in amplitude at different resolutions. A commonly sought goal is to obtain a low-error signal reconstruction that can be uniformly resampled with a temporal resolution as fine as desired, hence facilitating subsequent data analyses.

This Ph.D. thesis consists of a compendium of four papers that incrementally investigate the task of sequentially reconstructing a signal from a stream of multivariate time series of quantization intervals under several requirements encountered in practice and detailed next.

First, we investigate how to track signals from streams of quantization intervals while enforcing low model complexity in the function estimation. Specifically, we explore the use of reproducing kernel Hilbert space-based online regression techniques expressly tailored for such a task. More specifically, the core techniques we devise and employ are influenced by the abundant theoretical and practical benefits in the literature about proximal operators and multiple kernel approaches.

Second, we require the signal to be sequentially reconstructed, subject to smoothness constraints, and as soon as a data sample is available (zero-delay response). These well-motivated requirements appear in many practical problems, including online trajectory planning, real-time control systems, and high-speed digital-to-analog conversion. We address this challenge through a novel spline-based approach underpinned by a sequential decision-making framework and assisted with deep learning techniques. Specifically, we use recurrent neural networks to capture the temporal dependencies among data, helping to reduce the roughness of the reconstruction on average.

Finally, we analyze the requirement of consistency, which amounts to exploiting all available information about the signal source and acquisition system to optimize some figure of reconstruction merit. In our context, consistency means guaranteeing that the reconstruction lies within the acquired quantization intervals. Consistency has been proven to entail a profitable-in-practice asymptotic error-rate decay as the sampling rate increases. Particularly, we investigate the impact of consistency on zero-delay reconstruction and also incorporate the idea of exploiting the spatiotemporal dependencies among multivariate signals.

Sammendrag

I dag genererer utplassering av cyberfysiske nettverkssystemer enorme datastrømmer, med datahastigheter som øker etter hvert som tiden går. Denne trenden er spesielt merkbar i flere ganske automatiserte sektorer, som energi eller telekommunikasjon. Sammenlignet med de siste tiårene representerer dette ikke bare et ekstra stort volum av data å utforske og et behov for mer effektive og skalerbare dataanalysemetoder, men reiser også ytterligere utfordringer i utformingen og analysen av sanntids strømnings databehandling algoritmer. I mange applikasjoner av interesse er det nødvendig å behandle en sekvens av prøver fra flere, muligens korrelerte, datatidsserier som er innhentet ved forskjellige samplingshastigheter og som kan kvantiseres i amplitude ved forskjellige oppløsninger. Et vanlig mål er å oppnå en rekonstruksjon med lav feilsignal som kan uniformt gjensamples med en så fin temporal oppløsning som ønsket, og dermed fasilitere etterfølgende dataanalyser.

Denne Ph.D. avhandlingen består av et kompendium av fire artikler som inkrementelt undersøker oppgaven med å sekvensielt rekonstruere et signal fra en strøm av multivariate tidsserier av kvantiseringsintervaller under flere krav som møtes i praksis og detaljert deretter.

Først undersøker vi hvordan vi kan spore signaler fra strømmer av kvantiseringsintervaller mens vi håndhever lav modellkompleksitet i funksjonsestimeringen. Spesielt utforsker vi bruken av reproducing kernel Hilbert-rombaserte online regresjonsteknikker som er spesielt skreddersydd for en slik oppgave. Mer spesifikt er kjerneteknikkene vi utarbeider og bruker påvirket av de mange teoretiske og praktiske fordelene i litteraturen om proksimale operatører og flere kjernetilnærminger.

For det andre krever vi at signalet rekonstrueres sekvensielt, underlagt jevnhetsbegrensninger, og så snart en dataprøve er tilgjengelig (null-forsinkelsesrespons). Disse godt motiverte kravene dukker opp i mange praktiske problemer, inkludert online baneplanlegging, sanntidskontrollsystemer og høyhastighets digital-til-analog konvertering. Vi takler denne utfordringen gjennom en ny spline-basert tilnærming underbygget av et sekvensielt beslutningsrammeverk og assistert med dyplæringsteknikker. Spesifikt bruker vi recurrent nevrale nettverk for å fange opp de tidsmessige avhengighetene mellom data, og bidrar til å redusere grovheten til rekonstruksjonen i gjennomsnitt.

Til slutt analyserer vi kravet om konsistens, som utgjør å utnytte all tilgjengelig informasjon om signalkilden og innsamlingssystemet for å optimalisere noen form for rekonstruksjonsverdi. I vår sammenheng betyr konsistens å garantere at rekonstruksjonen ligger innenfor de innhentede kvantiseringsintervallene. Konsistens har vist lønnsom praktisk asymptotisk fofall av feilraten når samplingsfrekvensen øker. Spesielt undersøker vi virkningen av konsistens på null-forsinkelsesrekonstruksjon og inkluderer også ideen om å utnytte de spatiotemporale avhengighetene blant multivariate signaler.

Publications

Paper A **E. Ruiz-Moreno** and B. Beferull-Lozano, "Tracking of quantized signals based on online kernel regression," in *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1-6, IEEE, 2021.*

Paper B **E. Ruiz-Moreno** and B. Beferull-Lozano, "An online multiple kernel parallelizable learning scheme," *IEEE Signal Processing Letters*, 2023.

Paper C **E. Ruiz-Moreno**, L. M. López-Ramos and B. Beferull-Lozano, "A trainable approach to zero-delay smoothing spline interpolation," *IEEE Transactions on Signal Processing*, vol. 71, pp. 4317-4329, 2023.

Paper D **E. Ruiz-Moreno**, L. M. López-Ramos and B. Beferull-Lozano, "Consistent signal reconstruction from streaming multivariate time series," submitted to *IEEE Transactions on Signal Processing.*

The following paper was published during the Ph.D. degree but is not included in the thesis.

S. Mollaebrahim, B. Beferull-Lozano and **E. Ruiz-Moreno**, "Decentralized subspace projection for asymmetric sensor networks," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall).*

Contents

List of Figures

List of Tables

Abbreviations

Adam	Adaptive moments
ADC	Analog-to-digital converter
AR	Autoregressive
BPTT	Backpropagation through time
CC	Cumulative cost
CFA	Cost function approximation
DCOL	Differentiable convex optimization layer
DP	Dynamic programming
ERM	Empirical risk minimization
GRU	Gated recurrent unit
KAPSM	Kernel adaptive projected subgradient method
LSTM	Long short-term memory
MAE	Mean absolute error
MSE	Mean square error
NORMA	Naive online R_{reg} minimization algorithm
RERM	Regularizaed empirical risk minimization
RK	Reproducing kernel
RKHS	Reproducing kernel Hilbert space
VAR	Vector autoregressive
WORM	Windowed online regularized cost minimization

Chapter 1

Introduction

The study of temporal measurements is a matter of great importance in many domains of applied science and engineering, such as statistics or signal processing. In this context, *time series* are the object of study, consisting of a sequence of data samples or observations indexed in time order. Accordingly, the study techniques are usually referred to as methods for time-series analysis.

This chapter motivates the importance of methods for time series analysis, reveals some related research gaps, and discusses possible research avenues to address them.

1.1 Motivation

Streaming data can be understood as a time series whose terms are continuously generated, possibly from different sources. Streaming data is of paramount value and necessity in the present day, being applied in a broad range of industries, including finance, transportation, or the industrial Internet of Things, among others [15].

In many practical tasks involving streaming data, the ground truth behind the process being measured is unknown. Instead, one only has access to the received time series observations. These observations may be very different in nature for each task, for instance, real-valued, discrete, or even symbolic data.
When dealing with physical phenomena, the observed data samples are collected through a data *acquisition system*, usually comprised of a set of networked sensors and a computing unit. In this way, and due to the acquisition system limitations, the resulting observations typically consist of signal values sampled in time and quantized in amplitude, not necessarily uniformly for both.
A common goal for these tasks is to recover the ground truth process from the sequence of observations, for example, via some function approximation method, with the lowest possible error.

In most cases of interest, the required methods must account for large data volumes and, hence, be able to run online, sometimes under real-time constraints. Such constraints become prevalent when a safety-critical application requires instantaneous decisions based on the latest received data. Also, highly-competitive settings, such as algorithmic trading [16], give an advantage to those methods that minimize the latency between signal acquisition and order execution. In user-facing applications, avoiding delay is positive for user experience and system usability. For instance, virtual reality devices rely on real-time signal processing to ensure a seamless and immersive experience [17]. A fast response of actuators based on sensor

input is also beneficial on board drones, power grids, or wind turbines.

On the other hand, these methods must be suitable for time series, possibly multiple ones concurrently, of misaligned, non-uniformly sampled, and quantized intervals. This is because patterns from connected systems can improve processing results. Moreover, misaligned samples occur frequently, for example, in information fusion from different sensors [18] or when synchronization between devices is challenging.

Finally, and whenever possible, it is convenient that these methods exploit prior information about the underlying physical process, such as the smoothness and low model complexity shared by most physical signals and their spatiotemporal dynamics, to reduce the reconstruction error. Keeping the smooth nature and complexity of the signal reconstruction low is important to avoid artifacts or misleading interpretations that worsen its generality over unseen data.

1.2 Summarized State-of-the-art

Online optimization methods for signal reconstruction problems generating a series of function estimates from streaming data are useful to reduce computational complexity in large-scale problems [19], to dynamically adapt to new patterns in the data [20], and also to enable acting under real-time constraints [21].

Among them, online kernel-based penalized regression approaches stand out in popularity, mostly because they allow estimating functions of arbitrary complexity by finding non-linear patterns at a moderate computational cost [12, 22]. Within the considered framework of streamed time series, several sliding window kernel-based methods have been proposed to increase the reliability of the signal estimation by processing consecutive observations concurrently.

Arguably, the most suited approach to tackle our posed problem is the one presented in [23], which can provide a low-delay response by running a single processing iteration over the quantized data samples within a sliding window every time an observation is received.

However, it cannot guarantee that the estimated signal passes through all the observed quantization intervals, thus yielding function estimates inconsistent with the acquisition system. Another issue is that explicit regularization is not included, and therefore, the reduced model complexity associated with smooth and bounded underlying physical signals is not fully promoted. Lastly, function estimates modeled as a kernel expansion make it challenging to guarantee smoothness in a sequentially reconstructed signal; thus, using them as tracking systems rather than for smooth signal reconstructions is preferable.

On the other hand, online interpolation methods can be suitable candidates for overcoming the aforementioned issues. These methods typically use piecewise-defined functions to model a sequence of local signal estimates [24, 6]. Piecewise-modeled signal estimates can be updated every time an observation is received by assembling a new function piece while guaranteeing smoothness and a low-complexity model.

Several studies have explored the use of *spline*-based online interpolation methods from streaming data under real-time constraints [25, 26]. However, most of the existing spline-based online interpolation methods require a multi-step lookahead or sliding window mechanism, leading to a delay.

To the best of our knowledge, the only existing spline-based *zero-delay* interpolation method in the literature is the myopic approach (referred to as the "classical greedy approach" in [26]), a purely local method; thus, it does not exploit nonlocal information for improved signal reconstruction. For instance, it cannot accommodate the possible long-term spatiotemporal relationships within any multivariate time series.

Finally, consistent signal reconstruction methods exploit all available knowledge about the underlying physical signal, such as the signal class (e.g., bandlimited signals) and the acquisition process (e.g., analog-to-digital converters), to improve the error-rate decay as the sampling rate increases. These have been proven useful under offline settings [27, 28]. However, there is a research gap regarding online consistent signal reconstruction methods.

1.3 Research Questions

The motivation for this thesis work can be summarized in the following research questions derived from the current State-of-the-art:

RQ1 Can we design an online kernel-based method to reliably track the underlying physical signal from a univariate time series of quantization intervals?

RQ2 In addition to satisfying the low model complexity (implicitly imposed) in **RQ1**, if we require a more strict zero-delay response constraint and smoothness in the reconstructed signal, how can we exploit the temporal dependencies of univariate time series of data points to achieve an accurate spline-based signal reconstruction?

RQ3 Besides the previous requirements in **RQ1** and **RQ2**, how can we ensure a consistent signal reconstruction while exploiting the spatiotemporal relations of multivariate time series of quantization intervals to achieve an accuracy-improved spline-based signal reconstruction?

1.4 Outline

This Ph.D. thesis consists of a compendium of four papers. These papers have been either published or submitted for publication in peer-reviewed international conference proceedings and journals. They can be found in the appendices of this thesis. The thesis is divided into five chapters, including the present one.

Chapter 1 introduces the thesis topic, motivates its importance, summarizes the state-of-the-art, and exposes several research gaps and possible research paths to address them. The key theoretical concepts in the compendium of papers are discussed in **Chapter 2**. Next, **Chapter 3** summarizes papers A and B, which address the **RQ1**. Similarly, **Chapter 4** summarizes papers C and D, addressing the **RQ2** and the **RQ3**, respectively. Finally, **Chapter 5** concludes the paper and presents the future work.

Chapter 2

Background Theory

This thesis work makes use of several machine-learning techniques as auxiliary tools to attain real-time signal reconstruction from streaming data. This background theory chapter is meant to cover the basic idea behind those techniques, serving as a prelude for the compendium of papers attached in the appendices.
Accordingly, the connection, in the context of this Ph.D. thesis, between the concepts discussed next is purposely omitted for the sake of conciseness and is instead established and detailed in the ensuing thesis chapters.

2.1 Definitions and Terminology

This section covers key definitions and technical terminology used in this Ph.D. thesis. Specifically, some concepts related to acquisition systems, such as quantization or, more generally, uncertainty intervals, and some others related to signal reconstruction methods, such as delay, smoothness, and spline models.

2.1.1 Quantization

In signal processing, *quantization* is the process of partitioning the image domain of a given signal by mapping measured signal values, often from a continuous set, to values in a countable smaller set, possibly finite [29]. It is commonly used to reduce signal data size for more efficient storing, faster processing, and transmission.

Any device or algorithm that performs quantization can be referred to as a quantizer. For example, a uniform mid-tread quantizer, which quantizes the signal values at any tth time stamp $\psi(x_t)$, with a quantization stepsize Δ, as

$$y_t = \text{round}\left(\frac{\psi(x_t)}{\Delta}\right)\Delta, \tag{2.1}$$

where round() denotes the round function. In this case, the tuple x_t, y_t, and Δ can describe any of the resulting quantized signal values.

2.1.2 Uncertainty intervals

An *uncertainty interval* can be defined as a continuous region in the image domain of the underlying signal being acquired that reflects our confidence, or knowledge,

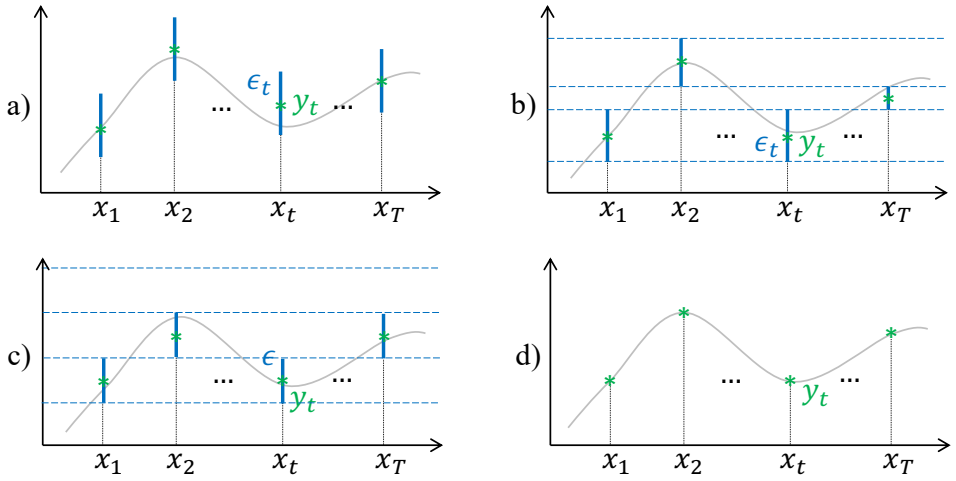


Figure 2.1: Example streams of uncertainty intervals. In a), we have intervals of different sizes that can overlap (in the image domain), thus not necessarily partitioning the image domain; an example of such data is safety regions in a trajectory curve. In b) and c), we respectively show a stream of non-uniform and uniform ($\epsilon_t = \epsilon$ for all time steps) quantization intervals. Finally, in d), we present a stream of data points, where $\epsilon_t = 0$, for all time steps.

about the signal acquisition at a given time stamp. It is also the primary data type considered in this Ph.D. work, which we sometimes refer to as a data sample.

For a formal definition, consider a stream of uncertainty intervals; then, the tth observed uncertainty interval is expressed as $\boldsymbol{o}_t = [x_t, y_t, \epsilon_t]^\top$, where $x_t \in \mathbb{R}$ denotes its time stamp, $y_t \in \mathbb{R}$ is the center of the interval, and $\epsilon_t \in \mathbb{R}_+$ stands for the interval half step. This definition encompasses many data types encountered in practice, such as safety regions, quantization intervals, and data points (data samples with no uncertainty in the image domain), as illustrated in Fig. 2.1.

Even though the methods proposed in Paper A, Paper B, and Paper D are designed to work with any uncertainty interval, in this Ph.D. thesis, we restrict to (not necessarily uniform) quantization intervals as a common use case.

2.1.3 Zero-delay Response

In machine learning, online learning is a well-established paradigm devised for sequential data [30, 31]. Online learning methods are typically used to alleviate the computational complexity of "offline" algorithms, i.e., those techniques designed to work over the entire data set at once, usually over huge data sets in practice. Similarly, they are also used in tasks where adapting to new patterns in the data or satisfying real-time constraints is required.

In the context of online learning for signal reconstruction, delay can be understood as i) the difference between the time a sample is received and the time its corresponding signal reconstruction is delivered and ii) the number of samples that are necessary to process before a signal reconstruction is proposed.
This thesis adopts the concept of delay in terms of the number of samples. Therefore, an algorithm sequentially reconstructing a signal without waiting for the next sample is deemed as operating under a *zero-delay* response.

2.1.4 Smoothness

Given a univariate function, its *smoothness* refers to the number of continuous derivatives it has over the interior of its domain. At the very minimum, a function could be considered smooth if it is differentiable (and hence continuous) everywhere. At the other end, it might also possess derivatives of all orders in its domain, in which case it is said to be infinitely differentiable.

In addition, the term smoothness is usually used as the opposite of the concept of *roughness*, a derivative-based metric that quantifies the model complexity of certain smooth functions [32]. Mathematically, given a space of functions $\mathcal{F}_\rho$ over the domain $\mathrm{dom}(f)$ with $\rho - 1$ absolutely continuous derivatives and with a squared integrable ρ-th derivative, their roughness $r_\rho : \mathcal{F}_\rho \to \mathbb{R}$ is computed as

$$r_\rho(f) = \int_{\mathrm{dom}(f)} \left(D_x^\rho f(x)\right)^2 \, dx. \tag{2.2}$$

In summary, the word "smoothness" may refer to the fact that the roughness of a function is minimized or to the fact that the function has a certain number of continuous derivatives. This Ph.D. thesis deals with both these distinct but related concepts since, to minimize the ρ-roughness, we need the function to be at least $(\rho - 1)$-smooth. Whenever the meaning is unclear from the context, the latter will be referred to as the degree of smoothness.

2.1.5 Spline models

By convention, a spline model is defined as a piecewise polynomial function. In this thesis work, we denote any one-dimensional spline composed of T function pieces as

$$f_T(x) = \begin{cases} g_1(x), & \text{if } x_0 < x \leq x_1, \\ g_2(x), & \text{if } x_1 < x \leq x_2, \\ \vdots & \\ g_T(x), & \text{if } x_{T-1} < x \leq x_T, \end{cases} \tag{2.3}$$

where every tth function piece, or function section, $g_t : (x_{t-1}, x_t] \to \mathbb{R}$ is a polynomial of the form

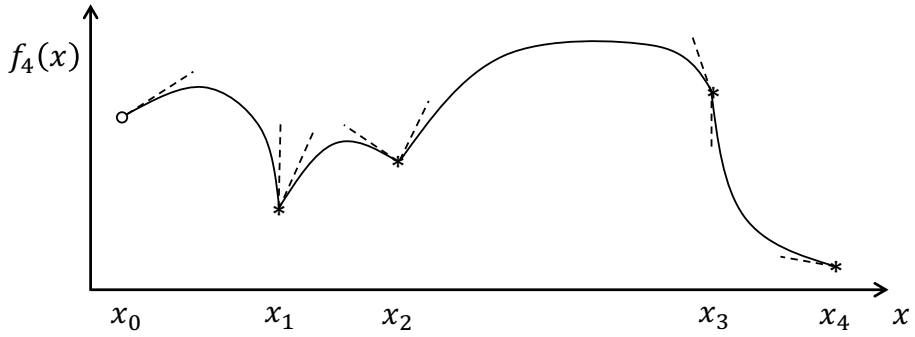
$$g_t(x) = \boldsymbol{a}_t^\top \boldsymbol{p}_t(x), \tag{2.4}$$

with combination coefficients $\boldsymbol{a}_t \in \mathbb{R}^{d+1}$ and basis vector function $\boldsymbol{p}_t : (x_{t-1}, x_t] \to \mathbb{R}^{d+1}$ given by

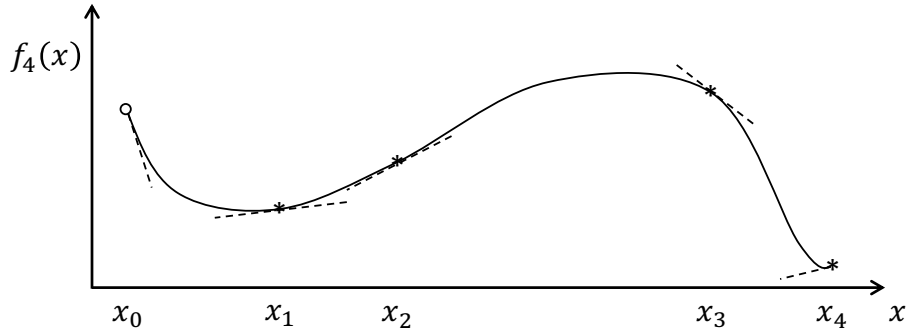
$$\boldsymbol{p}_t(x) = [1, (x - x_{t-1}), \ldots, (x - x_{t-1})^d]^\top, \tag{2.5}$$

where the positive integer d denotes the order of the spline.
On the other hand, a spline model is said to have a degree of smoothness $\varphi \in \mathbb{N}$ if it has φ continuous derivatives over the interior of its domain $\mathrm{dom}(f_T) = \bigcap_{t=1}^{T}(x_{t-1}, x_t]$. Some spline configurations with different order and degree of smoothness are illustrated in Fig. 2.2.

It is worth mentioning that there are many alternative ways of equivalently representing the spline model in (2.3); for instance, by using overlapping basis functions implicitly accommodating the smoothing constraints, such as in B-splines or cardinal splines models [33]. Although these alternative representations are equivalent,



(a) Spline of order d with discontinuous first derivative.



(b) Spline of order d with at least the zeroth and first derivatives continuous.

Figure 2.2: Example of interpolation splines. The asterisks represent data points; in this case, they also coincide with the spline knots (joint points).

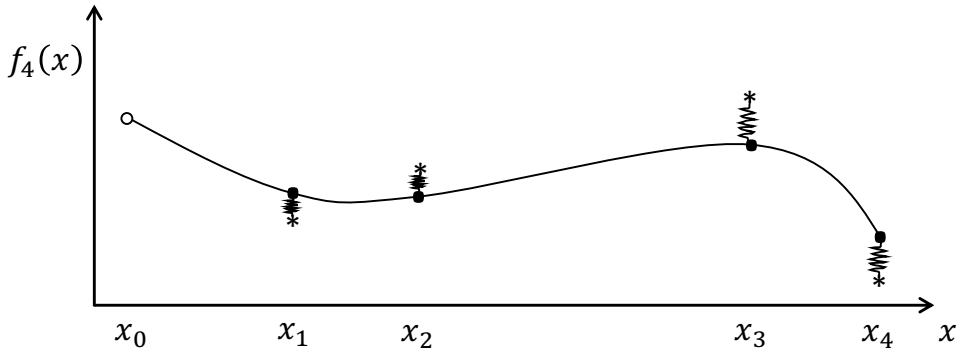


Figure 2.3: Physical analogy of a smoothing spline [6]. The asterisks represent data points, while the black points represent the spline knots. Here, the smoothing spline can be viewed as a strip hooked up to the data points by ideal springs (for small elongations).

it is preferable to use the piecewise representation for some tasks since we can directly impose continuity constraints into the model representation. Otherwise, the continuity constraints are embedded into the basis expansion, hence making such a task cumbersome, as further discussed in Paper C.

2.1.5.1 Smoothing spline interpolation

Let us analyze the naming conventions that constitute the notion of "smoothing spline interpolation" by parts. The term "smoothing" refers to a controlled trade-off between fitting data points and proposing a smooth (low complexity) signal estimate. By "interpolation," one usually refers to estimating new data within the time domain delimited by the range of known data points rather than simply connecting them.
On the other hand, "interpolation splines" and "smoothing splines" may seem completely different constructs, especially from an implementation perspective, since the former passes through the known data points and the latter does not necessarily. However, by construction, "interpolation splines" can be seen as a particular limit case of "smoothing splines."
This is because the smoothing spline is defined as

$$f_T(x) = \arg\min_{f \in \mathcal{F}_\rho} \sum_{t=1}^{T} (f(x_t) - y_t)^2 + \frac{\eta}{2} r_\rho(f), \tag{2.6}$$

where the class of functions $\mathcal{F}_\rho$ and roughness metric r_ρ, are the same as introduced in Sec. 2.1.4. Thus, notice that as the parameter $\eta \to 0$, the smoothing spline converges to the interpolating spline. Some intuition behind the smoothing spline, defined in (2.6), is provided in Fig. 2.3.

2.2 Proximal Algorithms

Suppose we are given a differentiable convex function $f : \mathbb{R}^D \to \mathbb{R}$ whose minimum value(s) cannot be found in closed form or that finding them is computationally prohibitive. In this case, one can resort to iterative methods [34].
Roughly, an iterative method is an algorithm that generates a sequence of iterates, e.g., $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$. These algorithms are typically devised so that the sequence of iterates converges to an optimal solution, denoted as $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{R}^D} f(\boldsymbol{x})$, as the number of iterations, or steps, increases.

Arguably, the most emblematic iterative algorithm is the *gradient descent* algorithm [35], whose core program is exposed next in **Algorithm 1**.

Algorithm 1 Gradient Descent

Input: Initial $\boldsymbol{x}_1$, a convergence criterion and a step-size sequence $\{\eta_j\}_j$ or rule.
1: **for** $i = 1, 2, \dots$ **do**
2: $\quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \eta_i \nabla f(\boldsymbol{x}_i)$.
3: $\quad$ Break if the convergence criterion is satisfied.
4: **end for**
Output: The last iterate.

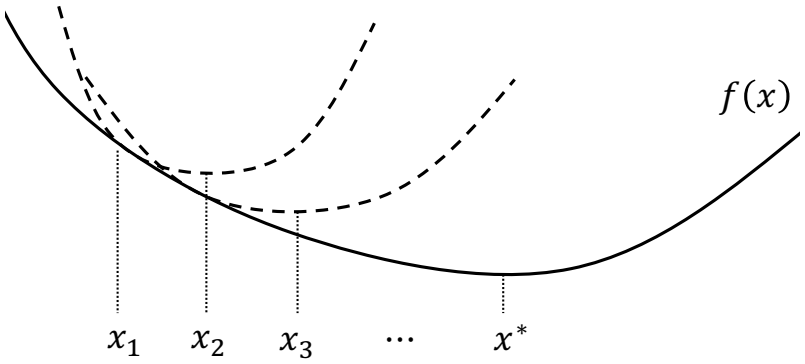


Figure 2.4: Gradient descent as the minimization of successive quadratic local approximations for $D = 1$.

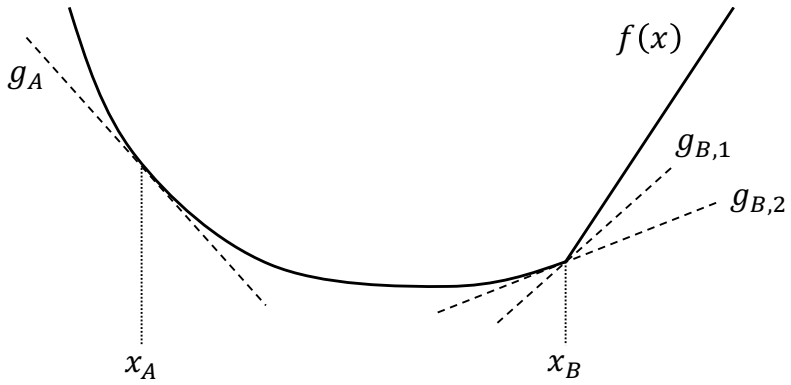


Figure 2.5: Example of subgradients for $D = 1$. At x_A the function f is differentiable, and g_A is the unique derivative. On the other hand, at the point x_B the function f is non-differentiable and has many subgradients (for convenience, only two subgradients $g_{B,1}$ and $g_{B,2}$ are shown).

The descending step in **Algorithm 1** (line 2) admits several interpretations. One of them, which allows for a more in-depth comparison analysis later on, is the following: From the second-order Taylor approximation around $\boldsymbol{x}_i$, i.e.,

$$f(\boldsymbol{x}) \simeq \tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^\top (\boldsymbol{x} - \boldsymbol{x}_i) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^\top \mathbf{H}(f(\boldsymbol{x}_i))(\boldsymbol{x} - \boldsymbol{x}_i), \quad (2.7)$$

we can replace the Hessian $\mathbf{H}(f(\boldsymbol{x}_i))$ for a more computationally friendly value $\frac{1}{\eta_i}\mathbf{I}_D$, where $\mathbf{I}_D$ is the identity matrix of size D and η_i is a positive real parameter closely related to the curvature of a quadratic approximation, obtaining

$$f(\boldsymbol{x}) \simeq \tilde{f}_{\eta_i}(\boldsymbol{x}) = f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^\top (\boldsymbol{x} - \boldsymbol{x}_i) + \frac{1}{2\eta_i}\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2. \quad (2.8)$$

Then the next iterate is given by $\boldsymbol{x}_{i+1} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^D} \tilde{f}_{\eta_i}(\boldsymbol{x})$, which can be found as the stationary point satisfying that

$$\nabla \tilde{f}_{\eta_i}(\boldsymbol{x}_{i+1}) = 0, \quad (2.9a)$$

$$\text{or, equivalently, } \nabla f(\boldsymbol{x}_i) + \frac{1}{\eta_i}(\boldsymbol{x} - \boldsymbol{x}_i) = 0. \quad (2.9b)$$

The corresponding visual representation is provided in Fig. 2.4.

In many applications, the objective function f may be non-differentiable. In such cases, it is not possible to compute the gradient everywhere; thus the gradient descent method cannot be used. On the contrary, the *subgradient* method is a simple algorithm for minimizing non-differentiable convex functions [36].
The main difference with respect to gradient descent lies in the "descending" step in **Algorithm 1** (line 2). Instead, the subgradient method iterates as

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \eta_i \, \boldsymbol{g}_i, \quad (2.10)$$

where $\boldsymbol{g}_i$ is any subgradient of f at $\boldsymbol{x}_i$, that is, any vector in $\mathbb{R}^D$ such that

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_i) + \boldsymbol{g}_i^\top (\boldsymbol{x} - \boldsymbol{x}_i)\,, \quad (2.11)$$

is satisfied, as illustrated in Fig. 2.5.
However, the subgradient method is not a descent method; that is, the function value at the next iterate can increase, hence slowing the convergence compared to gradient descent methods.

Differently, *proximal* algorithms can deal with a non-smooth function f while matching the gradient descent algorithm convergence rate [37]. They make use of the proximal operator, in this case, a mapping $\text{prox}_f^\eta : \boldsymbol{v} \in \mathbb{R}^D \mapsto \text{prox}_f^\eta(\boldsymbol{v}) \in \mathbb{R}^D$ that comprises between minimizing f and being near to a given point $\boldsymbol{v}$, where η can be seen as a trade-off parameter between these terms.
In proximal algorithms, the equivalent to the descending step in **Algorithm 1** (line 2) at each iteration consists of evaluating the proximity (prox) operator of a function, which involves solving a convex optimization subproblem. This subproblem can be solved with standard optimization methods, but it often admits closed-form representations or can be solved very quickly with specialized methods [37].

As an example, consider a convex non-differentiable function f that can be decomposed as $f(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x})$, where g is convex and differentiable and h is also

convex but not necessarily differentiable.
Similarly to the gradient descent method, we can perform a Taylor expansion of only the differentiable term in the objective, i.e., we expand g around the iterate $\boldsymbol{x}_i$ and use the stationary point of the resulting approximation as the next iterate. Mathematically,

$$\boldsymbol{x}_{i+1} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^D} \left\{ g(\boldsymbol{x}_i + \nabla g(\boldsymbol{x}_i)^\top (\boldsymbol{x} - \boldsymbol{x}_i) + \frac{1}{2\eta}\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 + h(\boldsymbol{x}) \right\} \tag{2.12a}$$

$$= \arg\min_{\boldsymbol{x}\in\mathbb{R}^D} \left\{ \frac{1}{2\eta} \|\boldsymbol{x} - (\boldsymbol{x}_i - \eta\nabla g(\boldsymbol{x}_i))\|_2^2 + h(\boldsymbol{x}) \right\} \tag{2.12b}$$

$$\triangleq \operatorname{prox}_h^\eta \left(\boldsymbol{x}_i - \eta\nabla g(\boldsymbol{x}_i)\right), \tag{2.12c}$$

where the algebraic step (2.12c) applies the definition of the prox operator.
This algorithm is known in the literature as the proximal gradient descent method or the proximal forward-backward iterative scheme [38].

2.3 Learning with Kernels

Empirical risk minimization (ERM) is arguably one of the most popular supervised approaches to designing learning algorithms [39]. The general idea consists of minimizing the empirical loss

$$\frac{1}{T}\sum_{t=1}^{T} \ell(f(x_t), y_t), \tag{2.13}$$

as a proxy for the expected loss

$$\mathop{\mathbb{E}}_{x,y\sim p(x,y)} [\ell(f(x), y)], \tag{2.14}$$

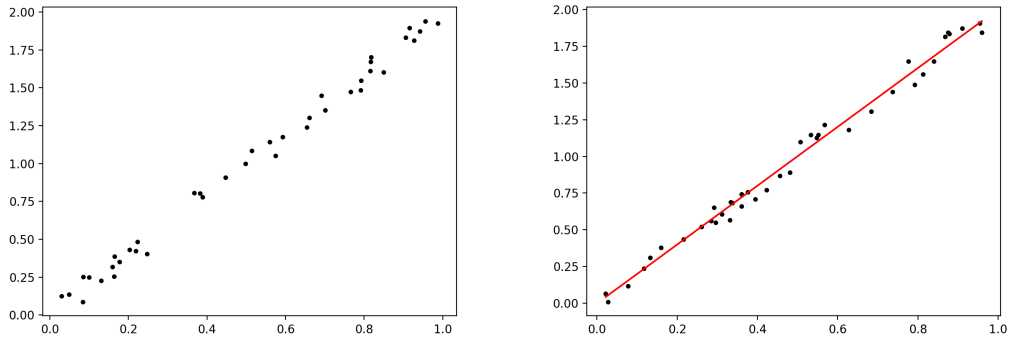
since, in most cases, the expectation cannot be obtained due to a prohibitive computational cost or because the joint probability $p(x, y)$ is unknown.

To put the above algorithmic idea into practice, one needs to fix a suitable function space and then minimize the empirical loss over it. Additionally, the function space should allow feasible computations and be rich (encompassing a large number of candidate functions) when the model complexity is unknown a priori. Here, the space of linear functions is conceivably the simplest example.

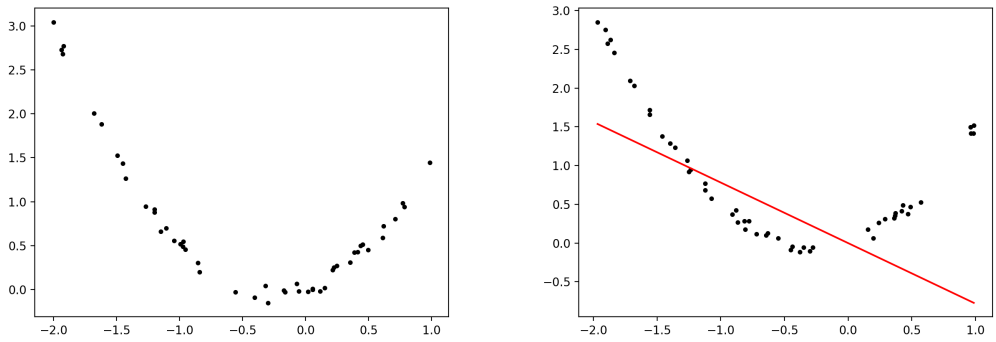
On the other hand, if the function space is too rich, solving an ERM problem can cause overfitting, i.e., poor generalization to unseen data. Fortunately, regularization techniques can enforce stability (by reducing the model complexity) and promote generalization [40]. One possible way to do this is by adding a regularization term (penalizing a metric of the complexity of the learned function) to the empirical loss.

2.3.1 Beyond Linear Models

Suppose we have a set of data points $\mathcal{S} = \{(x_t, y_t)\}_{t=1}^T$, where $x_t, y_t \in \mathbb{R}$, to which we want to fit a linear model $f : x \in \mathbb{R} \mapsto wx \in \mathbb{R}$, with $w \in \mathbb{R}$. Then, if certain conditions, such as a linear relationship between the independent and dependent variables in the data set, are not met, the linear model provides inaccurate solution estimates, as shown in Fig. 2.6.



(a) Accurate fit.



(b) Underfit. The space of linear functions is not rich enough to properly capture the data relationships.

Figure 2.6: Linear model fit via ridge regression for two different data sets.

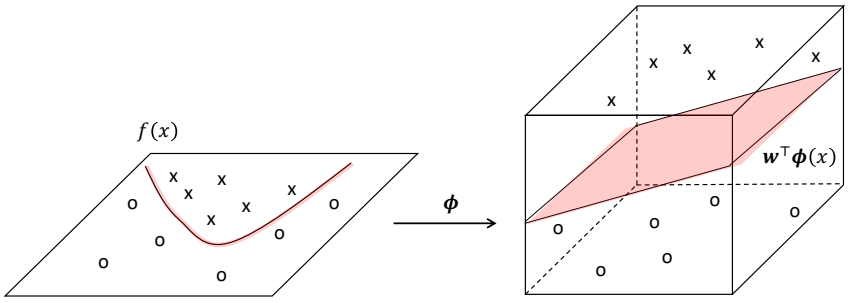


Figure 2.7: Geometrical interpretation of the feature map. On the left is the input space, where the model f is seen as a non-linear function. On the right is the feature space, where the model is seen as linear with respect to the coefficients in $\boldsymbol{w}$.

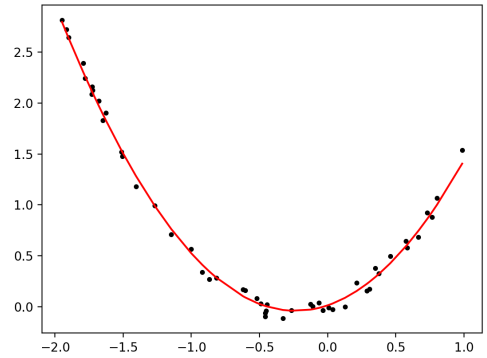


Figure 2.8: Fit of a quadratic function as a linear combination of features via ridge regression.

At this point, one may wonder whether it is possible to learn a linear model from a richer function space to overcome underfitting. The answer is yes.

In general, we can use any feature map $\boldsymbol{\phi}(\boldsymbol{x}) = [\varphi_1(\boldsymbol{x}), \ldots, \varphi_F(\boldsymbol{x})]^\top$, where $\varphi_i : \mathbb{R}^D \to \mathbb{R}$ and $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^F$ (usually for $F >> D$). This leads to non-linear models in the input space but linear in the feature space of the form

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{F} w_i \varphi_i(\boldsymbol{x}), \tag{2.15}$$

where $\boldsymbol{w} \in \mathbb{R}^F$. This is illustrated in Fig. 2.7.
As an example, the regression problem posed before in Fig. 2.6b corresponds to $D = 1$ and can be solved with a feature map in $F = 3$ of the form $\boldsymbol{\phi}(x) = [1, x, x^2]^\top$, as shown in Fig. 2.8.

On the other hand, thanks to the Representer Theorem [41], we know that for any given loss $\ell : \mathbb{R}^2 \to \mathbb{R}$ the solution to the optimization problem

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w} \in \mathbb{R}^F} \frac{1}{T} \sum_{t=1}^{T} \ell(\boldsymbol{\phi}(\boldsymbol{x}_t)^\top \boldsymbol{w}, y_t) + \frac{\eta}{2} \|\boldsymbol{w}\|_2^2, \tag{2.16}$$

can be written as $\boldsymbol{w}^* = \boldsymbol{\alpha}^\top \boldsymbol{\Phi}$, where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_T)]^\top \in \mathbb{R}^{T \times F}$ is the data matrix in the feature space, and $\boldsymbol{\alpha} \in \mathbb{R}^T$ is some real-valued vector of coefficients. Then, the solution model is of the form

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{w}^* \tag{2.17a}$$

$$= \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\Phi}^\top \boldsymbol{\alpha} \tag{2.17b}$$

$$= \sum_{t=1}^{T} \alpha_t \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}_t), \tag{2.17c}$$

that is, $f(\boldsymbol{x})$ is expressed by using inner products between feature vectors.
In fact, we do not even need to explicitly define a feature map $\boldsymbol{\phi}$, but only a reproducing kernel function $k(\boldsymbol{x}, \boldsymbol{x}_t) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}_t)$. This procedure is commonly referred to in the literature as the *kernel trick* [42].

Finally, every kernel is associated with a unique reproducing kernel Hilbert space (RKHS), and conversely. The RKHS theory is outstanding; the underlying definitions are conceptually simple yet powerful and broadly applicable in practice [43].

2.4 Finite Horizon Stochastic Dynamic Programming

There exist several classes of techniques for optimal decision-making in dynamic state spaces. Classical control theories, discrete-time dynamic programming (DP), and machine learning-based approaches, often referred to as reinforcement learning [44], aim to determine the best possible action or sequence of actions in a given situation, considering the evolution of states of a dynamic environment over time. In this section, we will delve into deterministic and stochastic DP.

Deterministic dynamic programming problems involve a discrete-time dynamic system of the form

$$\boldsymbol{s}_{t+1} = F_t(\boldsymbol{s}_t, \boldsymbol{a}_t), \text{ for } t = 0, \ldots, T-1, \tag{2.18}$$

where t is the time index, $\boldsymbol{s}_t$ is the tth state, $\boldsymbol{a}_t$ is the tth action, F_t describes the state update mechanism at time step t, and T is the total number of steps or horizon.
The set of all possible states $\boldsymbol{s}_t$ is referred to as the state space at time t. It can be any set and can depend on t; this generality is one of the great strengths of the DP methodology [45]. Similarly, the set of all possible actions $\boldsymbol{a}_t$ is called the action space at time t. Again, it can be any set and can depend on t. The problem also involves an additive real-valued cost function in the sense that the cost incurred at time t, denoted by $g_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$, accumulates over time, and may depend on t. For a given initial state $\boldsymbol{s}_0$, the total cost of an action sequence $\{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{T-1}\}$ is

$$J(\boldsymbol{s}_0; \boldsymbol{a}_0, \ldots, \boldsymbol{a}_{T-1}) = g_T(\boldsymbol{s}_T) + \sum_{t=0}^{T-1} g_t(\boldsymbol{s}_t, \boldsymbol{a}_t), \tag{2.19}$$

where $g_T(\boldsymbol{s}_T)$ is a terminal cost incurred at the end of the process.
The objective is to minimize the total cost (2.19) over all sequences of feasible actions.

The finite-horizon stochastic DP problem differs from the deterministic version primarily in the nature of the discrete-time dynamic system that governs the evolution of the states. This system includes a random variable w_t, which accounts for the uncertainty and is characterized by a probability distribution $p_t(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$ that may depend explicitly on $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$, but is independent of prior values of $w_{t-1}, \ldots, w_0$.
In this case, the state update mechanism is of the form

$$\boldsymbol{s}_{t+1} = F_t(\boldsymbol{s}_t, \boldsymbol{a}_t, w_t), \text{ for } t = 0, \ldots, T-1, \tag{2.20}$$

and the cost per time step g_t also depends on the corresponding random uncertain variable w_t. Additionally, one optimizes the expected cost not over sequences of actions but rather over policies or decision strategies that consist of a sequence of functions $\pi = \{\mu_0, \ldots, \mu_{T-1}\}$ mapping states into actions.
Policies are more general objects than sequences of actions. In the presence of stochastic uncertainty, policies often incur a lower cost than any fixed sequence of actions since they allow choices of actions that incorporate knowledge of the current state. Without this knowledge, the agent may not act appropriately, and the cost can be adversely affected. In other words, policies encode the necessary knowledge

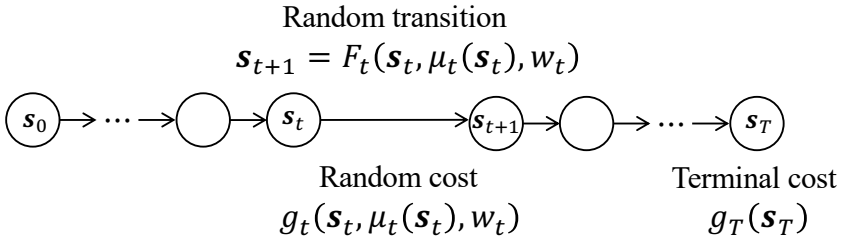


Figure 2.9: Illustration of a T-steps finite horizon stochastic dynamic programming problem.

to adapt to any state of the environment and minimize the expected cost.
The general idea is visually summarized in Fig. 2.9.

Thus, for given cost functions g_t for $t = 0, \ldots, T$, the expected total incurred cost by following a given policy π starting at $\boldsymbol{s}_0$ is

$$J_\pi(\boldsymbol{s}_0) = \mathbb{E}\left[g_T(\boldsymbol{s}_T) + \sum_{t=0}^{T-1} g_t(\boldsymbol{s}_t, \mu_t(\boldsymbol{s}_t), w_t)\right], \tag{2.21}$$

where the expectation is taken over all random variables $\boldsymbol{s}_t$ and w_t.
An optimal policy, denoted as π^*, is the one that minimizes (2.21), that is, $\pi^* = \arg\min_{\pi\in\Pi} J_\pi(\boldsymbol{s}_0)$, where Π is the set of all policies.

On the other hand, Bellman's principle of optimality says that the tail of an optimal sequence is optimal for the tail subproblem [46]. That is, the tail $\{\boldsymbol{a}_t^*, \cdots, \boldsymbol{a}_{T-1}^*\}$ of an optimal sequence of actions $\{\boldsymbol{a}_0^*, \cdots, \boldsymbol{a}_{T-1}^*\}$ is optimal for the tail subproblem that starts at the state $\boldsymbol{s}_t^*$ of the optimal trajectory of states $\{\boldsymbol{s}_1^*, \cdots, \boldsymbol{s}_T^*\}$, as shown in Fig. 2.10.

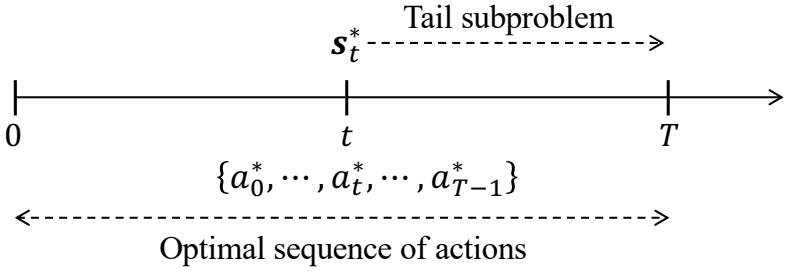


Figure 2.10: Illustration of the principle of optimality.

Based on the principle of optimality, we can compute the tth optimal action $\boldsymbol{a}_t^* = \mu_t^*(\boldsymbol{s}_t)$ by a backward induction process, as

$$\mu_t^*(\boldsymbol{s}_t) \in \arg\min_{\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s}_t)} \mathbb{E}\left[g_t(\boldsymbol{s}_t, \boldsymbol{a}, w_t) + J_{t+1}^*(F_t(\boldsymbol{s}_t, \boldsymbol{a}, w_t))\right], \tag{2.22}$$

where $\mathcal{A}(\boldsymbol{s}_t)$ denotes the set of feasible actions from the state $\boldsymbol{s}_t$, and J_{t+1}^* is the optimal cost-to-go from the $(t+1)$th step.
However, computing the optimal cost-to-go can be computationally expensive. This motivates (suboptimal) approximation techniques, where, for example, one replaces J_t^* with an approximate cost-to-go function, denoted as $\tilde{J}_t$, that is less complex to obtain.

2.4.1 Policy Parametrization Through Cost Parametrization

Suppose we start a backward induction procedure with a parametric cost-to-go function approximation $\tilde{J}_t(\boldsymbol{s}_t;\boldsymbol{\theta})$, for every step t, where $\boldsymbol{\theta}$ is a vector of P trainable parameters. Then, we can obtain an approximation in policy space of parametric policies $\tilde{\pi} = \{\tilde{\mu}_0, \ldots, \tilde{\mu}_{T-1}\}$ defined through the following optimization problem

$$\tilde{\mu}_t(\boldsymbol{s}_t;\boldsymbol{\theta}) \in \arg\min_{\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s}_t)} \mathbb{E}\left[g_t(\boldsymbol{s}_t,\boldsymbol{a},w_t) + \tilde{J}_{t+1}\left(F_t(\boldsymbol{s}_t,\boldsymbol{a},w_t);\boldsymbol{\theta}\right)\right]. \tag{2.23}$$

An important issue here is the computation of the expected value and the minimization over $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s}_t)$. Both of these operations may involve substantial computational load, which is of particular concern when the minimization is to be performed online.

One possibility to eliminate the expected value from the expression (2.23) is to choose a typical value (depending on the application) of uncertain variable $\tilde{w}_t$, and use the action that solves the resulting deterministic problem. On top of this, both the cost functions g_t and cost-to-go function approximations $\tilde{J}_t$ can be chosen so the optimization problem (2.23) admits a closed-form solution. Moreover, g_t and $\tilde{J}_t$ can also be chosen to be stationary, i.e., $g \equiv g_t = g_{t+1}$ and $\tilde{J} \equiv \tilde{J}_t = \tilde{J}_{t+1}$, for all time steps, which is a more suitable choice for tasks of variable horizon (undetermined number of time steps), such as those involving streaming data.

2.5 Recurrent Neural Networks

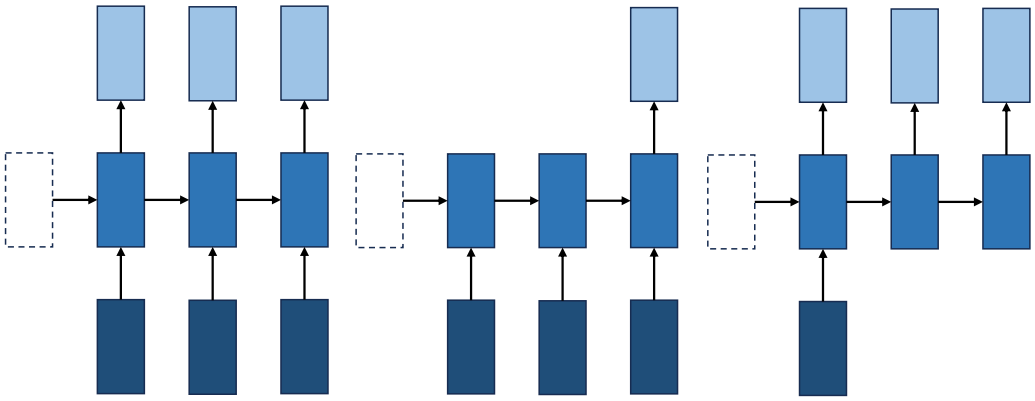
Recurrent neural networks (RNNs) are a family of neural networks designed to process sequential data [47]. RNNs find applications in domains ranging from time series forecasting and natural language processing to multimedia analysis and control systems. They specialize in handling data sequences with a length that would be impractical for other types of neural networks. Furthermore, most RNNs have the ability to process sequences of varying length.

The key idea behind RNNs is sharing parameters across different parts of the model. If an RNN had separate parameters for each sequence data term, it would neither be able to generalize to sequences of arbitrary length nor to share learned behaviors across different terms. Such sharing is particularly important when a specific piece of information can occur at multiple positions within the data sequence. There is a wide variety of RNNs; some representative examples are shown in Fig. 2.11.

Computing the gradient through an RNN is straightforward. One can apply the backpropagation algorithm to the unrolled computational graph of the RNN. This is known as backpropagation through time (BPTT) [48]. Gradients obtained by BPTT may then be used together with any gradient-based optimizer to train an RNN.

The mathematical challenge of learning long-term dependencies in RNNs is that gradients propagated over many steps tend to either vanish or explode [49]. In the scalar case, this implies multiplying a weight w by itself many times. The product $w \cdot w \cdots w$ will either vanish or explode depending on the magnitude of w.

On the other hand, even if we assume that the parameter values are such that the RNN is stable, the difficulty with long-term dependencies arises from the exponentially smaller weights given to long-term interactions (involving the multiplication of



(a) Many-to-many RNN (b) Many-to-one RNN (c) One-to-many RNN

Figure 2.11: Common RNN architectures. From darker blue (bottom) to lighter (top), we have the input, the hidden layer, and the output. The dashed empty cell represents an initial hidden state. Some practical applications for the architectures in Fig. 2.11a, Fig. 2.11b, and Fig. 2.11c are time-series forecasting [7], sentiment analysis [8], and image captioning [9], respectively.

many Jacobians) compared to short-term ones. Whenever the model can represent long-term dependencies, the gradient related to a long-term interaction has an exponentially smaller magnitude than the gradient related to a short-term interaction. This means not that it is impossible to learn but that it might take a very long time to learn long-term dependencies because these dependencies will tend to be hidden by the fluctuations arising from short-term dependencies.
Nowadays, one of the most effective models for data sequences used in practical applications is the gated RNNs. They are based on creating paths through time with derivatives that seldom vanish nor explode. This is done through a mechanism that adds or subtracts (instead of multiplying) relevant or irrelevant information to the shared state. Noticeable examples of gated RNNs are the long short-term memory (LSTM) [50] architecture and the gated recurrent unit (GRU) [51].

2.6 Consistent Signal Estimates

Given a bandlimited signal ψ, it is well known that we can ensure a reversible discretization in time by sampling at, or above, its Nyquist sampling rate [52, 53]. In practice, one cannot avoid an additional discretization in amplitude due to the physical limitations of acquisition systems, causing an irreversible loss of information and, therefore, making a perfect signal reconstruction no longer possible. These physical limitations of acquisition systems are typically modeled through quantization mappings.

Even though quantization is a deterministic operation, the signal reconstruction error can be analyzed effectively from a stochastic point of view when certain assumptions are met [54, 55]. Under these assumptions, low-pass filtering reduces the mean squared error (MSE) of a reconstructed signal f by a factor proportional to the squared quantization step size Δ^2 and inversely proportional to the oversampling

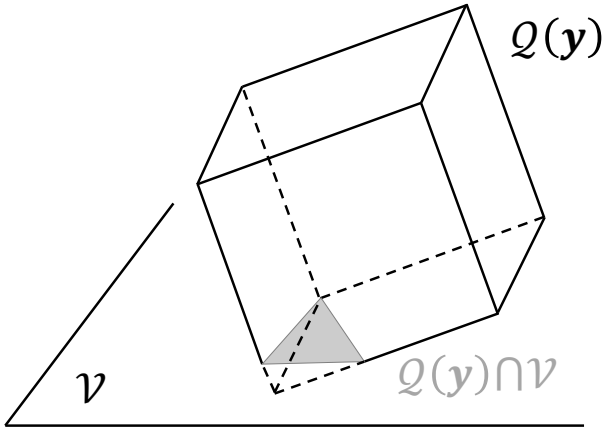


Figure 2.12: Geometric representation of the set of consistent signal estimates $\mathcal{Q}(\boldsymbol{y})\cap\mathcal{V}$.

ratio R. That is,

$$\mathbb{E}\left[\|f-\psi\|_2^2\right] \propto \frac{\Delta^2}{R}, \tag{2.24}$$

where the expectation is taken over the amplitude of the reconstruction error, modeled as a uniform random variable [54].

On the other hand, deterministic analyses, based on deterministic bounds over the quantization levels, show that it is possible to reconstruct quantized signals with a squared norm error that is asymptotically proportional to the square of the quantization step size and inversely proportional to the square of the oversampling ratio, that is

$$\|f-\psi\|_2^2 = \mathcal{O}\left(\frac{\Delta^2}{R^2}\right), \tag{2.25}$$

via *consistent signal estimates* [27, 56].
In this case, a consistent signal estimate f is a bandlimited function that passes through all quantization intervals obtained after acquiring ψ.
More rigorously, the acquired signal $\boldsymbol{\psi} = [\psi(x_1),\ldots,\psi(x_T)]^\top \in \mathbb{R}^T$ is known only through its quantized version $\boldsymbol{y} \triangleq q(\boldsymbol{\psi}) = [q(\psi(x_1)),\ldots,q(\psi(x_T))]^\top$. Thus, the full information available about the acquisition of ψ is that $\boldsymbol{\psi} \in \mathcal{Q}(\boldsymbol{y})$, where $\mathcal{Q}(\boldsymbol{y})$ is the set of possible input signals with quantized version $\boldsymbol{y}$. As an example, when q is a uniform mid-tread quantizer with quantization step Δ, i.e., $q(\psi(x_t)) = \text{round}(\psi(x_t)/\Delta)\cdot\Delta$, $\mathcal{Q}(\boldsymbol{y})$ is a hypercube in $\mathbb{R}^T$ of side Δ, see Fig. 2.12. On the other hand, we additionally know that ψ belongs to the space of bandlimited signals $\mathcal{V}$. Hence, the elements of $\mathcal{Q}(\boldsymbol{y})\cap\mathcal{V}$ (illustrated in Fig. 2.12) are referred to as consistent signal estimates of ψ.

The behavior presented in (2.25) indicates that when a uniformly quantized and oversampled signal is reconstructed using consistent estimates, the reconstruction error can be reduced at the same asymptotic rate by either increasing the oversampling ratio R or decreasing the quantization step size Δ. This balanced signal reconstruction error-rate decay has important practical implications. For instance, in practice, it is more convenient to increase the oversampling ratio because the implementation cost (including monetary cost) of the analog circuitry necessary for

halving the quantization step size is much higher than that for doubling the oversampling ratio [57].

In more general settings, one can consider non-uniform quantization (the hypercube in Fig. 2.12 becomes a rectangular hyper-parallelepiped), and a different signal space than $\mathcal{V}$; for example, the space of non-bandlimited signals with finite rate of innovation [58], such as signals modeled by a train of delta functions or splines. This can be done so that the trend behavior in (2.25) is met [59, 28].

Chapter 3

Kernel-based Signal Tracking from Univariate Time Series of Quantization Intervals

This chapter summarizes Paper A and Paper B and delves into the synergies between them for the task of tracking univariate time series of quantization intervals via kernel-based online regression. The background theory relevant to this chapter can be found in Sec. 2.1, 2.2, and 2.3.

3.1 Motivation

Regression techniques based on reproducing kernels have shown remarkable success under stochastic optimization frameworks [60, 22]. However, most kernel-based methods in the literature are unsuited for tracking streamed quantized data via regression under dynamic environments. This shortcoming can primarily be attributed to the lack of flexibility in selecting the best-suited functional cost or to the mathematical structure of the regression problem not being adequately exploited.
In Paper A, we overcome these issues by introducing a novel method consisting of an online regression problem whose functional cost can accommodate quantized data and a stochastic proximal method that leverages its mathematical structure.

On the other hand, the choice of the reproducing kernel can significantly affect the performance of the proposed method in Paper A. Selecting an appropriate reproducing kernel can result in a computational challenge [61], especially when dealing with data-rich tasks without prior information about the solution domain.
We address this challenge in Paper B by devising a learning scheme combining several single kernel-based online methods, including the one proposed in Paper A. Our learning scheme uses a multi-kernel learning formulation [62], which can expand the solution space, thereby increasing the plausibility of finding higher-performance solutions while reducing the kernel-selection bias. Additionally, our learning scheme is parallelizable, allowing for the distribution of computational load across multiple computing units.

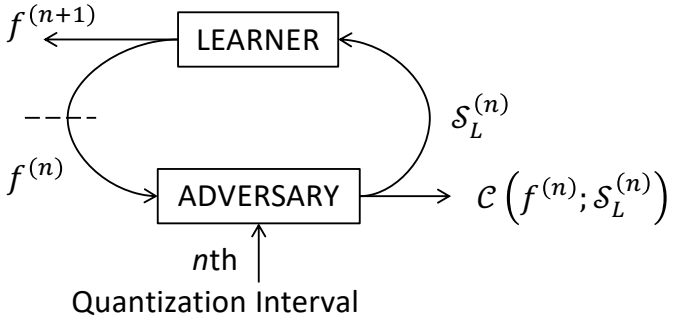


Figure 3.1: Visual representation of the online setting in Paper A and Paper B.

3.2 Problem Formulation

Online settings can be adopted to achieve low run-time complexity in exchange for a tolerable loss of accuracy [63, 64], e.g., by processing only a few data samples every iteration or discarding data samples after a few processing steps.
Paper A and Paper B share the same online setting core in their problem formulation, as summarized next.

Given a possibly infinite sequence $\mathcal{S}$ of quantization intervals, acquired at strictly increasing and non-necessarily equispaced time stamps, consider the following online setting, where the quantization intervals become available sequentially:
At each step n, the learner proposes a function estimate, which we denote as $f^{(n)}$. In response, an adversary penalizes the proposed function estimate with an incurred cost $\mathcal{C}(f^{(n)}; \mathcal{S}_L^{(n)})$, where $\mathcal{S}_L^{(n)} \subseteq \mathcal{S}$ denotes a sliding data window of (at most) length L. Then, the adversary reveals the nth data window $\mathcal{S}_L^{(n)}$ to the learner, which can be used at the next step $n+1$. This online setting is illustrated in Fig. 3.1.

It is worth mentioning that, in this case, both the learner and the adversary know the "shape" of the functional cost $\mathcal{C}$. Moreover, at step n, the learner uses the previously proposed function estimate $f^{(n-1)}$ to compute $f^{(n)}$, except at $n = 1$ for which no quantization interval has been received yet and thus, $f^{(1)}$ is chosen arbitrarily.

3.3 Proposed Solution

The functional cost considered in Paper A penalizes a given trade-off between a weighted sum of distances to the quantization intervals within a sliding data window and the model complexity of the function estimate.
Particularly, each of the distances to the quantization intervals are computed through the distance functional

$$d(f; H_i) = \inf_{h \in H_i} \|f - h\|_{\mathcal{H}}, \tag{3.1}$$

where $\mathcal{H}$ is the RKHS under consideration and

$$H_i \triangleq \{h \in \mathcal{H} : |h(x_i) - y_i| \leq \epsilon_i\}, \tag{3.2}$$

is the hyperslab associated with the ith quantization interval, i.e., the set of all functions $h \in \mathcal{H}$ that pass through the ith quantization interval.

Regarding the term accounting for the model complexity, we use the squared Hilbert norm, i.e., $\|f\|_{\mathcal{H}}^2$.

As a result, the functional cost in Paper A consists of a composite of a non-smooth and a smooth function term. Motivated by this mathematical structure, our proposed Windowed Online Regularized cost Minimization (WORM) algorithm uses the stochastic proximal average functional gradient descent method due to its favorable convergence performance [65, 37].

3.3.1 Proposed Learning Scheme

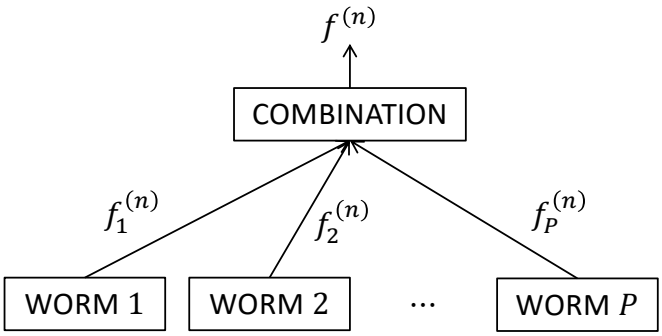


Figure 3.2: Visual representation, at the nth step, of the learning scheme proposed in Paper B for a set of P WORM methods.

In Paper B, we propose a learning scheme that can be seen as a higher-level learner that iteratively chooses the lowest incurring "learning" cost[1] convex combination of function estimates provided by lower-level learners, e.g., several P WORM algorithms, over different reproducing kernels or kernel hyperparameters, running in parallel. The complexity of the combination step we devise can scale down to linearly with P on average, i.e., $\mathcal{O}(P)$. Fig. 3.2 illustrates this learning scheme.

3.4 Performance Analyses

Under an online setting, as described in Sec. 3.2, the incurred cost accumulated over time steps receives the name of *cumulative cost*. As an illustration, suppose that the sequence $\mathcal{S}$ contains N terms; then, the cumulative cost is computed as $\sum_{n=1}^{N} \mathcal{C}(f^{(n)}, \mathcal{S}_L^{(n)})$.
Notice that the cumulative cost is a measure of performance against overfitting. This is because every nth function estimate $f^{(n)}$ must be proposed before the nth sliding data window $\mathcal{S}_L^{(n)}$ becomes available to the learner, see Fig. 3.1.
Popular performance metrics are constructed from the cumulative cost [66]. For instance, the *dynamic regret* metric is defined as

$$\mathbf{Reg}(f^{(1)}, \ldots, f^{(N)}) \triangleq \sum_{n=1}^{N} \mathcal{C}(f^{(n)}; \mathcal{S}_L^{(n)}) - \mathcal{C}(f_*^{(n)}; \mathcal{S}_L^{(n)}), \tag{3.3}$$

[1]Learning cost refers to a cost incurred while "learning" over a revealed sliding data window.

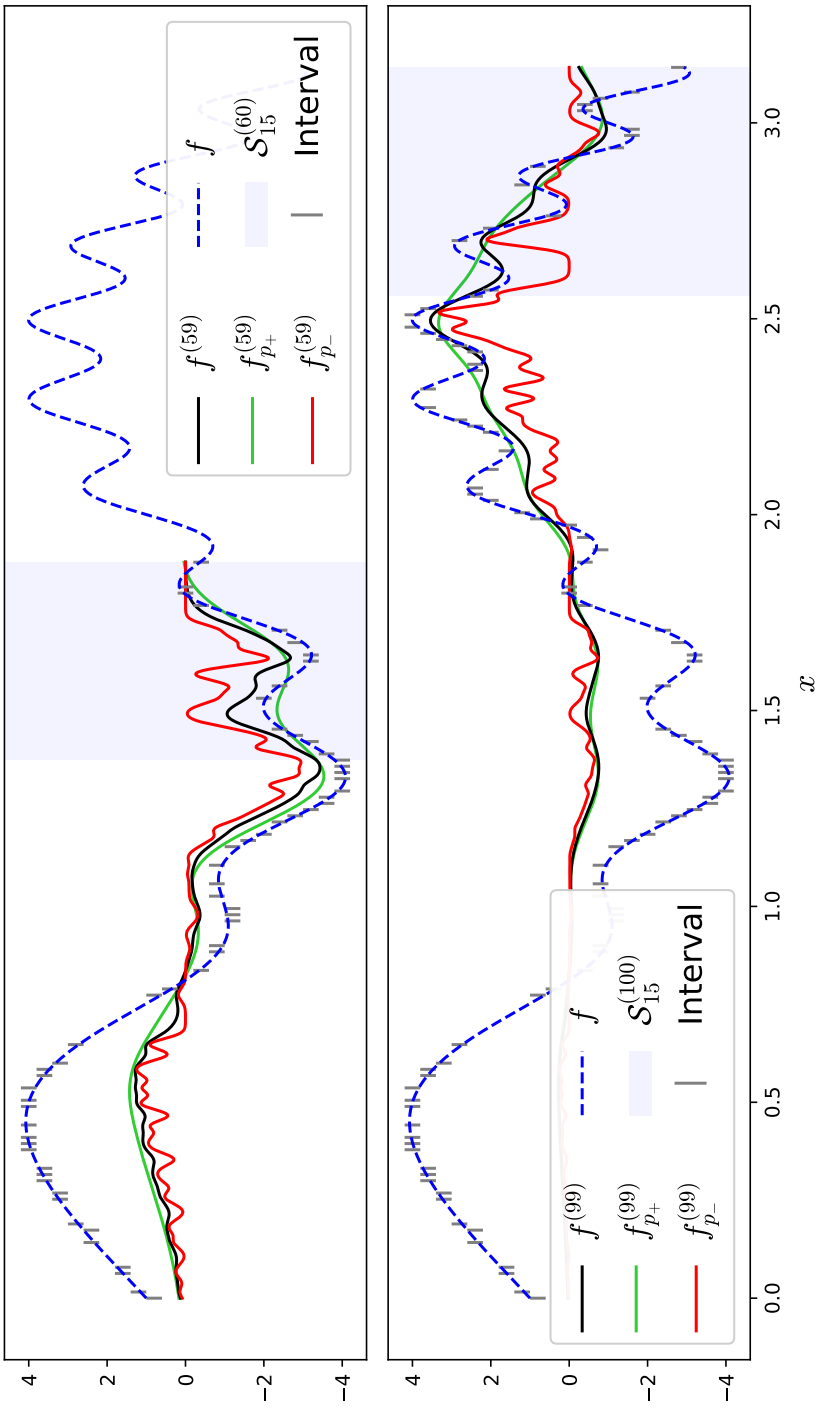


Figure 3.3: Signal tracking using learning scheme proposed in Paper B and $P = 10$ WORM algorithms with Gaussian kernels, i.e., $k(x,t) = \exp(-(x-t)^2/\sigma^2)$, with different widths σ spanning from 0.015 to 0.1, with a window of length $L = 15$, a regularization parameter of 0.2 and a learning rate of 2. The best and the worst individual WORM algorithms, in terms of the cumulative cost up to time step n (as shown in Fig. 3.4), are denoted by the indices p_+ and p_-, respectively. The data samples have been randomly sampled from $f(x) = \sin(2\pi x^2) + \cos(\pi x) + 3\sin(\pi x)$ and uniformly quantized with a half stepsize $\epsilon = 0.2$.

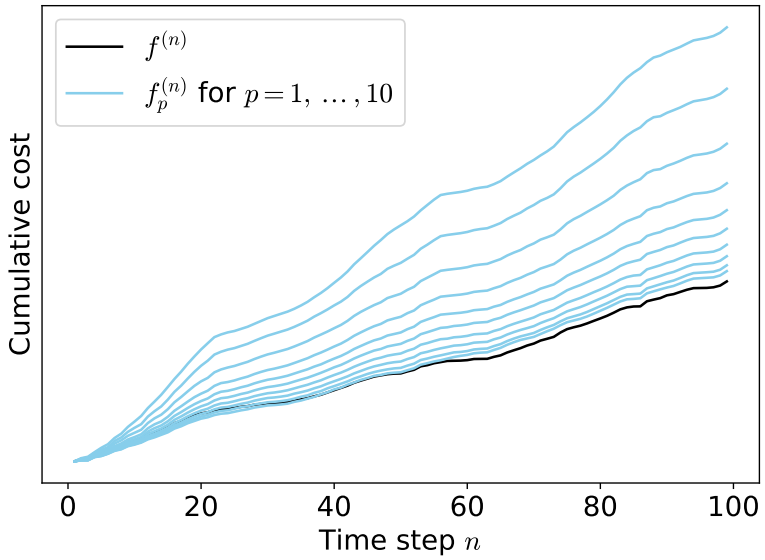


Figure 3.4: Cummulative cost up to time step n, i.e., $\sum_{i=1}^{n} \mathcal{C}(f^{(n)}; \mathcal{S}_L^{(n)})$, for the functional cost proposed in Paper A, and introduced in Sec. 3.3, for the $P = 10$ WORM algorithms and proposed learning scheme discussed in Fig. 3.3. As we can see, the proposed learning scheme in Paper B outperforms the best of the individual WORM algorithms used eventually.

where $f_*^{(n)} = \arg\min_f \mathcal{C}(f; \mathcal{S}_L^{(n)})$, and it indicates how well the sequence of function estimates $\{f^{(n)}\}_{n=1}^N$ matches the sequence of optimal decisions $\{f_*^{(n)}\}_{n=1}^N$.

In Paper A, we derive a theoretical upper bound for the dynamic regret incurred by the WORM algorithm.
On the other hand, Paper B shows experimentally how the proposed learning scheme outperforms the best combination of some single-kernel methods in terms of the regularized least squares cumulative cost.

For an example explicitly combining our proposed WORM algorithm and our proposed learning scheme, see Fig. 3.3 and Fig. 3.4.

3.5 Contributions

This section summarizes the main contributions of Paper A (♡) and Paper B (♣).

♡ We design a novel online algorithm called WORM to minimize a non-differentiable windowed cost assisted by the proximal average functional gradient descent method. We consider this algorithm for the problem of regression-based tracking of quantized signals.

♡ We provide tracking performance guarantees for WORM through a dynamic regret theoretical upper bound and experimental performance analyses using synthetic data.

♣ We present a multi-kernel learning scheme that combines online single kernel-based methods, outperforming the best of them individually in terms of the

cumulative regularized least squares cost metric, with a comparable computational load per computing unit. This corroborates the ability of the proposed scheme to effectively accommodate a larger function space (from which to draw function estimates) of multi-kernel methods while keeping the lower computational complexity of online single-kernel-based methods.

Chapter 4

Zero-delay Spline Interpolation Policies

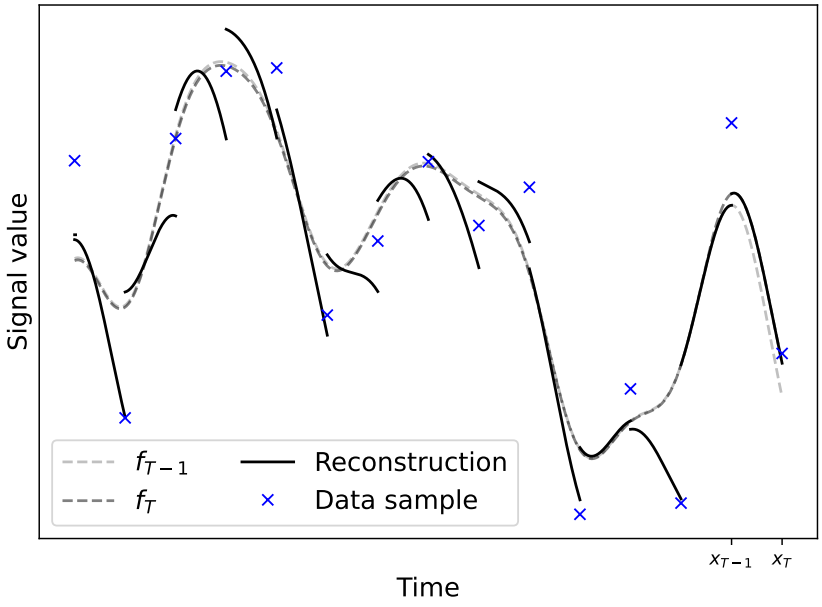
This chapter summarizes Paper C and Paper D and delves into the common aspects between them for the task of zero-delay smoothing spline interpolation from streaming time-series data. The background theory relevant to this chapter can be found in Sec. 2.1, 2.4, 2.5 and 2.6.

4.1 Motivation

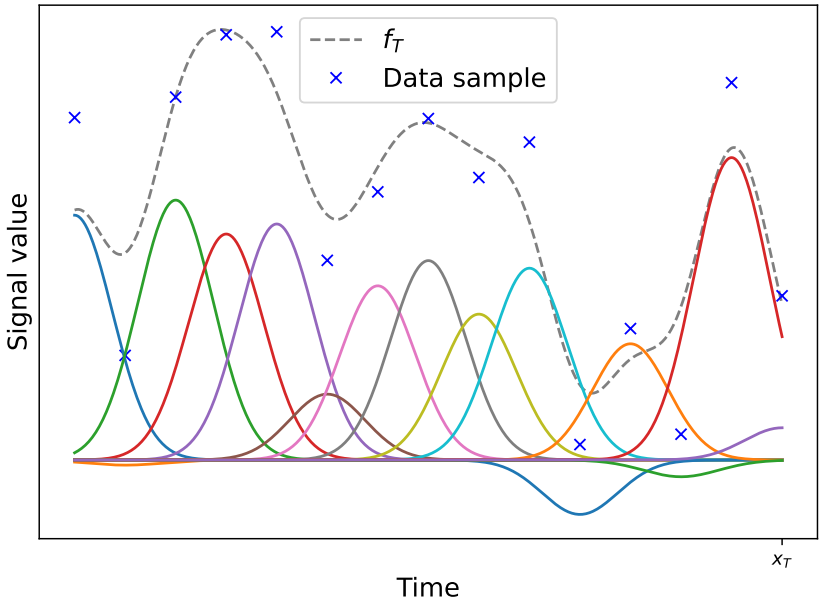
In the previous Ch. 3, we address the task of tracking signals from univariate time series of quantization intervals. However, some tasks may impose more demanding requisites, such as zero-delay response and smoothness in the reconstruction; for instance, online trajectory planning [67, 68], real-time control systems [69, 70], and high-speed digital to analog conversion [71], among others.
Unlike RKHS-based online methods, online interpolation methods can be suitable candidates for smooth signal reconstruction under zero-delay response and low model complexity requirements. This is because interpolation methods allow shaping piecewise-modeled signal estimates by sequentially assembling new function pieces. Piecewise polynomial functions, or splines, are arguably the most widely used ones [72, 73, 74, 75], presumably because of their model simplicity and approximation capabilities over functions of arbitrary complexity.

Based on this, Paper C presents a novel approach to zero-delay smoothing spline interpolation underpinned by a sequential decision-making framework [76]. This approach allows us to model the impact of each decision (i.e., each interpolated function piece) on future interpolations and, therefore, on the cumulative average cost, in this case, a roughness-based metric, resulting in a more accurate reconstruction. Then, an interpolation technique is proposed, assisted by an RNN that exploits the temporal dependencies between streamed data points to minimize the accumulated cost on average.
On the other hand, the method proposed in Paper C is unsuitable for quantized data or multivariate time series; these limitations are addressed in Paper D.
As a result, the proposed method in Paper D achieves one of the main goals of this thesis, namely, the zero-delay smooth signal reconstruction from streamed multivariate time series of quantization intervals.

The rest of this section presents the state-of-the-art on the zero-delay smooth



(a) Signal estimates and reconstruction proposed by NORMA. The reconstructed signal has inevitable discontinuities even though each of the successive signal estimates is continuous (and infinitely smooth).



(b) Every successive signal estimate, in this case, the Tth signal estimate, is constructed as a kernel expansion of overlapping terms. Each one of the expansion terms is plotted with a continuous colored line.

Figure 4.1: Example of the limitation of an online RKHS-based method, in this case, the naive online R_{reg} (regularized risk) minimization algorithm (NORMA) with a Gaussian kernel, for guaranteeing the smoothness of the reconstruction.

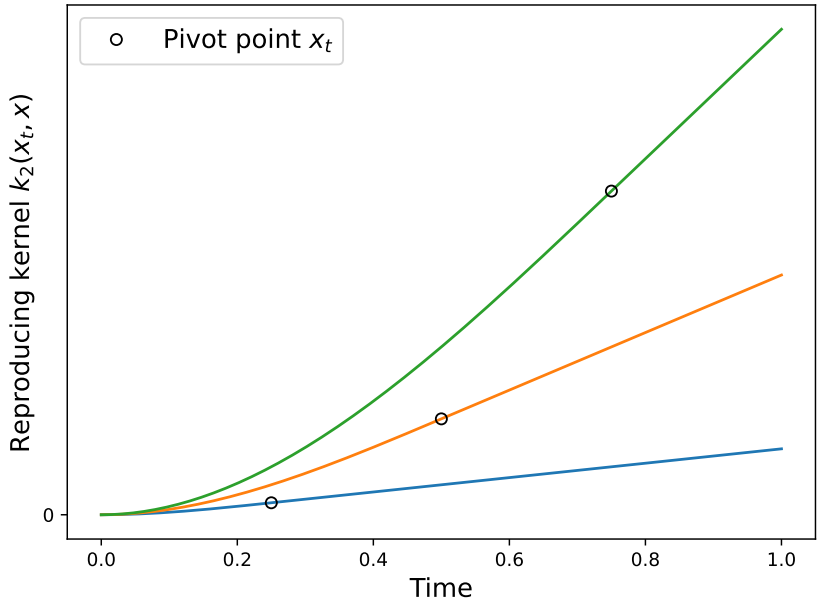


Figure 4.2: The reproducing kernel shown in (4.4) "centered" at different pivot points x_t.

signal reconstruction from streaming data points problem and further describes the limitations of RKHS-based methods for such tasks.

4.1.1 State-of-the-art

Some works have explored how to sequentially interpolate streaming data under real-time requirements using splines [25, 26]. However, the online methods developed in these works involve a sliding data window, which introduces a delay. This is because a delayed response allows them to successively correct the signal estimates as long as they are updated within the delay limits.
To the best of our knowledge, the myopic approach introduced in Ch. 1 is the only spline interpolation method that operates under a zero-delay response. However, this approach completely disregards any information that the past samples contain about the forthcoming data samples. As a result, signal reconstruction tasks under zero-delay response and smoothness requirements may be adversely affected since any of the current reconstructed signal portions can confine the future portions through the continuity constraints, thus affecting their accuracy through suboptimal results, possibly resulting in instabilities. This drawback is further explained in Paper C.

4.1.2 Limitation of RKHS-based Approaches

Online RKHS-based methods aim at yielding a sequence of signal estimates with regret guarantees [77]. To this end, they initially propose a signal estimate that is updated, possibly globally (along the whole timespan of the received data samples), as new data samples arrive. Their goal is to refine the signal estimate rather than to reconstruct new signal pieces smoothly. Therefore, neither smoothness nor continuity of the sequentially reconstructed signal is guaranteed.

As shown in Fig. 4.1, any spline-based signal estimate constructed as a reproducing kernel (basis) expansion suffers from overlapping between kernels. That is, if the tth expansion term $\alpha_t k(x_t, x)$ is non-zero, then, when added (or updated), it will affect the output of the overall expansion (signal estimate) and, hence, does not guarantee the continuity of the reconstruction. This happens for any valid kernel, and Fig. 4.1 illustrates it for the case of a Gaussian reproducing kernel.
As a closely related example, the smoothing spline interpolation problem introduced in Sec. 2.1.5.1 can be equivalently formulated using an RKHS framework [24]. That is,

$$\underset{f\in\mathcal{H}_\rho}{\text{minimize}} \sum_{t=1}^{T} \left(f(x_t) - y_t\right)^2 + \eta \|f\|^2_{\mathcal{H}_\rho}, \tag{4.1}$$

where (x_t, y_t) is the tth data point of a given sequence of T terms, $\eta \geq 0$ is the regularization hyperparameter, and $\mathcal{H}_\rho$ is the RKHS induced by the inner product

$$\langle f, h\rangle_{\mathcal{H}_\rho} = \int_0^1 D_x^\rho f(x)\, D_x^\rho h(x)\, dx, \tag{4.2}$$

with corresponding RKHS norm $\|f\|_{\mathcal{H}_\rho} = \sqrt{\langle f, f\rangle_{\mathcal{H}_\rho}}$. It is well known, thanks to the Representer Theorem [60], that the solution to (4.1) is of the form

$$f_T(x) = \sum_{t=1}^{T} \alpha_t k_\rho(x_t, x), \tag{4.3}$$

where $\alpha_t \in \mathbb{R}$, for $t = 1, \ldots, T$, are the kernel expansion coefficients and $k_\rho : [0, 1]^2 \to \mathbb{R}$ is the reproducing kernel associated with $\mathcal{H}_\rho$. For convenience, let us choose $\rho = 2$. Then, the reproducing kernel associated to $\mathcal{H}_2$ [24, 78] is given by

$$k_2(x_t, x) = \frac{1}{2}\max\{x_t, x\}\min^2\{x_t, x\} - \frac{1}{6}\min^3\{x_t, x\}. \tag{4.4}$$

The "shape" of the reproducing kernel in (4.4) is shown in Fig. 4.2. From the figure, one can notice that there is an overlap among different expansion terms over the whole time domain with non-zero values.

4.2 Problem Formulation

Paper C shows that interpolating a sequence of data points under a zero-delay response and smoothness requirements can be done with a given decision strategy or policy, where the resulting action takes the form of spline coefficients. From this observation, we pose the problem of finding the policy that achieves the lowest incurred cost on average from any time series of data points acquired from a given signal source, formulated as a policy search by cost minimization [45].
More specifically, the cost mentioned above consists of a weighted sum between the sum of squared residuals and a derivative-based measure of roughness, as in the smoothing spline interpolation problem described in (2.6), but from an action-state representation. All the remaining details are fully explained in Paper C.

Paper D formulates a similar problem to Paper C but is oriented to multivariate time series of uncertainty intervals and subject to consistency in the signal reconstruction. In this way, an additional component of the problem formulation in Paper D is how to properly extend the concept of consistency to an online setting.

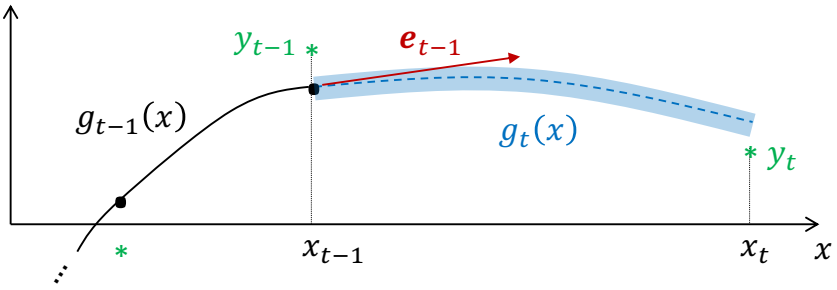


Figure 4.3: Action-state interpretation of the smoothing spline interpolation problem. The tth state $\boldsymbol{s}_t$ contains the necessary information to continue the interpolation task from time stamp x_{t-1}, that is, $\boldsymbol{s}_t = [x_{t-1}, x_t, y_t, \boldsymbol{e}_{t-1}^\top]^\top$. The tth action $\boldsymbol{a}_t$ are the coefficients of the tth function section g_t, and in our case the are computed through a parametric policy as $\boldsymbol{a}_t = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}_t)$.

In the ensuing sections, we use the content of Paper C as the common thread. Nevertheless, we include and expand the main contributions in Paper D when convenient.

4.3 Proposed Solution

The problem of finding the most suitable policy inside the space of candidate policies can be overwhelmingly complex due to the diversity and large number of available policies within [45]. Policy approximation techniques can help to reduce the policy space effectively. 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 and the smoothness requirement.
Specifically, in Paper C, we propose a family of parametrized policies through cost parametrization. The temporal dependencies are captured thanks to an RNN incorporated into the policy design, and the smoothness requirement is imposed as a constraint in a differentiable convex optimization layer (DCOL) [14].

Mathematically, the proposed policy takes the following form:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}_t) = \arg \min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s}_t) \subseteq \mathbb{R}^{d+1}} \left\{ \kappa_\eta(\boldsymbol{s}_t, \boldsymbol{a}) + \boldsymbol{J}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}; \boldsymbol{h}_t) \right\}, \tag{4.5}$$

where $\boldsymbol{s}_t$ represents the tth state representation of the interpolation task; in this case, $\boldsymbol{s}_t = [x_{t-1}, x_t, y_t, \boldsymbol{e}_{t-1}^\top]^\top$, where each vector $\boldsymbol{e}_{t-1} \in \mathbb{R}^{\varphi+1}$ contains $\varphi + 1$ continuity constraint values. Accordingly, $\mathcal{A}(\boldsymbol{s}_t)$ is the set of spline coefficients $\boldsymbol{a} \in \mathbb{R}^{d+1}$ that satisfy the continuity constraints in $\boldsymbol{e}_{t-1}$ at time step t. A more visual explanation is provided in Fig. 4.3.
On the other hand, κ_η is the action-state representation of the smoothing spline interpolation cost in (2.6), and computing $\boldsymbol{J}_{\boldsymbol{\theta}}$ involves evaluating an RNN with latent state at time step t denoted by $\boldsymbol{h}_t \in \mathbb{R}^H$. Moreover, κ_η, $\boldsymbol{J}_{\boldsymbol{\theta}}$ and $\mathcal{A}(\cdot)$ are chosen such that evaluating the policy in (4.5) consists in solving a convex optimization problem that admits a closed-form solution that can be differentiated.
In this way, we can understand the evaluation of $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ as a forward pass of a DCOL.

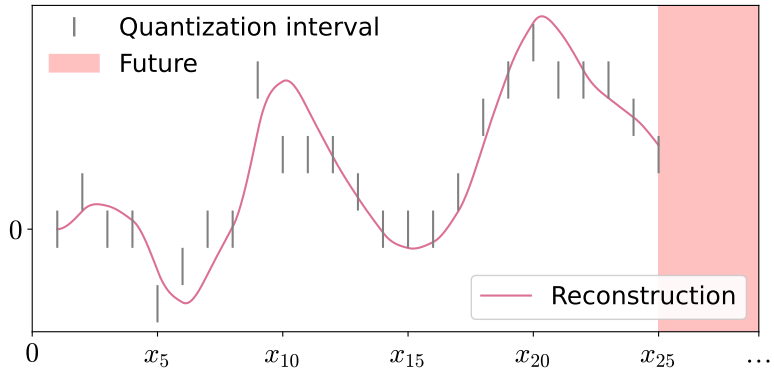


Figure 4.4: Snapshot at time x_{25} of a zero-delay smoothing spline interpolation using the policy proposed in Paper C. The signal reconstruction is done using only the center of the quantization intervals. Consequently, the policy is unaware of the quantization step sizes and thus cannot enforce consistency. The image is borrowed from Paper D for the sake of illustration.

4.3.1 Enforcing Consistency

The family of policies proposed in Paper C is only suitable for data points and not for any other uncertainty interval, including quantized data. As a result, the information contained within the quantization intervals is ignored, as shown in Fig. 4.4.
On the other hand, and as discussed in Sec. 2.6, consistent signal reconstruction methods [27, 28] exploit such information and have proved profitable (in the sense of implementation cost) in practice [57].

Paper D successfully extends the family of policies proposed in Paper C to deal with quantized data while ensuring consistency. This is done by formalizing the concept of consistent signal reconstruction from streaming data under zero-delay response requirements. In this way, our formulation generalizes the concept of consistency beyond offline settings rather than establishing a new one.

4.3.2 Multivariate Case

The family of candidate policies proposed in Paper C successfully exploits temporal relationships within univariate time series. However, they ignore any possible spatial relationships among multivariate time series. Paper D fixes this limitation by adequately extending the family of policies proposed in Paper C again. The overall idea is shown in Fig. 4.5.

4.4 Performance Analyses

Both of the proposed methods in Paper C and Paper D are compared with respect to the state-of-the-art, i.e., the myopic approach, and the best possible solution (obtained under an offline setting in practice) in terms of a roughness-based cost. The results show a considerable improvement with respect to the state-of-the-art.
On the other hand, the experimental results in paper D, describing the benefits

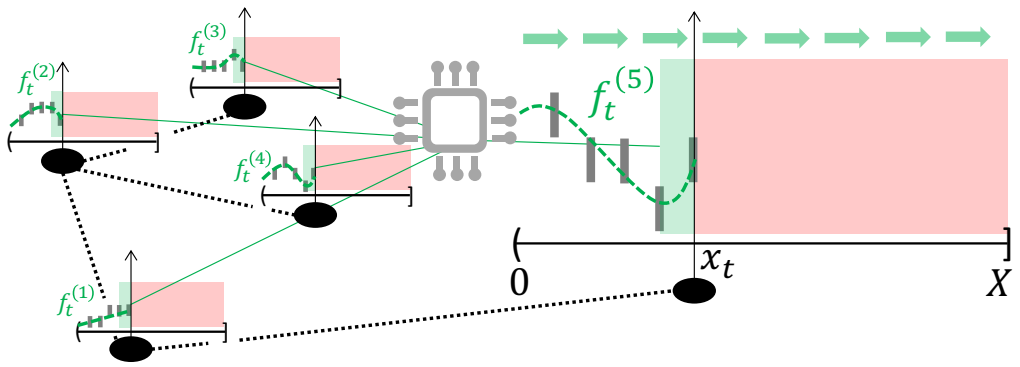


Figure 4.5: Snapshot at time x_t of a multivariate time series of quantization intervals (aligned in time across series) being reconstructed. The policy that achieves a zero-delay consistent signal reconstruction while accounting for the spatiotemporal dependencies among data samples is represented visually as a microprocessor. The upper indices (n) on the signal estimates refer to the series they reconstruct. The green shaded areas indicate the currently reconstructed portion of the multivariate signal, while the red ones represent the future. The image is borrowed from paper D for illustration purposes.

of zero-delay consistently reconstructed signals, agree with the expected behavior known for offline settings.

Specifically, we experimentally find that the MSE of a consistent zero-delay reconstruction f_{cons} decreases roughly at the same rate regardless of whether we reduce the quantization step size Δ or increase the oversampling ratio R (possible thanks to the finite rate of innovation of spline-based signals), as

$$\text{MSE}(f_{\text{cons}}) \propto \frac{\Delta^{0.99\pm0.09}}{R^{0.812\pm0.008}}. \tag{4.6}$$

Moreover, the error-rate decay in (4.6) roughly doubles (in logarithmic scale) the one achieved by an inconsistent zero-delay reconstruction f_{inc}, such as the one proposed in Paper C and illustrated in Fig. 4.4, with respect to the oversampling ratio R. Expressly,

$$\text{MSE}(f_{\text{inc}}) \propto \frac{1}{R^{0.435\pm0.002}}. \tag{4.7}$$

Finally, we observe that the proposed algorithms could not find stable policy configurations for certain polynomial orders and degrees of smoothness of the spline. However, these instabilities also manifest in the myopic configuration (existing before our method), hinting at a more fundamental problem that requires further research.

4.5 Contributions

This section summarizes the main contributions of Paper C (♢) and Paper D (♠).

♢ We rigorously formulate the problem of smoothing spline interpolation from streaming 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 response requirement). Due to its nature, it is formulated as a sequential decision-making problem.

- ♢ Unlike previously proposed zero-delay methods (e.g., the myopic approach, which is not trainable), our method trains a policy to minimize the smoothing interpolation problem cost metric on average. To capture the temporal, possibly long-term, dependencies between the streamed data samples and exploit them to reduce the cost metric on average, we incorporate an RNN able to capture the temporal signal dynamics.

- ♢ The proposed policy guarantees that the reconstructed signal is smooth. 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 the zero-delay smoothing spline interpolation approach over synthetic and real data. Additionally, we show how such an approach outperforms the state-of-the-art (namely, the myopic approach) zero-delay methods in terms of the smoothing spline interpolation problem cost metric on average.

- ♠ We formalize the concept of consistent signal reconstruction from streaming data under zero-delay response requirements. The resulting formulation generalizes the concept of consistency beyond offline settings.

- ♠ To the best of our knowledge, Paper D devises the first trainable method for zero-delay signal reconstruction from multivariate time series of quantization intervals, which can also enforce consistency.

- ♠ We show experimentally that the reconstruction error incurred by the zero-delay consistent signal reconstruction proposed method decays at the same rate by decreasing the quantization step size or increasing the oversampling ratio. Moreover, the error-rate decay slope with respect to the oversampling ratio doubles (in logarithmic scale) the one obtained with a similar zero-delay smooth signal reconstruction method that does not enforce consistency.

Chapter 5

Concluding Remarks

5.1 Conclusion

In this Ph.D. thesis, we address the problem of signal reconstruction from streamed multivariate time series of quantization intervals subject to zero-delay and smoothness requirements.
Although most existing algorithms can manage the problem requirements of estimating smooth signals or delivering a zero-delay response separately, addressing them together becomes challenging. This challenge is the core concept of this dissertation, and it has been solved incrementally through Papers A to D. A summary of the requirements satisfied by the proposed methods is provided in Table 5.1.

Table 5.1: Scope of the methods proposed in the compendium of papers.

	Zero-delay	Quantization	Smooth	Consistency	Multivariate
Paper A, B[1]	✔ [2]	✔	✖	✖	✖
Paper C	✔	✖	✔	✖	✖
Paper D	✔	✔	✔	✔	✔

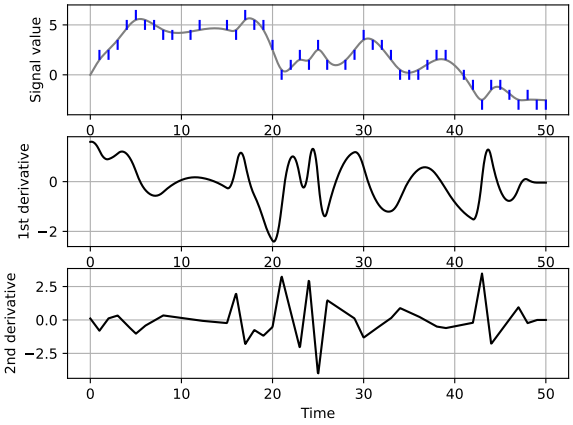
5.2 Future Work

The work discussed in this dissertation can serve as the foundation and motivation for some noteworthy research directions.

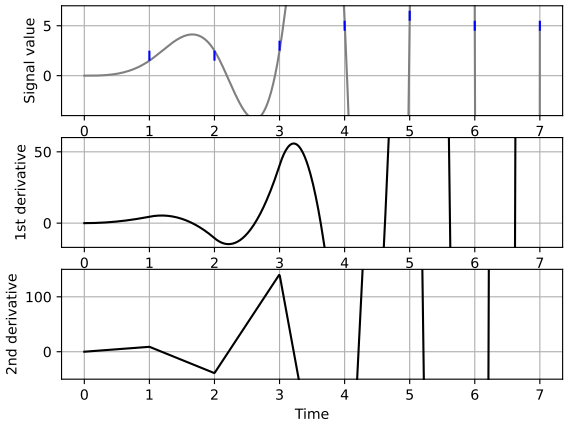
5.2.1 Fundamental Research on Zero-delay Smoothing Spline Instability Issues

We experimentally observe that the myopic policy described in Paper C is not stable for a degree of smoothness greater than 2, that is, smooth above the second derivative regardless of the spline order. We also observe instability under the myopic policy

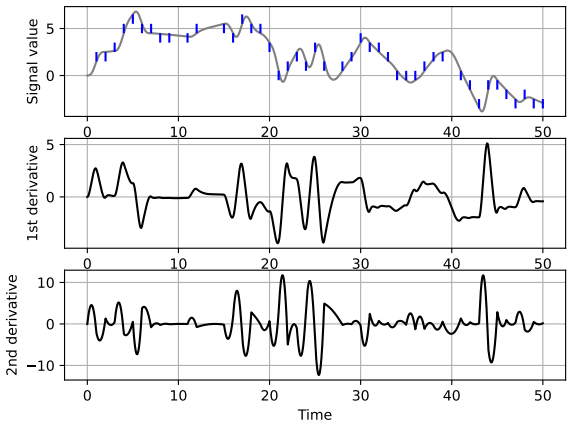
[1] Considering that the learning scheme is applied to the algorithm proposed in Paper A.
[2] The sliding window can be shrunk to the length of one data sample.



(a) Batch solution with a spline of order 3 and degree of smoothness 2. No observed instability issues.



(b) Myopic policy with a spline of order 3 and degree of smoothness 2. Instability is observed.



(c) Myopic policy with a spline of order 4 and degree of smoothness 2. No observed instability issues.

Figure 5.1: Instability issues suffered by the myopic policy against a stable configuration and the batch solution, for a degree of smoothness 2.

for a spline signal estimate of order 3 and degree of smoothness 2. This is illustrated in Fig. 5.1.
Our proposed methods in Paper C and Paper D are unstable for the same spline configurations since they implicitly use the myopic policy as a guided starting point.

This motivates further theoretical instability studies, the design of alternative policy architectures, or the exploration of low-delay approaches by relaxing the zero-delay constraint.

5.2.2 Real-time Safe Trajectory Planning

Suppose a trajectory planning problem is subject to passing through safety boxes of any dimension, e.g., spatial dimensions, angles, etc., at a given set of time instants. Then, the safety boxes can be seen as a set of temporarily aligned intervals, one for each dimension separately. In other words, a trajectory planning problem subject to safety boxes at certain time instants can be formulated as a signal reconstruction from a multivariate time series of uncertainty intervals that can overlap, as shown in Fig. 5.2.

For this reason, real-time safe trajectory planning tasks can benefit from the method proposed in Paper D.
This is because, in addition to passing through all safety boxes, even when the next safety box is revealed only one time step ahead, the proposed method can ensure smoothness, e.g., a continuous acceleration, and low-model complexity, e.g., damped oscillations.

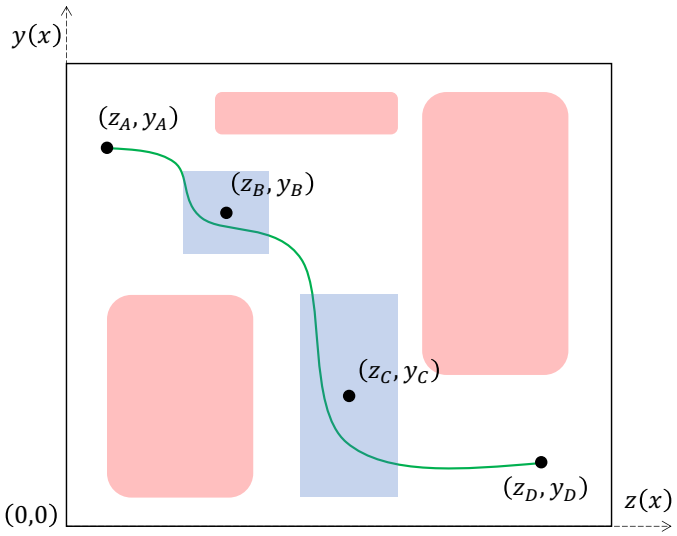
5.2.3 Compression for Reconstruction and Inference

Piecewise polynomial approximation techniques can be used for time-series compression [79]. Normally, they divide a time series into several segments of variable length and find the best polynomial approximating them (according to some metric). One of their main advantages is that, despite the compression being lossy, a maximum deviation from the original (uncompressed) data can be fixed in advance to enforce a certain reconstruction accuracy.

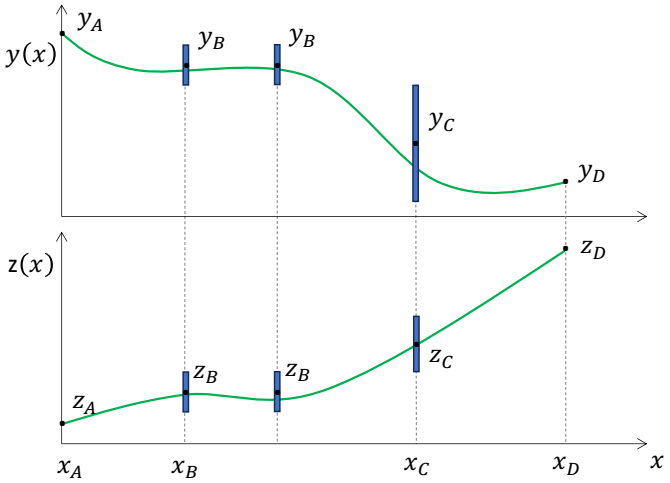
We believe our proposed technique in Paper D can be readily adapted to compress streaming time-series data under a maximum reconstruction deviation and real-time constraints once the stability issue is fixed. The compressed space would consist of a set of interdependent spline coefficients due to the smoothness constraints (fixing some degrees of freedom); therefore, only a few coefficients must be stored.
For example, suppose we want to use a quadratic spline with tth function section given by $g_t(x) = a_{t,1} + a_{t,2}(x - x_{t-1}) + a_{t,3}(x - x_{t-1})^2$ and continuity up to the first derivative, i.e., the following recursive relations

$$g_t(x_t) = g_{t+1}(x_t) \to a_{t+1,1} = a_{t,1} + a_{t,2}u_t + a_{t,3}u_t^2, \tag{5.1a}$$
$$g_t'(x_t) = g_{t+1}'(x_t) \to a_{t+1,2} = a_{t,2} + 2a_{t,3}u_t, \tag{5.1b}$$

are satisfied, where $u_t \triangleq x_t - x_{t-1}$. Then the coefficients $\{a_{t,1}\}_t$ and $\{a_{t,2}\}_t$ can be obtained from an initial tuple $\{a_{0,1}, a_{0,2}, a_{0,3}\}$, and the sequences $\{x_t\}_t$ and $\{a_{t,3}\}_t$ through the recursive relations in (5.1).
From here, notice that if most of the stored coefficients $\{a_{t,3}\}_t$ are zero-valued, we can achieve competitive compression ratios. This can be done, for instance, by



(a) Two-dimensional trajectory. The blue squared boxes represent safety regions that must be visited. The red rounded boxes represent dangerous areas that should be avoided.



(b) Decomposition of the 2-dimensional trajectory from Fig. 5.2a in a multivariate time series. The safety boxes become uncertainty intervals that can overlap.

Figure 5.2: Two-dimensional representation of a zero-delay safe trajectory planning task. The method proposed in Paper D can ensure that the trajectory passes through the safety regions at a given instant and promotes low-model complexity and smoothness.

promoting sparsity through a cost term.
In this case, the reconstruction step would recover all coefficients via (5.1) and then evaluate the spline model. Moreover, it would not only recover the original data but an estimate of intermediate points with a time resolution as high as desired.

On the other hand, some tasks may benefit from working directly in the compressed state (without the need for the reconstruction step). For those tasks, we might be interested in promoting a certain task-dependent performance metric and not only sparsity. We refer to this type of compression as compression for inference, and Fig. 5.3 illustrates how the architecture of our model can be adapted for such purposes.

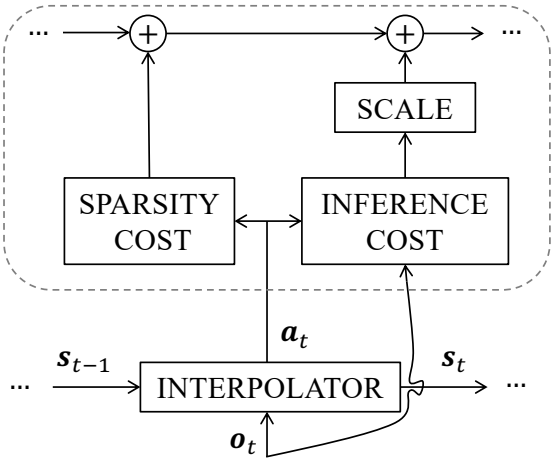


Figure 5.3: Architecture prototype of the proposed spline-based compressor for inference. The INTERPOLATOR cell may contain trainable parameters, and it ensures a spline reconstruction under a given maximum deviation. The SCALE cell acts as a user-defined trade-off hyperparameter. The notations $\boldsymbol{s}_t$, $\boldsymbol{a}_t$, and $\boldsymbol{o}_t$ refer to the tth state, action, and observation, respectively. The components in the gray dashed rounded box are used only during the training phase.

Appendix A

Paper A

Title:	Tracking of quantized signals based on online kernel regression
Authors:	**Emilio Ruiz-Moreno** and Baltasar Beferull-Lozano
Conference:	IEEE International Workshop on Machine Learning for Signal Processing

---

Appendix B

Paper B

Title: An Online Multiple Kernel Parallelizable Learning Scheme

Authors: **Emilio Ruiz-Moreno** and Baltasar Beferull-Lozano

Journal: IEEE Signal Processing Letters

Appendix C

Paper C

Title: A Trainable Approach to Zero-delay Smoothing Spline Interpolation

Authors: **Emilio Ruiz-Moreno**, Luis Miguel López-Ramos and Baltasar Beferull-Lozano

Journal: IEEE Transactions on Signal Processing

A Trainable Approach to Zero-delay Smoothing Spline Interpolation

Emilio Ruiz-Moreno, Luis Miguel López-Ramos and Baltasar Beferull-Lozano

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.**

C.1 Introduction

Online learning has been studied and applied in a broad range of research fields, including optimization theory [30, 63, 64], signal processing [121], and machine learning [122, 45, 44]. Within these fields, online methods generating a series of estimates from sequentially streamed data are especially useful to reduce complexity in large-scale problems [19], to dynamically adapt to new patterns in the data [20], and to enable acting under real-time requirements [21].

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 [123]; 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 [124] 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

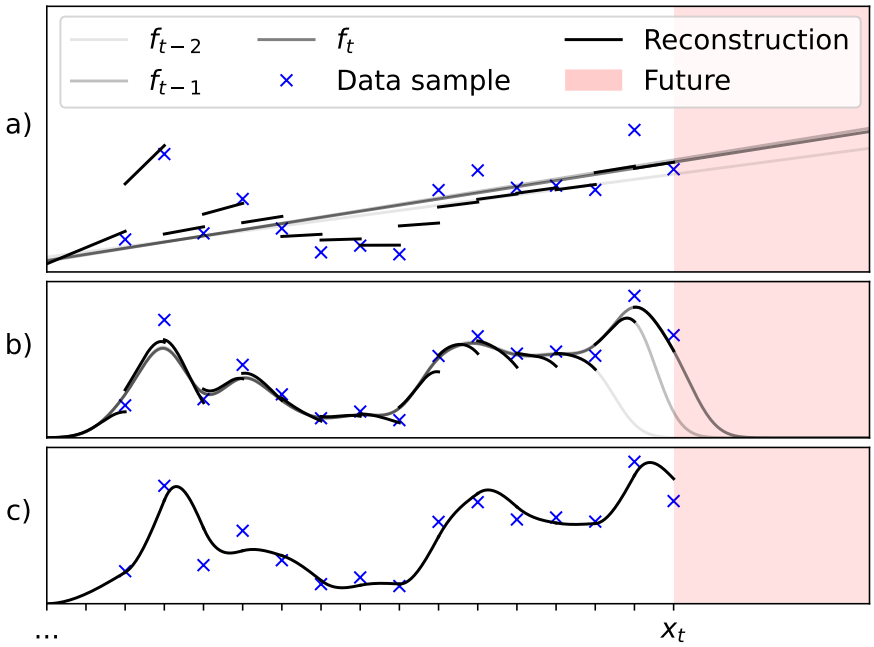


Figure C.1: A signal reconstructed from stream data points by different methods and models. The symbols x_t and f_t represent the current time and signal estimate, respectively. a) Recursive least squares [10, 11] with a linear model. b) NORMA [12] with Gaussian kernels. c) Smoothing myopic interpolation with cubic Hermite splines [13].

online trajectory planning [67, 68], real-time control systems [69, 70], and high-speed digital to analog conversion [71], 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 [60, 1] and online Gaussian processes [125, 126]. They aim at yielding a sequence of signal estimates with convergence guarantees or sublinear regret [77]. 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. C.1a) and C.1b). 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. C.1c). Among them, piecewise polynomial functions, also known as *splines*, are arguably the most representative ones [73, 75]. Actually, splines have been used since ancient times [74], long before their mathematical foundations were even established [72],

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 [24, 33, 127], 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 [25, 26] 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 [26], 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 [128] 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) [14] on top of a recurrent neural network (RNN) [129, 130]. 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

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

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.

The rest of the paper is structured as follows: Sec. C.2 introduces the notation and presents some basic concepts and definitions. Then, in Sec. C.3, we provide our problem formulation. Next, in Sec. C.4 and Sec. C.5, we respectively provide a solution, a benchmark, and a baseline. Thereafter we experimentally validate our solution in Sec. C.6. Finally, Sec. C.7 concludes the paper.

C.2 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.

C.2.1 Notation

Vectors and matrices are denoted by bold lowercase and capital letters, respectively. Given a vector $\boldsymbol{v} = [v_1, \dots, v_C]^\top$, 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 $[v_i, \dots, v_j]^\top \in \mathbb{R}^{j-i+1}$. We use Euler's notation for the derivative operator; thus, D_x^k denotes the kth derivative over the variable x.

C.2.2 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.2.3 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

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

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

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

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

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

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 $\mathrm{dom}(f_T) = \bigcup_{t=1}^{T} \mathcal{T}_t$. 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 (C.1), we can enforce its degree of smoothness to be $\varphi \leq d$ by imposing the following equality constraint

$$[\boldsymbol{a}_t]_{1:\varphi+1} = \boldsymbol{e}_{t-1}, \tag{C.4}$$

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

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

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 Appendix C.8.

C.2.4 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

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

known as smoothing spline interpolation [131, 6]. The name is due to the unique solution to the optimization problem (C.6) being a spline conformed of T function sections, as in (C.1). More specifically, the solution of (C.6) is a spline of order $2\rho - 1$ with $2\rho - 2$ continuous derivatives and natural boundary conditions [24]. 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 (C.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 (C.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 (C.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. 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.

C.3 Problem formulation

Once the problem data, the description of spline-based signal estimates, and the smoothing spline interpolation problem have been introduced in Sec. C.2, 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) [46] 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.

[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 [132]. However, they are out of the scope of this paper, as we justify in the ensuing Sec. C.6.2.

C.3.1 Characterization of the problem data

In our problem, the data described in Sec. C.2.2 are observed sequentially. Before every tth time step, the observation about to be received $\boldsymbol{o}_t$ 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 $P_{Y_{\mathcal{X}}}$.

C.3.2 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 $\boldsymbol{s}_t \in \mathcal{S}$ is constituted by the corresponding observation $\boldsymbol{o}_t$, and the condition at which the reconstruction was left, which is specified by the vector $\boldsymbol{e}_{t-1}$ whose components are given as in (C.5), and the time instant x_{t-1}. Formally, every tth state $\boldsymbol{s}_t$ is expressed as $\boldsymbol{s}_t = [\boldsymbol{o}_t^\top, \boldsymbol{e}_{t-1}^\top, x_{t-1}]^\top$. Since every state $\boldsymbol{s}_t$ is uniquely determined once the spline coefficients $\boldsymbol{a}_{t-1}$ are fixed, we can explicitly describe the state update mechanism, by means of a deterministic mapping, as

$$\boldsymbol{s}_{t+1} = F\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}\right). \tag{C.7}$$

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

C.3.3 Action space

Immediately after receiving the tth observation, we propose a function section as in (C.2), and we implicitly select the spline coefficients $\boldsymbol{a}_t$. This is because the function section is determined as soon as $\boldsymbol{a}_t$ is chosen (the basis vector defined in (C.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, $\boldsymbol{a}_t \in \mathcal{A} \subseteq \mathbb{R}^{d+1}$ 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 (C.4). Notice that the set of admissible actions depends on the current state. Therefore, we accordingly denote the admissible action space as $\mathcal{A}(\boldsymbol{s}_t)$.

C.3.4 Policy space

A policy $\pi = \{\boldsymbol{\mu}_t : \mathcal{S} \to \mathcal{A}\}_{t \in \{1,2,\ldots\}}$ 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 $\pi(\boldsymbol{s}_t) \in \mathcal{A}(\boldsymbol{s}_t)$ for all time steps are termed *admissible policies*. Separately, *stationary policies* are policies that do not change over time, i.e., $\boldsymbol{\mu} \equiv \boldsymbol{\mu}_t = \boldsymbol{\mu}_{t+1}$ for all time steps. Hence, a stationary policy is unequivocally defined by the mapping $\boldsymbol{\mu}$. 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. C.2.4. To this end, we resort to parametric policy approximation [45] denoting any approximated stationary policy as $\boldsymbol{\mu}_{\boldsymbol{\theta}}$, where the vector $\boldsymbol{\theta} \in \mathbb{R}^P$ 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.

C.3.5 Total expected cost

The following **Proposition 2** shows that the smoothing spline interpolation objective introduced in Sec. C.2.4, equation (C.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 (C.6) can be equivalently computed additively as

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

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

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

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

Proof: See Appendix C.9.

Based on **Proposition 2**, we can express the objective of the smoothing spline interpolation problem (C.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

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

where $\boldsymbol{M}_t$ is constructed as in (C.9). This is because each tth state-action pair contains all necessary information. From here and under a given policy of interest $\boldsymbol{\mu}_{\boldsymbol{\theta}}$, as described in Sec. C.3.4, the metric

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

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 (C.7). The expectation in (C.11) is performed over the random process modeling the dynamics of the environment through the observations within the states.

C.3.6 Policy search by cost optimization

Computing the expectation in (C.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

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

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

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}_{m,t}) \in \mathcal{A}(\boldsymbol{s}_{m,t}), \forall m, t, \tag{C.12c}$$

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.

C.4 Proposed solution

The previous Sec. C.3 has provided the necessary definitions and considerations to arrive at a rigorous problem formulation. An exact solution to the problem (C.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 (C.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) [48], a gradient computation technique borrowed from the deep learning literature [47]. Our proposed solution can effectively solve the problem formulated in Sec. C.3.6 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. C.6.2.

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. C.3.6.

C.4.1 Policy form

Parametric policy approximation via parametric cost function approximation (CFA) [45] is a method that seeks through the policy space, in our case Π, among those

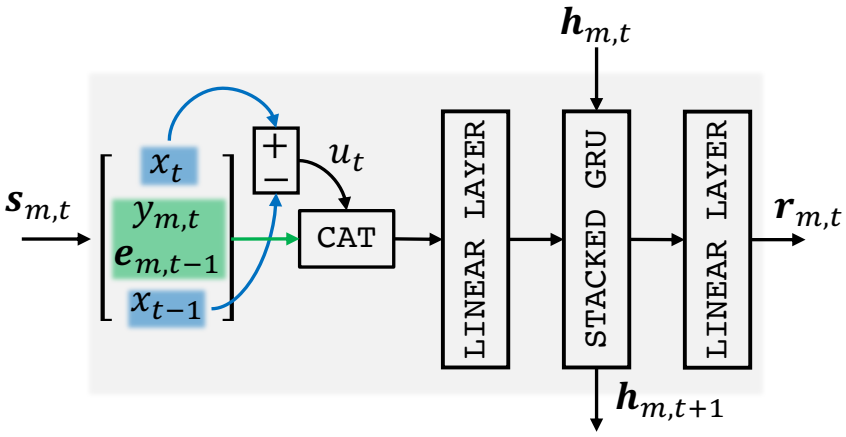


Figure C.2: Visual representation of our RNN architecture satisfying the relation in (C.15). The elements onto the gray shaded area constitute the mapping $R_{\boldsymbol{\theta}'}$. The CAT cell stands for a concatenation operation.

policies defined as an optimization problem with parametrized objectives. In this work, we are interested in CFA-based policies of the form

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

where the map κ denotes the cost described in (C.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 [133, 134, 135].

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

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

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

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

with $\boldsymbol{\theta} = [\lambda, \boldsymbol{\theta}'^{\top}]^{\top}$ and $\boldsymbol{\theta}' \in \mathbb{R}^{P-1}$, exemplified in Fig. C.2. From now on, we refer to the policy in (C.13) with cost-to-go as in (C.14) as the RNN-based policy. Finally, notice that besides being parametric, the RNN-based policy is admissible and stationary by design.

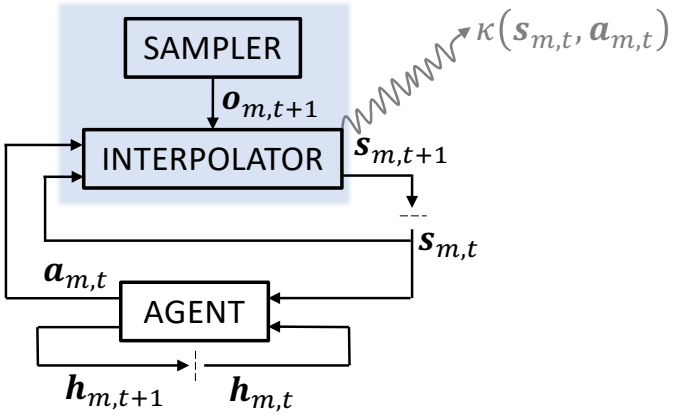


Figure C.3: Rolled representation of the environment-agent system. The sampler cell samples the random process modeling the dynamics of the environment. The interpolator cell performs the reconstruction, evaluates the cost, and updates the states. The agent cell contains the policy and the RNN. The blue-shaded area encompasses the environment. The cost (in gray) is used during the training phase but not for evaluation.

C.4.2 Policy evaluation

Evaluating the proposed policy involves solving the optimization problem stated in (C.13). Notice that both the cost κ built as in (C.10), and the cost-to-go approximation $J_{\boldsymbol{\theta}}$ described in (C.14), are convex with respect to the actions and hence, the objective in (C.13) is convex too. Moreover, the admissible action set, described in Sec. C.3.3, is convex. Therefore, the optimization problem in (C.13) is convex thus, any locally optimal action is globally optimal [113].

Additionally, the optimization problem in (C.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 Appendix C.10 for the derivation of the closed-form evaluation.

C.4.3 Policy training

As explained in Sec. C.4.1, we have reduced the search space of problem (C.12) by restricting the policy space to a family of policies of the form given in (C.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 (C.12) is commonly referred to as policy training. Unfortunately, the objective (C.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. C.4.2 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 [136]. This point of view is schematized in Fig. C.3, where traveling the given mth series, by following a policy $\boldsymbol{\mu}_{\boldsymbol{\theta}}$, allows to construct the cumulative objective in (C.12a) used for training. Ad-

ditionally, and thanks to the closed-form policy evaluation discussed in Sec. C.4.2, 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 [137] or using specific numerical tools for DCOLs such as CVXPY Layers [14].

C.5 Benchmark and baseline methods

Recall from Sec. C.2.4 that the batch formulation provides the optimal reconstruction with hindsight. The batch solution can be found by solving the optimization problem (C.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.

C.5.1 Myopic benchmark

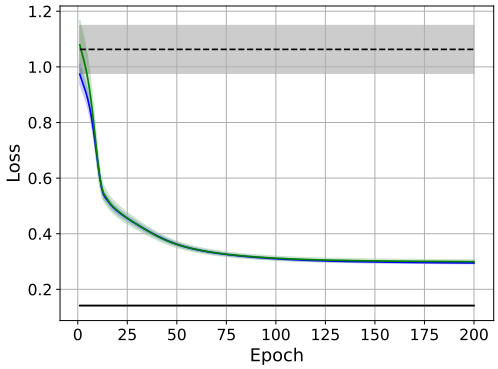
A policy that chooses the action that minimizes the current or instantaneous cost is commonly referred to as myopic. It can be constructed as

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

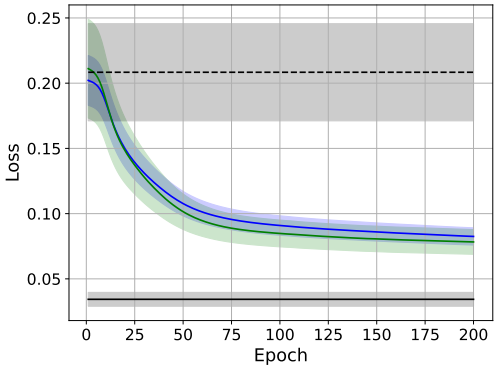
with cost κ as in (C.10) and admissible action set as described in Sec. C.3.3. Notice that since the myopic policy does not contain trainable parameters, it does not need to be trained. Moreover, the myopic approach is carried out as a parameterless CFA-based policy, hence, becoming a particular case of (C.13). For this reason, it also admits a unique and closed-form evaluation. See Appendix C.10 for more details.

C.6 Experiments

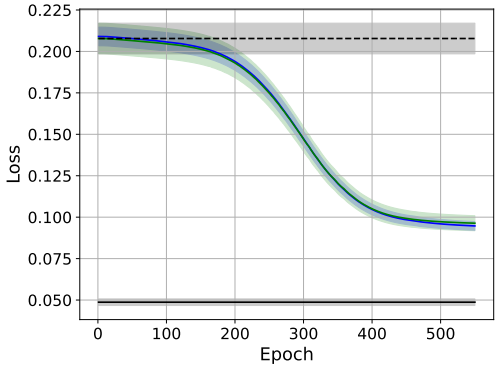
In this section, we experimentally validate the effectiveness of the proposed RNN-based policy, introduced in Sec. C.4. To this end, we first describe the time-series datasets used. Then, we outline the possible policy configurations, i.e., the possible types of splines as well as the RNN architecture. Afterward, we report how the experiments have been carried out. Finally, we present and comment on the experimental results.



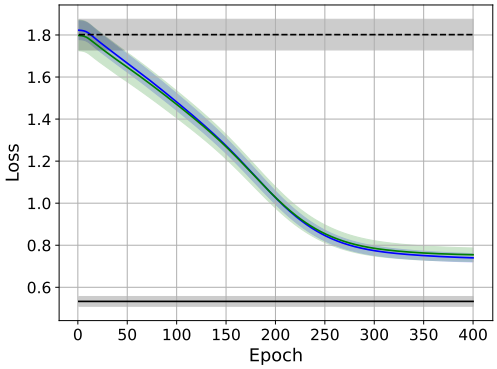
(a) (4,2), $\eta = 10$, and synthetic dataset.



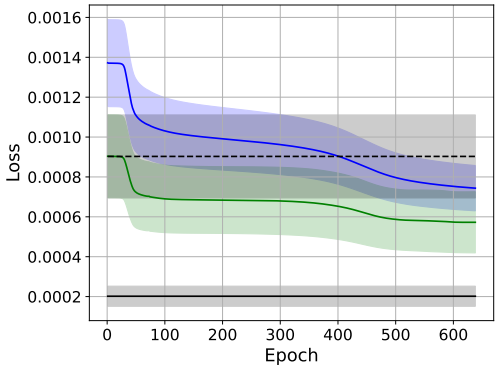
(b) (3,1), $\eta = 1$, and R1.



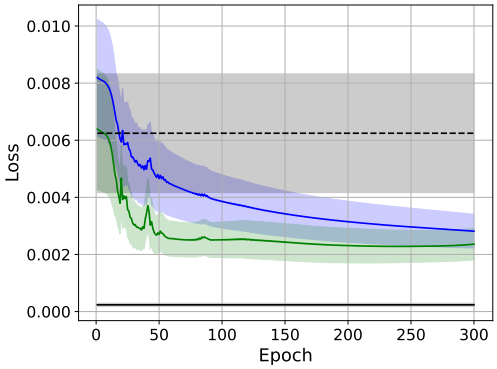
(c) (4,2), $\eta = 10$, and R2.



(d) (4,2), $\eta = 1$, and R3.



(e) (4,2), $\eta = 1$, and R4.



(f) (3,1), $\eta = 10$, and R5.

Figure C.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. C.3.5). The shaded areas represent one standard deviation.

Table C.1: Performance metrics averaged over the test partitions, for different η values and policy configurations with $\rho = 2$. Recall that the improvement metric (C.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.

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

C.6.1 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 [138] 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) [139]. 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) [140], 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) [141] 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) [142] 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. C.6.4.

C.6.2 Policy configuration

We experimentally observe that the myopic policy described in Sec. C.5.1 is not stable for values of $\rho > 2$. Recall that the value of ρ affects the policy cost, set as in (C.10), and delimits the order and degree of smoothness of the spline signal estimate, as explained in Sec. C.2.3. 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 (C.16) and (C.13) with a near-zero initial value of λ. Although further 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

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

configurations, within the search function space described in Sec. C.2.4, 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. C.4.1 and illustrated in Fig. C.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 [51], 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.6.3 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 [143], 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).

C.6.4 Results and discussion

Some of the training-validation curves are presented in Fig. C.4. As expected, we observe that randomly initialized RNN-based policies (except for the parameter λ, which controls the length of the initial performance gap, as discussed in Sec. C.6.2, 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.

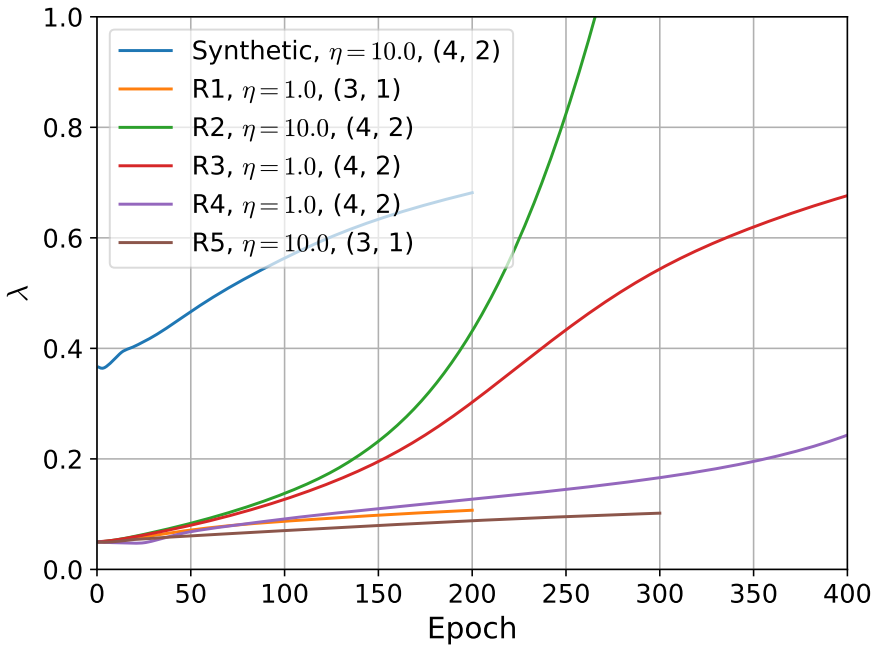


Figure C.5: Training curves of the parameter $\lambda \in \mathbb{R}_+$ introduced in (C.14). The elements displayed coincide with those shown in Fig. C.4.

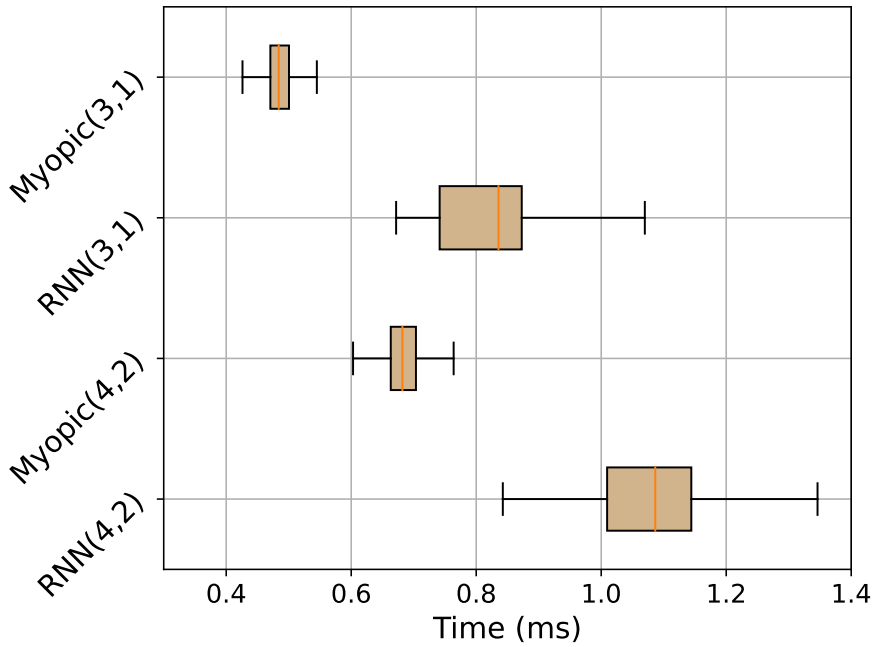


Figure C.6: Box plot of the policy execution time per interpolation step over the test partitions and η values $\{0.1, 1, 10\}$. It illustrates (excluding outliers) the minimum, first quartile, median, third quartile, and maximum.

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

$$I = \frac{\ell_M - \ell_R}{\ell_M - \ell_B}, \tag{C.17}$$

where ℓ_M, ℓ_R, ℓ_B denote the loss metric displayed in Fig. C.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.,

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

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 C.1. From Fig. C.4 and Table C.1, 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 C.1 also shows standard performance descriptors such as the mean squared error (MSE) and mean absolute error (MAE). See Appendix C.11 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 (C.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. C.2) in the reconstruction task. As an illustration, Fig. C.5 shows the training curves corresponding to the parameter λ for the policy configurations presented in Fig. C.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. C.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. C.5.1), and our configuration of the RNN-based policy (see Sec. C.6.2) contains approximately 3400 trainable parameters, which is arguably a reduced model size for most tasks.

Finally, and for the sake of completeness, Fig. C.7 shows a snapshot of a zero-delay smooth signal reconstruction alongside its two first derivatives using our proposed method.

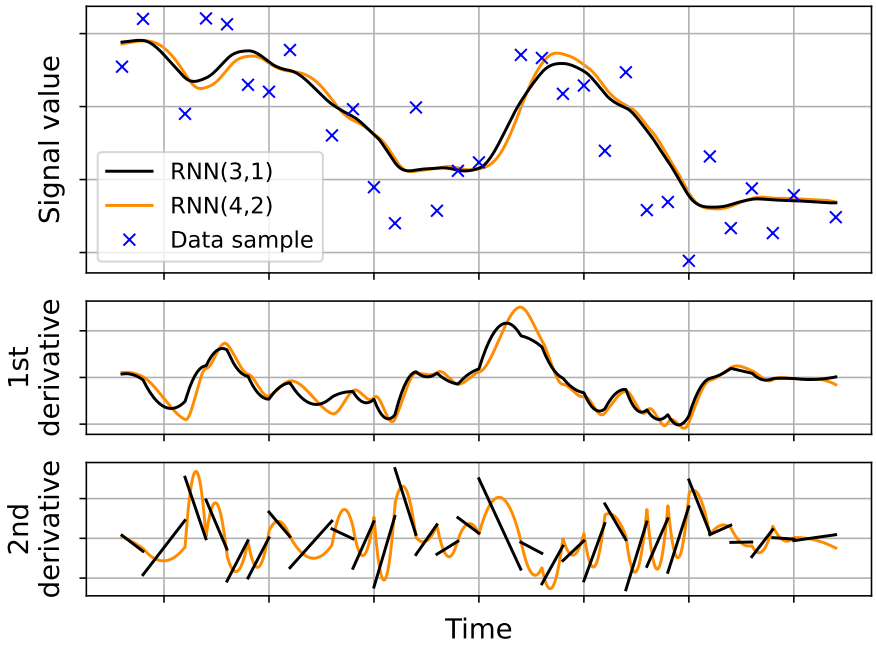


Figure C.7: Snapshot of a reconstructed time series from the test partition of the synthetic dataset. Both of the presented policies have been trained over the training partition before reconstruction.

C.7 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.

C.8 Proof of Proposition 1

Recall from Sec. C.2.3 that every spline f_T, as in (C.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 (C.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

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

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

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

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

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

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 straightforwardly computed as

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

Now observe that

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

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

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

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

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

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

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

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 (C.25b) uses the relations described in (C.20) and (C.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 (C.19) by $k!$, using the relations derived in (C.24) and (C.25), and appropriately renaming the indices we obtain the **Proposition 1**.

C.9 Proof of Proposition 2

Recall from Sec. C.2.4 that the solution to the optimization problem stated in (C.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 (C.19). From here, we can equivalently compute the regularization term in the objective in (C.6) (second term) as

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

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

$$\begin{aligned}\int_{\mathcal{T}_t} \left(D_x^\rho\, g_t(x)\right)^2 dx &= \int_{\mathcal{T}_t} \left(D_x^\rho\, \boldsymbol{a}_t^\top \boldsymbol{p}_t(x)\right)^2 dx &\text{(C.27a)}\\ &= \boldsymbol{a}_t^\top \boldsymbol{M}_t \boldsymbol{a}_t, &\text{(C.27b)}\end{aligned}$$

with

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

From the relation in (C.22), it is clear that the first ρ rows and columns of the matrix defined in (C.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

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

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

$$\sum_{t=1}^{T} \left(f_T(x_t) - y_t\right)^2 = \sum_{t=1}^{T} \left(g_t(x_t) - y_t\right)^2, \tag{C.30}$$

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

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

C.10 Closed-form policy evaluation

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

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

Table C.2: Terms in (C.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 C.2. The form in (C.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 (C.4) and satisfied by the actions in the admissible set $\mathcal{A}(\boldsymbol{s}_t)$) in the objective of (C.31), by restating

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{e}_{t-1} \\ \boldsymbol{0}_{d-\varphi} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{\varphi+1} \\ \boldsymbol{\alpha} \end{bmatrix}, \tag{C.32}$$

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 (C.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

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

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

$$\boldsymbol{\alpha}_t = \arg\min_{\boldsymbol{a}\in\mathbb{R}^{d-\varphi}} \{\boldsymbol{\alpha}^\top \boldsymbol{B}_2^\top \boldsymbol{A}_t \boldsymbol{B}_2 \boldsymbol{\alpha} \tag{C.34a}$$

$$+ 2\left(\boldsymbol{e}_{t-1}^\top \boldsymbol{B}_1^\top \boldsymbol{A}_t \boldsymbol{B}_2 + \boldsymbol{b}_t^\top \boldsymbol{B}_2\right)\boldsymbol{\alpha}\}, \tag{C.34b}$$

with closed-form solution given by

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

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

C.11 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 [144], 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 $\mathcal{B} \cap \overline{\mathcal{B}} = \emptyset$

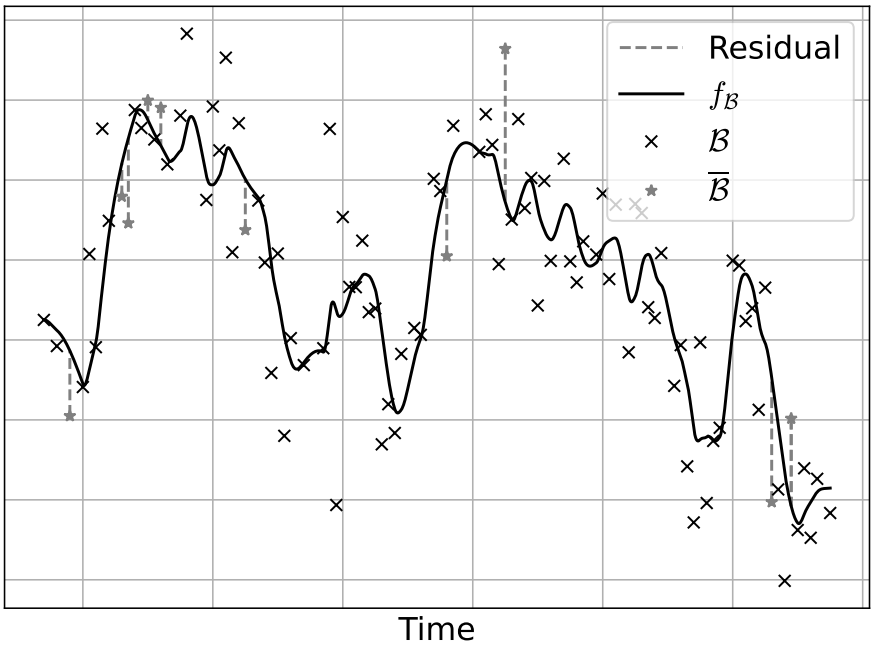


Figure C.8: Illustration of the residuals used to estimate the MSE via bootstrapping. The test partition comes from the synthetic data and has been reconstructed with a Myopic(3,1) policy with $\eta = 1$ and $\rho = 2$. $\mathcal{B}$ contains 90% of the test partition and $\overline{\mathcal{B}}$ the remaining 10%.

to estimate the MSE and MAE performance metrics. Mathematically, this can be expressed as:

$$\mathrm{MSE}\left(f_{\mathcal{B}}\right) = \frac{1}{|\overline{\mathcal{B}}|} \sum_{i \in \overline{\mathcal{B}}} \left(f_{\mathcal{B}}(x_i) - \psi(x_i)\right)^2, \tag{C.36a}$$

$$\mathrm{MAE}\left(f_{\mathcal{B}}\right) = \frac{1}{|\overline{\mathcal{B}}|} \sum_{i \in \overline{\mathcal{B}}} \left|f_{\mathcal{B}}(x_i) - \psi(x_i)\right|, \tag{C.36b}$$

where $f_{\mathcal{B}}$ is the signal estimate constructed from the test data subset $\mathcal{B}$. This procedure is illustrated in Fig. C.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 $\mathcal{B}, \bar{\mathcal{B}}$ with replacement (i.e. they may repeat) and average the result. Particularly, we perform 10 repetitions.

Appendix D

Paper D

Title:	Consistent Signal Reconstruction from Streaming Multivariate Time Series
Authors:	**Emilio Ruiz-Moreno**, Luis Miguel López-Ramos and Baltasar Beferull-Lozano
Journal:	Submitted to IEEE Transactions on Signal Processing

Bibliography

[1] E. Ruiz-Moreno and B. Beferull-Lozano, "Tracking of quantized signals based on online kernel regression," in *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2021.

[2] E. Ruiz-Moreno and B. Beferull-Lozano, "An online multiple kernel parallelizable learning scheme," *IEEE Signal Processing Letters*, 2023.

[3] E. Ruiz-Moreno, L. M. López-Ramos, and B. Beferull-Lozano, "A trainable approach to zero-delay smoothing spline interpolation," *IEEE Transactions on Signal Processing*, vol. 71, pp. 4317–4329, 2023.

[4] E. Ruiz-Moreno, L. M. López-Ramos, and B. Beferull-Lozano, "Zero-delay consistent signal reconstruction from streamed multivariate time series," *arXiv preprint arXiv:2308.12459*, 2023.

[5] S. Mollaebrahim, B. Beferull-Lozano, and E. R. Moreno, "Decentralized subspace projection for asymmetric sensor networks," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pp. 1–5, IEEE, 2020.

[6] S. N. Wood, *Generalized additive models: an introduction with R*. chapman and hall/CRC, 2006.

[7] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[8] Q. T. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat, and A. Rehman, "Sentiment analysis using deep learning techniques: a review," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017.

[9] H. Sharma, M. Agrahari, S. K. Singh, M. Firoj, and R. K. Mishra, "Image captioning: a comprehensive survey," in *2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC)*, pp. 325–328, IEEE, 2020.

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

[11] M. H. Hayes, *Statistical digital signal processing and modeling*. John Wiley & Sons, 1996.

[12] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE transactions on signal processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[13] C. De Boor and C. De Boor, *A practical guide to splines*, vol. 27. springer-verlag New York, 1978.

[14] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[15] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama, "Machine learning for streaming data: state of the art, challenges, and opportunities," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, pp. 6–22, 2019.

[16] I. Roşu, "Fast and slow informed trading," *Journal of Financial Markets*, vol. 43, pp. 1–30, 2019.

[17] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[18] H. Boström, S. F. Andler, M. Brohede, R. Johansson, A. Karlsson, J. Van Laere, L. Niklasson, M. Nilsson, A. Persson, and T. Ziemke, "On the definition of information fusion as a field of research," 2007.

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

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

[21] R. Nishihara, P. Moritz, S. Wang, A. Tumanov, W. Paul, J. Schleier-Smith, R. Liaw, M. Niknami, M. I. Jordan, and I. Stoica, "Real-time machine learning: The missing pieces," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pp. 106–110, 2017.

[22] S. Van Vaerenbergh and I. Santamaría, "Online regression with kernels," *Regularization, Optimization, Kernels, and Support Vector Machines*, pp. 477–501, 2014.

[23] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing kernel hilbert spaces," in *Academic Press Library in Signal Processing*, vol. 1, pp. 883–987, Elsevier, 2014.

[24] G. Wahba, *Spline models for observational data*. SIAM, 1990.

[25] J. M. de Carvalho and J. V. Hanson, "Real-time interpolation with cubic splines and polyphase networks," *Canadian Electrical Engineering Journal*, vol. 11, no. 2, pp. 64–72, 1986.

[26] R. Debski, "Real-time interpolation of streaming data," *Computer Science*, vol. 21, no. 4, 2020.

[27] N. T. Thao and M. Vetterli, "Deterministic analysis of oversampled a/d conversion and decoding improvement based on consistent estimates," *IEEE Transactions on signal processing*, vol. 42, no. 3, pp. 519–531, 1994.

[28] I. Jovanovic and B. Beferull-Lozano, "Oversampled a/d conversion and error-rate dependence of nonbandlimited signals with finite rate of innovation," *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2140–2154, 2006.

[29] A. Gersho, "Principles of quantization," *IEEE Transactions on circuits and systems*, vol. 25, no. 7, pp. 427–436, 1978.

[30] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[31] Ó. Fontenla-Romero, B. Guijarro-Berdiñas, D. Martinez-Rego, B. Pérez-Sánchez, and D. Peteiro-Barral, "Online machine learning," in *Efficiency and Scalability Methods for Computational Intellect*, pp. 27–54, IGI global, 2013.

[32] P. Craven and G. Wahba, "Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation," *Numerische mathematik*, vol. 31, no. 4, pp. 377–403, 1978.

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

[34] C. Kelley, *Iterative Methods for Optimization*, vol. 18. SIAM, 1999.

[35] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[36] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, pp. 2004–2005, 2003.

[37] N. Parikh, S. Boyd, *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[38] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[39] V. Vapnik, "Principles of risk minimization for learning theory," *Advances in neural information processing systems*, vol. 4, 1991.

[40] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*, vol. 375. Springer Science & Business Media, 1996.

[41] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International conference on computational learning theory*, pp. 416–426, Springer, 2001.

[42] J. Mercer, "Xvi. functions of positive and negative type, and their connection the theory of integral equations," *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, vol. 209, no. 441-458, pp. 415–446, 1909.

[43] J. H. Manton, P.-O. Amblard, *et al.*, "A primer on reproducing kernel hilbert spaces," *Foundations and Trends® in Signal Processing*, vol. 8, no. 1–2, pp. 1–126, 2015.

[44] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[45] D. Bertsekas, *Dynamic programming and optimal control: Volume I*, vol. 1. Athena scientific, 2012.

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

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[48] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[49] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, Pmlr, 2013.

[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[51] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, 2014.

[52] A. J. Jerri, "The shannon sampling theorem—its various extensions and applications: A tutorial review," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596, 1977.

[53] H. D. Luke, "The origins of the sampling theorem," *IEEE Communications Magazine*, vol. 37, no. 4, pp. 106–108, 1999.

[54] W. R. Bennett, "Spectra of quantized signals," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 446–472, 1948.

[55] R. M. Gray, "Quantization noise spectra," *IEEE Transactions on information theory*, vol. 36, no. 6, pp. 1220–1244, 1990.

[56] N. T. Thao and M. Vetterli, "Reduction of the mse in r-times oversampled a/d conversion o (1/r) to o (1/r/sup 2/)," *IEEE Transactions on Signal Processing*, vol. 42, no. 1, pp. 200–203, 1994.

[57] T.-T. Nguyen, *Deterministic analysis of oversampled A/D conversion and Sigma Delta modulation, and decoding improvements using consistent estimates*. PhD thesis, Columbia University, 1993.

[58] M. Vetterli, P. Marziliano, and T. Blu, "Sampling signals with finite rate of innovation," *IEEE transactions on Signal Processing*, vol. 50, no. 6, pp. 1417–1428, 2002.

[59] B. Beferull-Lozano and A. Ortega, "Efficient quantization for overcomplete expansions in/spl ropf//sup n," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 129–150, 2003.

[60] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[61] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine learning*, vol. 46, pp. 131–159, 2002.

[62] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *The Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[63] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

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

[65] H. H. Bauschke, R. Goebel, Y. Lucet, and X. Wang, "The proximal average: basic theory," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 766–785, 2008.

[66] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Artificial Intelligence and Statistics*, pp. 398–406, 2015.

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

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

[69] J. NILSSON, "Real-time control systems with delay," *PhD thesis, Lund Institute of Technology*, 1998.

[70] A. Gambier, "Real-time control systems: a tutorial," in *2004 5th Asian Control Conference (IEEE Cat. No. 04EX904)*, vol. 2, pp. 1024–1031, IEEE, 2004.

[71] C. Schmidt, C. Kottke, V. Jungnickel, and R. Freund, "High-speed digital-to-analog converter concepts," in *Next-Generation Optical Communication: Components, Sub-Systems, and Systems VI*, vol. 10130, pp. 133–141, SPIE, 2017.

[72] E. Waring, "Vii. problems concerning interpolations," *Philosophical transactions of the royal society of London*, no. 69, pp. 59–67, 1779.

[73] I. J. Schoenberg, "Contributions to the problem of approximation of equidistant data by analytic functions," in *IJ Schoenberg Selected Papers*, pp. 3–57, Springer, 1988.

[74] E. Meijering, "A chronology of interpolation: from ancient astronomy to modern signal and image processing," *Proceedings of the IEEE*, vol. 90, no. 3, pp. 319–342, 2002.

[75] L. Schumaker, *Spline functions: basic theory*. Cambridge University Press, 2007.

[76] M. L. Littman, *Algorithms for sequential decision-making*. Brown University, 1996.

[77] A. Blum and Y. Monsour, "Learning, regret minimization, and equilibria," *Algorithmic Game Theory*, 2007.

[78] A. Nosedal-Sanchez, C. B. Storlie, T. C. Lee, and R. Christensen, "Reproducing kernel hilbert spaces for penalized regression: A tutorial," *The American Statistician*, vol. 66, no. 1, pp. 50–60, 2012.

[79] G. Chiarot and C. Silvestri, "Time series compression survey," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–32, 2023.

[80] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *Journal of mathematical analysis and applications*, vol. 33, no. 1, pp. 82–95, 1971.

[81] A. Koppel, A. S. Bedi, K. Rajawat, and B. M. Sadler, "Optimally compressed nonparametric online learning: Tradeoffs between memory and consistency," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 61–70, 2020.

[82] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3103–3131, 2012.

[83] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," in *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6, IEEE, 2012.

[84] S. Van Vaerenbergh, J. Via, and I. Santamaría, "A sliding-window kernel rls algorithm and its application to nonlinear channel identification," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, pp. V–V, IEEE, 2006.

[85] A. S. Bedi, A. Koppel, K. Rajawat, and B. M. Sadler, "Trading dynamic regret for model complexity in nonstationary nonparametric optimization," in *2020 American Control Conference (ACC)*, pp. 321–326, IEEE, 2020.

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

[87] R. Dixit, A. S. Bedi, R. Tripathi, and K. Rajawat, "Online learning with inexact proximal online gradient descent algorithms," *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1338–1352, 2019.

[88] Y.-L. Yu, "Better approximation and faster algorithm using the proximal average," in *Advances in neural information processing systems*, pp. 458–466, 2013.

[89] A. Beck, *First-order methods in optimization*, vol. 25. SIAM, 2017.

[90] F. Pérez-Cruz and O. Bousquet, "Kernel methods and their potential use in signal processing," *IEEE signal processing magazine*, vol. 21, no. 3, pp. 57–65, 2004.

[91] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on image processing*, vol. 16, no. 2, pp. 349–366, 2007.

[92] A. Pozdnoukhov, *Prior Knowledge in Kernel Methods*. PhD thesis, Verlag nicht ermittelbar, 2006.

[93] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society: Series B (Methodological)*, vol. 36, no. 2, pp. 111–133, 1974.

[94] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[95] F. Orabona and J. Luo, "Ultra-fast optimization algorithm for sparse multi kernel learning," in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[96] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[97] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on signal processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[98] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2008.

[99] J. B. Souza Filho and P. S. Diniz, "A fixed-point online kernel principal component extraction algorithm," *IEEE Transactions on Signal Processing*, vol. 65, no. 23, pp. 6244–6259, 2017.

[100] J. B. Souza Filho, L.-D. Van, T.-P. Jung, P. S. Diniz, *et al.*, "Online component analysis, architectures and applications," *Foundations and Trends® in Signal Processing*, vol. 16, no. 3-4, pp. 224–429, 2022.

[101] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[102] R. Money, J. Krishnan, and B. Beferull-Lozano, "Sparse online learning with kernels using random features for estimating nonlinear dynamic graphs," *IEEE Transactions on Signal Processing*, 2023.

[103] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a fixed budget," *Advances in neural information processing systems*, vol. 18, 2005.

[104] D. Sahoo, S. C. Hoi, and B. Li, "Online multiple kernel regression," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 293–302, 2014.

[105] Y. Shen, T. Chen, and G. Giannakis, "Online ensemble multi-kernel learning adaptive to non-stationary and adversarial environments," in *International Conference on Artificial Intelligence and Statistics*, pp. 2037–2046, PMLR, 2018.

[106] D. Sahoo, S. C. Hoi, and B. Li, "Large scale online multiple kernel regression with application to time-series prediction," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 1, pp. 1–33, 2019.

[107] U. Marteau-Ferey, D. Ostrovskii, F. Bach, and A. Rudi, "Beyond least-squares: Fast rates for regularized empirical risk minimization through self-concordance," in *Conference on learning theory*, pp. 2294–2340, PMLR, 2019.

[108] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[109] C. A. Micchelli, M. Pontil, and P. Bartlett, "Learning the kernel function via regularization.," *Journal of machine learning research*, vol. 6, no. 7, 2005.

[110] Y. Shen, T. Chen, and G. B. Giannakis, "Random feature-based online multi-kernel learning in environments with unknown dynamics," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 773–808, 2019.

[111] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Mathematical methods of operations research*, vol. 66, no. 3, pp. 373–407, 2007.

[112] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

[113] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[114] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[115] S. Shalev-Shwartz and Y. Singer, "Efficient learning of label ranking by soft projections onto polyhedra," *Journal of Machine Learning Research*, 2006.

[116] W. Wang and M. A. Carreira-Perpinán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," *arXiv preprint arXiv:1309.1541*, 2013.

[117] M. Blondel, A. Fujino, and N. Ueda, "Large-scale multiclass support vector machine training via euclidean projection onto the simplex," in *2014 22nd International Conference on Pattern Recognition*, pp. 1289–1294, IEEE, 2014.

[118] A. D. Mishra and D. Garg, "Selection of best sorting algorithm," *International Journal of intelligent information Processing*, vol. 2, no. 2, pp. 363–368, 2008.

[119] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l 1-ball for learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning*, pp. 272–279, 2008.

[120] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[121] A. Uncini, *Fundamentals of adaptive signal processing*. Springer, 2015.

[122] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[123] O. Kosheleva and V. Kreinovich, "Why physical processes are smooth or almost smooth: A possible physical explanation based on intuitive ideas behind energy conservation," *Mathematical Structures and Modeling*, 2021.

[124] H. S. Wilf, *Algorithms and complexity*. AK Peters/CRC Press, 2002.

[125] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*, pp. 63–71, Springer, 2004.

[126] Q. Lu, G. V. Karanikolas, and G. B. Giannakis, "Incremental ensemble gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[127] M. Unser and T. Blu, "Generalized smoothing splines and the optimal discretization of the wiener filter," *IEEE Transactions on Signal Processing*, vol. 53, no. 6, pp. 2146–2159, 2005.

[128] K. Frankish and W. M. Ramsey, *The Cambridge handbook of artificial intelligence*. Cambridge University Press, 2014.

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

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

[131] C. H. Reinsch, "Smoothing by spline functions," *Numerische mathematik*, vol. 10, no. 3, pp. 177–183, 1967.

[132] 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," *International Journal of Machine Tools and Manufacture*, vol. 96, pp. 27–46, 2015.

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

[134] A. M. Schäfer, *Reinforcement learning with recurrent neural networks*. PhD thesis, Osnabrück, Univ., Diss., 2008, 2008.

[135] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural network policies with continuous memory states," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 520–527, IEEE, 2016.

[136] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *NIPS 2017 Autodiff Workshop*, 2017.

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

[138] OSIsoft, "Pi system," *URL: https://www. osisoft. com/pi-system*, 1980.

[139] A. Hebrail, Georges & Berard, "Individual household electric power consumption." UCI Machine Learning Repository, 2012.

[140] 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, p. 4342, Oct. 2019.

[141] J. Huan, J. S. Bernstein, P. Difuntorum, N. V. R. Masna, N. Gravenstein, S. Bhunia, and S. Mandal, "A wearable skin temperature monitoring system for early detection of infections," *IEEE Sensors Journal*, vol. 22, no. 2, pp. 1670–1679, 2022.

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

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

[144] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[145] S. Rangan and V. K. Goyal, "Recursive consistent estimation with bounded noise," *IEEE Transactions on Information Theory*, vol. 47, no. 1, pp. 457–464, 2001.

[146] V. K. Goyal, J. Kovačević, and J. A. Kelner, "Quantized frame expansions with erasures," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 203–233, 2001.

[147] J. Kovacevic, V. K. Goyal, and M. Vetterli, "Fourier and wavelet signal processing," *Fourier Wavelets. org*, pp. 1–294, 2013.

[148] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *arXiv preprint arXiv:2305.01072*, 2023.

[149] I. J. Schoenberg, "On interpolation by spline functions and its minimal properties," in *On Approximation Theory/Über Approximationstheorie*, pp. 109–129, Springer, 1964.

[150] P. J. Green and B. W. Silverman, *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.

[151] B. Widrow and I. Kollár, *Quantization noise: roundoff error in digital computation, signal processing, control, and communications*. Cambridge University Press, 2008.