Automatica 180 (2025) 112469

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief Paper

On reconstructing high derivatives of noisy time-series with confidence intervals☆

Mazen Alamir

Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France

ARTICLE INFO

Article history:
Received 24 September 2024
Received in revised form 18 February 2025
Accepted 3 June 2025
Available online 30 June 2025

Keywords:
High derivatives reconstruction
Noisy time-series
Confidence interval
Cross-validation
Auto-tuned algorithms

ABSTRACT

Reconstructing high derivatives of noisy measurements is an important step in many control, identification and diagnosis problems. In this paper, a heuristic is proposed to address this challenging issue. The framework is based on a dictionary of identified models indexed by the bandwidth, the noise level and the required degrees of derivation. Each model in the dictionary is identified via cross-validation using tailored learning data. It is also shown that the proposed approach provides heuristically defined confidence intervals on the resulting estimation. The performance of the framework is compared to the state-of-the-art available algorithms showing noticeably higher accuracy. Although the results are shown for up to the 4-th derivative, higher derivation orders can be used with comparable results. An associated `python` module is made available via: `pip install ML_derivatives`.

1. Introduction

Reconstructing high derivatives of measurements is a key issue in some relevant dynamic systems-related topics such as system identification (Ljung, 2010), state/parameter estimation (Evensen, Vossepoel, & Van Leeuwen, 2022) and anomaly detection to cite but few examples. When dealing with this important issue, two kinds of paradigms should be formally distinguished, namely: (1) the filtering paradigm where a real-time updating of the estimated derivatives is operated, generally via simple difference equations implementing short memory filters and (2) the derivatives reconstruction paradigm that can be done off-line and might involve a more holistic view of the past measurements at the price of extra-computational cost.

Indeed, in the first case, new measurements are coming in a real-time stream and a processing algorithm has to take them into account *on the fly* in some updating iteration such as in Kalman Filtering (KF) (Khodarahmi & Maihami, 2023) so that the result can feed a control algorithm for instance. The excellent survey (Mojallizadeh, Brogliato, & Acary, 2021) provides a recent overview of the state of the art on real-time differentiation algorithms including sliding modes (Levant & Yu, 2018), Kalman Filter and some other few alternatives.

In the second case, one disposes of collected measurements, over some observation window, and can process them in order to deliver reconstructed profiles of dth derivatives. In this context, the reconstructed derivatives might serve in an off-line task such as nonlinear continuous-time identification (Brunton & Kutz, 2022) and or characterization of normality space (Bergweiler, 2001; de Souza & Ferreira, 2012). Spectral Derivation (SD) (Davies, 1995) is an example of such a solution since the computation of the Fourier transform needs a large measurement window to be processed. Obviously, there is no neat separation between the above mentioned classes of methods since window-based reconstruction process can be eligible for on-line real-time implementation if the characteristic time of the systems is not too small.

The starting point of this contribution lies in the recent results reported in Mojallizadeh et al. (2021) regarding the *filtering*-like algorithms. These results suggest that, relying on such filtering algorithms to reconstruct higher derivatives (derivation orders ≥ 2) is not realistic even for quite low noise ($< 3\%$). On the other hand, the emergence of `Machine Learning` culture where learning data-sets are collected and then processed off-line in order to fit various models and to gain some measurement-enforced understanding of hidden relationships triggered an interest in the second paradigm mentioned above. The framework proposed in this paper is the consequence of these two facts.

In a nutshell, the framework is based on a dictionary of identified models indexed by the bandwidth, the noise level and the required degrees of derivation. Each model in the dictionary is

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Andrea Garulli under the direction of Editor Alessandro Chiuso.

E-mail address: mazen.alamir@grenoble-inp.fr.

https://doi.org/10.1016/j.automatica.2025.112469

identified via cross-validation using tailored learning data. In the exploitation mode, the bandwidth and the noise-level are identified from the specific noisy time-series before the derivatives are reconstructed using the specific element in the dictionary of models. The effectiveness of the algorithm comes from the fact that several estimations are obtained for each sampling instant thanks to a moving-window process. This adds to the possibility of getting high reconstruction precision the ability to provide confidence intervals on the resulting estimation.

It is important to underline that, as explained above, the proposed heuristic is not intended to replace the filtering approaches in all situations. Rather, it can be preferred due to its higher precision in situations where real-time computation is not the key challenge. This might be because the dynamic system is quite slow or when off-line use is targeted for a continuous-time identification or modeling tasks to cite but two examples. The algorithm is implemented in a publicly available `python` module via: `pip install ML_derivatives`.

This paper is organized as follows. First of all, Section 2 clearly states the problem to be addressed. Section 3 introduces some definitions and notation used throughout the paper. The proposed high-order derivatives reconstruction framework is explained in Section 4. Finally Section 5 proposes some numerical investigations in order to assess the relevance of the proposed framework and to compare it to some other alternatives.

2. Problem statement

The objective of the paper is to provide an algorithm that takes as argument a triplet $(\mathbf{s}, d, \tau)$ where:

- ✓ $\mathbf{s}$ is a sequence $\mathbf{s} := (s_1, \dots, s_n)$ containing n successive uniformly distributed measurement instances;
- ✓ $d \in \mathbb{N}$ is the order of derivation;
- ✓ $\tau > 0$ is the sampling measurement acquisition period;

and delivers as output a sequence $\hat{\mathbf{s}}^{[d]} \in \mathbb{R}^n$ representing an estimation of the d-derivative of the original time-series $\mathbf{s}$ over the same window of length n. Moreover, the algorithm should deliver an associated standard deviation profile $\hat{\sigma}^{[d]} \in \mathbb{R}^n$ that enables to reconstruct a confidence tube (commonly defined using $\pm 3\hat{\sigma}^{[d]}$ around $\hat{\mathbf{s}}^{[d]}$). This targeted map can be denoted by:

$$(\hat{\mathbf{s}}^{[d]}, \hat{\sigma}^{[d]}) \leftarrow \texttt{est\_deriv}(\mathbf{s}, d, \tau). \tag{1}$$

A weaker version might require the caller of the function to provide the standard deviation of the measurement noise, referred to hereafter as the `noise_level`, namely:

$$(\hat{\mathbf{s}}^{[d]}, \hat{\sigma}^{[d]}) \leftarrow \texttt{est\_deriv}(\mathbf{s}, d, \tau, \texttt{noise\_level}). \tag{2}$$

The following section gives some definitions and notation that are extensively used in the presentation of the proposed computational framework.

Notice that since there are as many sampling acquisition periods as there are real-life use-cases, the following straightforward identity is worth recalling:

$$\texttt{est\_deriv}(\mathbf{s}, d, \tau) \equiv \frac{1}{\tau^d} \times \texttt{est\_deriv}(\mathbf{s}, d, 1) \tag{3}$$

which comes from the possibility of scaling the time so that the resulting scaled sampling period becomes equal to 1. This simply means that one can concentrate on the normalized case where $\tau = 1$, build the associated map and only when the concrete final estimation is operated using the designed map, one can introduce the sampling period-related correction given by (3). This is of a tremendous importance since the learning step can be done *universally* once for all independently of the effective acquisition period used in the application instances. Consequently, in the sequel, the normalized case $\tau = 1$ is considered dropping here and there the word *normalized* for the sake of brevity when no ambiguity is possible.

3. Definitions and notation

In the sequel, the following notation is used: Given a matrix M, the notation $M[i_0 : i_1, j_0 : j_1]$ denotes the sub-matrix containing the lines indexed i_0 to i_1 and the columns indexed from j_0 to j_1. When the initial index is absent, this means that all the initial indexes are considered up to i_1 or j_1. The set of n uniformly distributed values between two bounds v_1 and v_2 is denoted by $\texttt{linspace}(v_1, v_2, n)$. Similarly, $\texttt{logspace}(v_1, v_2, n)$ denotes the logarithmic version, namely the set defined by $\{10^\xi, \xi \in \texttt{linspace}(v_1, v_2, n)\}$. Finally, considering a time series $\mathbf{s} \in \mathbb{R}^n$, the notation $\mathbf{s}[i_1 : i_2]$ refer to the *slice* of $\mathbf{s}$ indexed by indices from i_1 to i_2:

$$\mathbf{s}[i_1 : i_2] := \begin{bmatrix} s_{i_1} & s_{i_1+1} & \dots & s_{i_2} \end{bmatrix} \tag{4}$$

Now assuming that `n_per_period` (=5 in the implementation) samples are considered to be necessary inside a period to reconstruct a sinusoidal signal, the following maximum normalized pulsation is defined:

$$\text{(Max normalized pulsation)} \qquad \bar{\omega} := \frac{2\pi}{\texttt{n\_per\_period}} \tag{5}$$

This enables to define a dense grid of pulsations $\bar{\Omega}_{\texttt{grid}} \in \mathbb{R}^{n_{\texttt{grid}}}$ ($n_{\texttt{grid}} = 200$ is used hereafter):

$$\bar{\Omega}_{\texttt{grid}} := \{\Omega_1, \dots, \Omega_{n_{\texttt{grid}}}\} := \texttt{logspace}(-3, 0, \ n_{\texttt{grid}}) \times \bar{\omega} \tag{6}$$

that are used to generate the learning data by randomly drawing a high number of samples representing each a different linear combination of sinusoidal signals with pulsations included in $\bar{\Omega}_{\texttt{grid}}$.

More precisely, given a sequence length n_t, one can construct the basis function matrices with columns inside the following set ($t \in \mathbb{R}^{n_t}$ is the time vector)

$$\Big\{\mathbf{1}\Big\} \cup \Big\{\sin(\omega t), \cos(\omega t)\Big\}_{\omega \in \bar{\Omega}_{\texttt{grid}}} \tag{7}$$

leading to a basis function matrix $\mathtt{B}$ such that:

$$\mathtt{B} \in \mathbb{R}^{n_t \times n_B} \quad \text{where} \quad n_B := (2n_{\texttt{grid}} + 1) \tag{8}$$

such that for each randomly sampled vector of coefficients $a \in \mathbb{R}^{n_B}$, a candidate *admissible* time-series $\mathtt{B}a \in \mathbb{R}^{n_t}$ is obtained.

Similarly, using the d derivative of the above defined columns, one defines the d-derivative basis functions, denoted hereafter by $\mathtt{B}_d$ so that for any $a \in \mathbb{R}^{n_B}$, the following holds true:

$$\Big[\mathtt{B}_d\Big]a \quad \text{is the } d\text{th derivative of } \Big[\mathtt{B}\Big]a \tag{9}$$

Notice that the n_B columns of $\mathtt{B}$ correspond to increasing pulsations $1, \Omega_1, \Omega_1, \Omega_2, \Omega_2, \dots$ that are inherited from the pulsations included in $\Omega_{\texttt{grid}}$ given by (7). Therefore giving a prescribed cutoff pulsation $\Omega \leq \bar{\omega}$, the sub-matrix obtained from $\mathtt{B}$ by keeping only the columns corresponding to pulsations that are lower than Ω is denoted hereafter by $\mathtt{B}[:, : \Omega]$ representing the (Bandwidth limited Basis):

$$\mathtt{B}[:, : \Omega] \quad \text{Only columns with pulsations} \leq \Omega \tag{10}$$

This sub-matrix can then be used to compute the projection of any time-series $\mathbf{s} \in \mathbb{R}^{n_t}$ on the sub-space of time-series of bandwidth lower than Ω according to:

$$\hat{\mathbf{s}}(\Omega) := \Pi(\Omega) \cdot \mathbf{s} \quad \text{where} \quad \Pi(\Omega) := \mathtt{B}[:, : \Omega]\mathtt{B}[:, : \Omega]^\dagger \tag{11}$$

More generally, since the length of the time-series to be processed is not necessary of length n_t, the same projection process

can be adapted to any time-series $\mathbf{s} \in \mathbb{R}^n$ of length $n \leq n_t$ by selecting the first n lines of the matrix B which are denoted hereafter by B[1:n,: Ω] leading to the associated projection matrix:

$$\Pi_n(\Omega) \coloneqq \mathrm{B}[1:\mathrm{n},:\Omega]\mathrm{B}[1:\mathrm{n},:\Omega]^{\dagger} \tag{12}$$

As mentioned earlier, it is a fact that the appropriate parameters of any differentiation algorithm heavily depend on the frequency content (the bandwidth) of the time-series on the considered observation window. This is the reason why a set of n_r models are identified hereafter for different n_r predefined design cutoff pulsations that belong to $\bar{\Omega}_{\mathtt{grid}}$. For obvious memory reasons, the number n_r of design pulsations is $\ll n_{\mathtt{grid}}$. More precisely, the following set is defined ($n_r = 21$ is used in the implementation):

$$\bar{\Omega}_{\mathtt{design}} \coloneqq \{\omega_1, \ldots, \omega_{n_r}\} \coloneqq \mathtt{linspace}(\Omega_1,\ \bar{\omega},\ n_r) \tag{13}$$

in which n_r values ranging from the minimal to the maximal values in $\bar{\Omega}_{\mathtt{grid}}$ are used.

So far, we have all that we need to describe the proposed algorithm. This is done in the following section.

4. The proposed derivation framework

The proposed framework is based on a sequence of ideas. In this section, each of these ideas is stated, then discussed before its algorithmic translation is given.

4.1. The main ideas and their implementation

In what follows, the notation $\mathbf{y} \in \mathbb{R}^{n_w}$ is used to denote time-series of a specific length n_w which is the dimension of the input space of the maps to be identified. These maps are then used to construct the d-derivatives of any time series $\mathbf{s}$ of length greater than n_w as shown later on. The first idea can be stated as follows:

IDEA 1: A DIFFERENTIATOR IS A LINEAR MAP THAT IS NOT UNIVERSAL

Given a time-series $\mathbf{y} \in \mathbb{R}^{n_w}$, the map $F^{(d)} : \mathbb{R}^{n_w} \to \mathbb{R}^{n_w}$ that estimates the d-th derivative of $\mathbf{y}$ is a linear map, namely

$$F^{(d)}(\mathbf{y}) = A\mathbf{y}.$$

The matrix A of this map depends on:

1. the bandwidth of $\mathbf{y}$ and
2. the `noise_level` (standard deviation) of the measurement noise.

Discussion. This idea comes from the fact that in the absence of noise, one can imagine a linear interpolation process of sufficiently high order over a functional basis of known derivatives which involves only linear operations in the input $\mathbf{y}$. In the presence of noise, the appropriate regularization that corresponds to a trade-off enabling tracking the signal without tracking the noise obviously depends on both the noise level and the dynamics of the system since both these characteristics affect the optimal tuning of the trade-off.

Implementation. Since the model depends on the bandwidth and the level of the measurement noise, let us consider the set $\bar{\Omega}_{\mathtt{design}}$ of Design Pulsations defined in (13) and indexed by the set of integers $j \in \{1, \ldots, n_r\}$ together with the following set of *normalized*[1] noise levels given by:

$$\mathcal{N} \coloneqq \left\{\nu_1, \ldots, \nu_q\right\} \subset [0.0, 0.01, \ldots, 0.25] \tag{14}$$

indexed by the integer $\ell \in \{1, \ldots, q = 21\}$. Now for each pair (ω_j, ν_ℓ) in the Cartesian product of the above cited sets, A set of `n_samples` **normalized** trajectories and their corresponding d-derivatives can be created:

$$X_{(j,\ell,d)} \coloneqq \left\{ \frac{\mathrm{B}_d[:\ \ n_w,:\ \ \omega_j]a^{[\kappa]}}{\|\mathrm{B}[:\ \ n_w,:\ \ \omega_j]a^{[\kappa]}\|_\infty} + \delta_{0d} \cdot \nu_\ell \cdot \mathbf{u}^{[\kappa]} \right\}_{\kappa \in \{1,\ldots,\mathtt{n\_samples}\}} \in \left[\mathbb{R}^{n_w}\right]^{\mathtt{n\_samples}} \tag{15}$$

where for each κ, $a^{[\kappa]}$ is a random vector of dimension n_B while $\mathbf{u}^{[\kappa]}$ is a random sequence of white noise of standard deviation=1. δ_{0d} is the Kronecker product that is equal to 0 for all $d \neq 0$ while $\delta_{00} = 1$.

This means that only the 0-derivative version is made noisy by adding the noise with the level ν_ℓ to the normalized randomly generated time-series.

This is done because the 0-derivative version is the argument of the maps to be learned and that comes always corrupted by the measurement noise. By opposition, the set of d-derivatives are not corrupted with noise since they will serve as the ground truth labels for the model's fitting process.

The previous discussion can be summarized as follows:

The lines of the matrix $X_{(j,\ell,0)} \in \mathbb{R}^{\mathtt{n\_samples} \times n_w}$ defined by (15) represent a set of `n_samples` time-series, each of amplitude lower than 1 should the noise be 0 and $X_{(j,\ell,d)}$ are their corresponding *exact* d-derivatives time-series. For a given bandwidth index j and a noise level index ℓ, the base model for the reconstruction of the d-derivative for $d \geq 1$ can be learned using[a]:

- the noisy $X_{(j,\ell,0)}$ **as features** matrix
- the ground-truth noise-free $X_{(j,\ell,d)}$ **as labels**

As for the reconstruction of the 0-derivative (filtering mode), the feature matrix and the label matrix are given by $X_{(j,\ell,0)}$ and $X_{(j,0,0)}$ respectively.

[a] In Machine Learning terminology, the feature matrix is a matrix where each line is an instance of the input x to the function F we are looking for such that $F(x)$ equal the associated target, also called label, say y. Machine Learning algorithms try to approximately solve the set of equations: $F(x^{(i)}) \approx y_i$ for $i = 1, \ldots, n$ where $x^{(i)}$ is the i-th line of the features matrix having n lines.

So assume a features matrix $X \in \mathbb{R}^{\mathtt{n\_samples} \times n_w}$ and a label matrix $L \in \mathbb{R}^{\mathtt{n\_samples} \times n_w}$ (there is such a pair for each value of the triplet (j, ℓ, d)), the model we are looking for is a matrix $A \in \mathbb{R}^{n_w \times n_w}$ that solves a regularized least-squares problem of the form:

$$\min_A \|XA - L\|^2 + \alpha\|A\|^2 \tag{16}$$

for an appropriate choice of the regularization parameter $\alpha > 0$. The role of the regularization parameter α is to precisely achieve a trade-off between the need for capturing the relationship we are looking for and capturing the specific realization of the noise as invoked earlier.

[1] In the sense that the corresponding measurement noise is added to normalized time-series to get time-series of amplitudes lower than (1) to build the learning data.

It is precisely in order to find this *appropriate choice* for each triplet (j, ℓ, d) that the second idea is invoked:

IDEA 2: α IS TUNED VIA CROSS-VALIDATION

The fine tuning of the regularization parameter α involved in the optimization problem (16) can be automatically obtained using the cross-validation technique widely used in the Machine Learning algorithms.

Discussion. The cross-validation technique consists in splitting the available learning data into `cv` (=2 hereafter) subsets of data. The model is learned (for a given regularization parameter value α) using `cv-1` subsets leaving one subset for the evaluation of the extrapolation error on unseen data. This is repeated `cv` times and the statistics of the extrapolation error is computed for this specific α. The finally chosen α is the one that minimizes the so-computed extrapolation error on unseen data.

Implementation. For linear models as the ones we are seeking here, the Machine Learning libraries offer an already implemented fitting algorithms with a cross-validation-based autotuning of regularization parameters. In particular this is the case for the `scikit-learn` library (Pedregosa et al., 2011) through the `RidgeCV` (used hereafter) and the `LassoLarsCV` models to cite but two examples. A typical `python` call involving the features and label matrices X and L takes the following form:

```
from sklearn.linear_model import RidgeCV
alphas = np.logspace(-4,3,20)
reg = RidgeCV(cv=2, alphas=alphas,
       fit_intercept=False).fit(X, L)
```

where `alphas` is a list of candidate values for the regularization parameter α. The function `RidgeCV` performs the above described cross-validation based optimization and return the appropriate matrix A = `reg.coef_` as an attribute of the fitted model `reg`.

Notice that for the sake of simplification, we omitted the reference to the triplet of indexes (j, ℓ, d) that underlines each of the associated solution A which results, each, from its own pair of features and label matrices as explained above.

As a matter of fact, upon exploring all possible values of the triplet (j, ℓ, d), a dictionary of models (matrices) is obtained such that:

$$\left\{A_{(j,\ell,d)} \in \mathbb{R}^{n_w \times n_w}\right\}_{(j,\ell,d)\in\{1,\dots,n_r\}\times\{1,\dots,q\}\times\{0,1,\dots,d_{max}\}} \tag{17}$$

Recall that j denotes the index of the maximum pulsation $\omega_j \in \bar{\Omega}_{\texttt{design}}$ [see (13)] contained in the time-series while ℓ denotes the noise level index associated to the standard deviation ν_ℓ defined in (14). We shall later explain how these appropriate indices are computed for a given time-series but let us before explain the main averaging idea that is in the heart of the precision improvement provided by the framework.

In order to explain the idea let us assume that the triplet (j, ℓ, d) is available and hence so is its associated model summarized by the matrix $A_{(j,\ell,d)} \in \mathbb{R}^{n_w \times n_w}$ such that for any time series $\mathbf{y} \in \mathbb{R}^{n_w}$ of bandwidth ω_j and noise level ν_ℓ, the estimation of its d-derivative time-series is given by:

$$\hat{\mathbf{y}}^{(d)} := \left[A_{(j,\ell,d)}\right]^T \cdot \mathbf{y} \tag{18}$$

let us consider a **longer time series** $\mathbf{s} \in \mathbb{R}^n$ with $n \gg n_w$. Notice that each slice of $\mathbf{s}$ of length n_w, namely $\mathbf{s}[i : i + n_w - 1]$ for some $i \leq n - n_w + 1$ can be viewed as a time-series $\mathbf{y}_{[i]}(\mathbf{s}) \in \mathbb{R}^{n_w}$ for which one can use the models $A_{(j,\ell,d)}$ to reconstruct the d-derivative via:

$$E^{(j,\ell,d)}(\mathbf{y}_{[i]}(\mathbf{s})) =: \begin{bmatrix} E_0^{(j,\ell,d)}(\mathbf{y}_{[i]}(\mathbf{s})) \\ \vdots \\ E_{n_w-1}^{(j,\ell,d)}(\mathbf{y}_{[i]}(\mathbf{s})) \end{bmatrix} := \left[A_{(j,\ell,d)}\right]^T \cdot \underbrace{\mathbf{s}[i : i + n_w - 1]}_{\mathbf{y}_{[i]}(\mathbf{s})} \tag{19}$$

where j and ℓ correspond to the bandwidth and the noise level of the time-series $\mathbf{s}$. Before we state the next averaging idea, let us summarize the last *rather complicated notation* as follows:

Given a time-series $\mathbf{s} \in \mathbb{R}^n$ where $n \geq n_w$ that corresponds to a bandwidth ω_j and noise-level close to ν_ℓ, the term $E_k^{(j,\ell,d)}(\mathbf{y}_{[i]}(s))$ defined by (19) provides an estimation of the d-derivative of $\mathbf{s}$ at instant $i + k$.
Consequently, given any $m \in \{1, \dots, n\}$, all pairs in the set of pairs defined by

$$\mathcal{I}_m^{(n,n_w)} := \left\{(i, k) \in \{1, \dots, n\} \times \{0, \dots, n_w - 1\} \mid i+k = m\right\}$$

provide as many **eligible** estimations of the d-derivative of $\mathbf{s}$ AT THE SAME INSTANT m as there are elements in $I_m^{(n,n_w)}$, namely $\texttt{card}(I_m^{(n,n_w)})$.

But it is easy to figure out that when m spans the set of indices of the time series $\mathbf{s}$, namely $\{1, \dots, n\}$, the number of estimations that can be gathered is given by:

$$\texttt{card}(I_m^{(n,n_w)}) := \begin{cases} m & \text{if } m < n_w \\ n_w & \text{if } n_w \leq m \leq n - n_w \\ n - m & \text{if } m > n - n_w \end{cases} \tag{20}$$

This means that when $n \gg n_w$, n_w estimations are obtained for the majority of time instants. This leads to the statement of the next idea:

IDEA 3: REDUCING NOISE IMPACT VIA MOVING WINDOW AVERAGING

By using high values of n_w and by considering long time-series $\mathbf{s}$, it is possible to get two benefits, namely:

1. Reducing the impact of noise by averaging multiple estimations of the d-th derivative for each single instant;
2. Getting confidence intervals of the reconstruction by measuring the dispersion of the different estimated values at each single instant.

Discussion. There are obviously some limitations on the use of larger values for n and n_w. Indeed, regarding n_w, one should keep in mind that for each triplet of values (j, ℓ, d) the base model involves a model of size n_w^2 (the number of elements in the matrix $A_{(j,\ell,d)}$) and all these models should be stored for possible use. Notice however that the memory can be drastically reduced using different possible compression techniques including the Singular Value Decomposition (SVD) technique that is used in the model explored later in this paper. In other words, all the results shown later are based on compressed models in order to check that the performances are those of the compressed models that should be ultimately used to get a light differentiation portable package. As for the limitation of the length n of the analyzed time-series, it stems from the risk of having *non differentiable incidents* that might impact the quality of the estimation of the bandwidth of

the signal. Moreover, even in the absence of such incidents, the associated risk is to consider high bandwidth pulsation ω_j that lasts only over a small portion of the window reducing the quality of the estimation on low frequency parts of the window.

In all the presented results, the implementation uses $n_w = 50$. The higher values of $n_w = 100, 200, 400$ have been tested showing slightly higher precision scores but the increment does not necessarily justify the impact on the memory footprint of the resulting dictionary of maps. Using $n_w = 50$, the resulting size of the cumulative memory of all the compressed models for all possible triplets put together is around 15Mb.

Implementation. Based on the above discussion, given a time-series $\mathbf{s} \in \mathbb{R}^n$, the estimation of the d-derivative at instant m is given by:

$$\hat{\mathbf{s}}_m^{[d]} := \frac{1}{\texttt{card}(I_m^{(n,n_w)})} \sum_{(i,k)\in I_m^{(n,n_w)}} E_k^{(j,\ell,d)}(\mathbf{s}[i:i+n_w-1]) \tag{21}$$

or in matrix form, denoting by $M[k,:]$ the kth line of a matrix M:

$$\hat{\mathbf{s}}_m^{[d]} := \frac{1}{\texttt{card}(I_m^{(n,n_w)})} \sum_{(i,k)\in I_m^{(n,n_w)}} A_{(j,\ell,d)}^T[k,:] \cdot \mathbf{s}[i:i+n_w-1] \tag{22}$$

Similarly, the standard deviation of the estimation can also be obtained according to:

$$\hat{\boldsymbol{\sigma}}_m^{[d]} := \left[\frac{1}{\texttt{card}(I_m^{(n,n_w)})} \sum_{(i,k)\in I_m^{(n,n_w)}} \left| E_k^{(j,\ell,d)}(\mathbf{s}[i:i+n_w-1]) - \hat{\mathbf{s}}_m^{[d]} \right|^2 \right]^{\frac{1}{2}} \tag{23}$$

which can also be put in matrix form.

This *almost* achieves the target as stated in Section 2 through (1). *Almost* because we considered that the bandwidth index j and the noise level index ℓ are available so that the appropriate model's matrix $A_{(j,\ell,d)}$ can be selected and used. The ways these indices are determined for a time-series $\mathbf{s}$ are successively described in the remainder of this section.

4.2. Bandwidth index selection

Recall that the appropriate index j we are looking for is the minimum index $j \in \{1,\dots,n_r\}$ such that the time-series $\mathbf{s}$ under scrutiny contains no pulsations that are greater than $\omega_j \in \bar{\Omega}_{\texttt{design}}$. The selection is based on the behavior of the error between the noisy signal $\mathbf{s}$ and its projection on the sub-space generated by the columns of the ω_j-truncated matrix defined by (10)–(12) when j increases from 1 to n_r. The error is computed using the projection matrix $\Pi_n(\omega_j)$ defined by (12) according to:

$$e_j := \left\| (\Pi_n(\omega_j) - \mathbb{I}_n) \cdot \mathbf{s} \right\| \tag{24}$$

The appropriate value $j^\star$ is obtained when the decrease of the error e_j becomes negligible meaning that adding higher pulsations (more columns) does not improve the approximation. More precisely the following selection rule is used:

$$j^\star := \inf\left\{ j \in \{1,\dots,n_r\} \quad | \quad (e_j - e_{n_r}) \le \texttt{th} \times (e_1 - e_{n_r}) \right\} \tag{25}$$

where $\texttt{th} > 0$ is a small threshold ($\texttt{th}$=0.1 is used in the implementation).

4.3. Estimating the noise level

Despite the fact that the level of the noise affecting the measured signal can be roughly guessed from a simple inspection of a given signal, the need for a systematic blind solution stems from the fact that a visual inspection step might not be allowed. Moreover, the level of noise can vary along large datasets that can span months if not years of data collection on the other hand. During such long periods, the sensors might be changed and/or deteriorated over time. The conclusion is that some systematic algorithmic correction should be considered. The way this can be done is explained in this section.

In order to understand the proposed solution, an important fact should be first stated: The precise knowledge of the noise level lead to a second order improvement of the quality of the reconstructed derivatives. This means that even if a model with *erroneous* noise level is used, the quality of the estimation of the 0-derivative of the signal (filtered version of the original signal) is still sufficiently good to get a much better estimation of the truly involved noise level. Once this estimation is available, a second call of the appropriate model can be done to get even better result.

This leads to the solution involving the following steps:

1. First determine the bandwidth index $j^\star$ as explained in the previous section [see (25)].
2. If available, use the user-provided value of the noise level. If no such a knowledge is available, use ℓ such that $\nu_\ell = 0.05$.
3. Compute the filtered version of the time-series $\mathbf{s}$ using the $(j^\star, \ell, 0)$ model, namely:
$$\mathbf{s}_f := \left[A_{(j^\star,\ell,0)} \right]^T \cdot \mathbf{s} \tag{26}$$
4. Estimate the standard deviation of the noise according to:
$$\sigma^\star := \texttt{std}\left[\mathbf{s} - \mathbf{s}_f \right] \tag{27}$$
5. Find the noise level index $\ell^\star$ such that:
$$\ell^\star := \texttt{arg}\min_{\ell=1}^{n_r} |\nu_\ell - \sigma^\star| \tag{28}$$
6. Use the model indexed by $(j^\star, \ell^\star, d)$ to compute the desired d-derivative of the time-series $\mathbf{s}$:
$$\hat{\mathbf{s}}^{[d]} := \left[A_{(j^\star,\ell^\star,d)} \right]^T \cdot \mathbf{s} \tag{29}$$

This ends the presentation of the proposed framework. It is now time to proceed to some numerical assessments and comparisons. This is the aim of the next section.

5. Assessment through numerical investigations

Before we show comparison with alternative solutions, let us first examine typical derivative reconstruction results that can be obtained using the proposed framework. This is shown in Fig. 1. In this figure, the yellow regions represent the 3σ-confidence intervals computed according to (23). The quality of the averaging-based reconstruction formulae (22) compared to the width of the confidence zone assesses the crucial role played by the moving window averaging in the quality of the derivative reconstruction.

The performance of the proposed algorithm is compared to the performance of some competing algorithms as described in the following section. For each algorithm, the tuning parameters are described and their ranges used in the forthcoming results are given. Regarding the settings of the parameters of the alternative

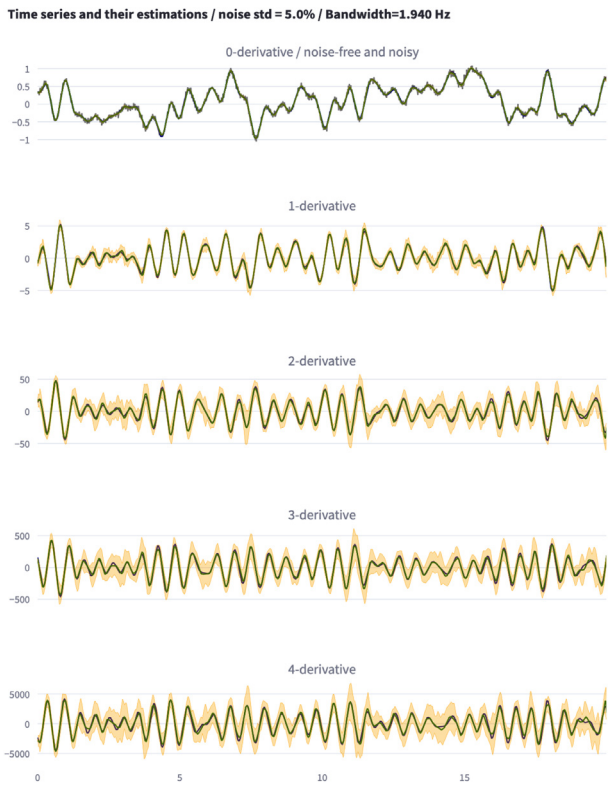

Fig. 1. Typical results of derivative reconstruction up to order 4 under noisy measurements with noise's standard deviation of 5%. Raw signal of bandwidth 1.94 Hz. Yellow bands show the 3σ-confidence intervals. Notice that both exact and estimated values are plotted. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

methods, the following approach is used which is extremely favorable to all the algorithms except the proposed one:

Tuning rule for alternative solutions

For each experiment and for each derivation order, the best tuning of the alternative methods (among the below defined set of tuning parameters values for each method) is chosen based on the ground truth of targeted derivatives. This leads to a favorable comparison for the alternative solutions since the ground truth is not supposed to be known and hence the associated optimal setting is impossible to know in real-life situations.

This leads, for the alternative algorithms, to a different ground-truth based setting, for each experiment and each derivative order to reconstruct inside the same experiment.

5.1. Alternative algorithms

The following derivatives estimation methodologies are examined in this section:

(1) **Kalman Filter**: This is a well known and widely used option that is based on building a state observer for the dynamic system given by:

$$\dot{x}_i = x_{i+1} \qquad i \in \{1, \dots, d_{max}\} \quad \dot{x}_{d_{max}+1} = 0 \tag{30}$$

when the estimation of the derivatives up to order d_{max} is required. The design of this filter needs the weighting matrices Q and R on the state and the measurement to be provided. To this end, the following parameterization is used:

$$R = 1 \quad ; \quad Q := \nu_s \times \mathtt{diag}\Big(\rho, \dots, \rho^{d+1}\Big) \tag{31}$$

where the two dimensional parameter $p_{\text{kalman}} := (\nu_s, \rho)$ takes possible tuning values in the set:

$$\mathbb{P}_{\text{kalman}} := \mathtt{logspace}(-21, 21, 25) \times \mathtt{logspace}(0, 8, 10) \tag{32}$$

which encompasses 250 different settings.

2) **Spectral derivation**. This algorithm is based on the property ($\mathcal{F}$ denoting the Fourier transform):

$$\mathcal{F}[y^{(d)}](s) = s^d \times \mathcal{F}[y](s) \tag{33}$$

which enables to estimate the d-derivative by applying the inverse Fourier transform to the multiplication, in the frequency domain, by s^d of the Fourier transform of the noisy original time series. The tuning parameter for this approach is the *smoothing* filter applied to the original signal before to process it as described above. In the implementation used hereafter, a Gaussian filter is used of the form $G(j\omega) := \exp(-\mu_f \cdot \omega^2)$ in which the smoothing parameter $p_{\text{spectral}} := \mu_f$ takes values in the tuning set given by:

$$\mathbb{P}_{\text{spectral}} := \mathtt{logspace}(-6, 0, 50) \tag{34}$$

leading to 50 different settings.

(3) The `scipy` **Savitzky–Golay** filter. This filter (Krishnan & Seelamantula, 2012) is based on an iterative window polynomial smoothing of the time series which result in two parameters: the window size n_w and the order of the polynomial r with the condition $r < n$. The following admissible sets are used for these two parameters:

$$\mathcal{N}_w := \Big\{1, 5, 11, 21, 41, 51, 101, 201, 401, 501\Big\}$$
$$\mathcal{R} := \{2, 3, 4, 5\}$$

leading to the following set of 40 possible settings of the parameter $p_{\text{savgol}} := (n_w, r)$:

$$\mathbb{P}_{\text{savgol}} := \mathcal{N}_w \times \mathcal{R} \tag{35}$$

(3) The Implicit **AO-STD filter**. This filter promoted in Mojallizadeh et al. (2021) implements the implicit version of a sliding mode filter which can provide any order derivatives. It takes the following form in which $z_{i,k}$ stands for the estimated ith derivative at instant k:

$$z_{n,k+1} = -h\lambda_n L \mathtt{sign}(\sigma_{0,k+1}) + z_{n,k} \tag{36}$$

$$z_{i,k+1} = -h\lambda_i L^{\frac{i+1}{n+1}} |\sigma_{0,k+1}|^{\frac{n-i}{n+1}} \mathtt{sign}(\sigma_{0,k+1}) + h z_{i+1,k+1} + z_{i,k} \tag{37}$$

which is an implicit equation in $\sigma_{0,k+1} := z_{0,k+1} - y_{k+1}$ representing the next estimation error on the original noisy signal y at the next instant $k+1$. The implementation uses exact implicit inversion via fixed-point iteration and the parameter λ_i suggested in Table 1 of Mojallizadeh et al. (2021). This leaves us with the hyper parameter L that is taken in the set defined by `logspace(-6,6,100)` which contains hundred possible settings the best of which is taken for every single profiles in the benchmark which is tremendously favorable to the competing solutions.

5.2. The benchmark

A set of validation time-series that are randomly generated using a set of pairs (ω, ν) of bandwidths and noise levels that

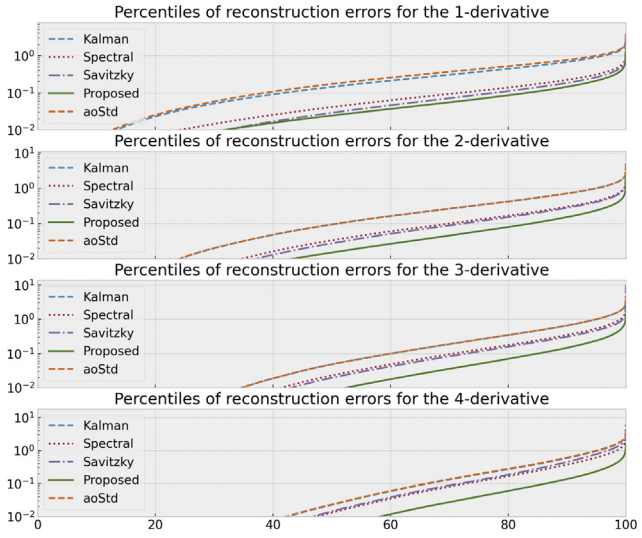

Fig. 2. Percentiles of reconstruction errors on the first four derivatives for the different algorithms (logarithmic scale). These statistics are computed over all the scenarios including the ones where a standard deviation of 10% is used for the noise. The total number of samples is equal to 192000. The y-scale is lower bounded by 10^{-2} for a better readability of the comparison. The best parameters is used based on the ground truth for each reconstruction of the alternative solutions while the proposed solution automatically determines its parameters.

belong to the Cartesian product of the sets:

$$\mathbb{W} := \Big\{0.01, 0.05, 0.1, 0.2, \ldots, 0.8, 0.9, 0.95\Big\} \tag{38a}$$

$$\mathbb{L} := \Big\{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.1\Big\} \tag{38b}$$

leading to a 96 time series of length 2000 each which corresponds to a total number of instants equal to 192000.

As for the performance evaluation, the following definition of the dth derivative's reconstruction error is used:

$$e := \frac{\texttt{percentile}\Big(|\hat{\mathbf{s}}^{(d)} - \mathbf{s}^{(d)}|, 95\Big)}{\texttt{percentile}(|\mathbf{s}^{(d)}|, 50)}, \tag{39}$$

in which $\hat{\mathbf{s}}^{(d)}$ and $\mathbf{s}^{(d)}$ stand for the reconstructed and the true d derivatives.

Fig. 2 and Fig. 3 show the comparison of the error as defined above over the whole validation dataset for the different derivative reconstruction algorithms described above. This figure suggests that the spectral and Savitzky–Golay options are very close to each other (as far as the adopted optimistic tuning rule is concerned) but they are both quite largely outperformed by the proposed algorithm and this is increasingly obvious as the degree of the derivation is increased.

In order to assess the relevance of the confidence intervals, Table 1 shows the ratio (over the 192,000 instants) of those where the reconstruction errors are lower than different thresholds: $\sigma/2$, σ, 2σ and 3σ.

Regarding the computation time, notice that the current implementation enables to compute the derivatives of a time-series of length 100 in less than 100 msec. This makes the approach eligible for real-time on-line implementation for a class of systems with comparable characteristic times.

	Proposed	noise_std=0.01			noise_std=0.05				noise_std=0.07			
	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4
50%	0.01	0.01	0. 00	0.00	0.03	0.02	0.01	0.00	0.04	0.02	0.01	0.01
80%	0.02	0.03	0.02	0.02	0.10	0.09	0.08	0.06	0.12	0.12	0.10	0.09
90%	0.04	0.04	0.04	0.04	0.15	0.17	0.16	0.16	0.19	0.20	0.19	0.19
	Savitzky	noise_std=0.01			noise_std=0.05				noise_std=0.07			
	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4
50%	0.01	0.01	0.01	0.01	0.04	0.03	0.02	0.01	0.04	0.04	0.03	0.02
80%	0.04	0.09	0.08	0.15	0.13	0.16	0.17	0.16	0.15	0.19	0.19	0.20
90%	0.07	0.17	0.17	0.40	0.20	0.29	0.32	0.44	0.23	0.33	0.34	0.48
	Spectral	noise_std=0.01			noise_std=0.05				noise_std=0.07			
	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4
50%	0.02	0.02	0.02	0.01	0.05	0.03	0.02	0.01	0.05	0.04	0.03	0.02
80%	0.07	0.1 0	0.12	0.12	0.15	0.18	0.17	0.16	0.18	0.2 0	0.2 0	0.19
90%	0.11	0.17	0.23	0.26	0.24	0.32	0.35	0.37	0.27	0.35	0.39	0.42
	AoStd	noise_std=0.01			noise_std=0.05				noise_std=0.07			
	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4	d=1	d=2	d=3	d=4
50%	0.17	0.09	0.05	0.02	0.17	0.08	0.04	0.02	0.18	0.1 0	0.05	0.03
80%	0.52	0.42	0.34	0.26	0.52	0.42	0.32	0.25	0.52	0.41	0.33	0.28
90%	0.78	0.69	0.64	0.56	0.78	0.71	0.63	0.57	0.77	0.70	0.65	0.63

Fig. 3. Evolution of the statistics of the error as the measurement noise increases for the different algorithms. The best parameters is used based on the ground truth for each reconstruction of the alternative solutions.

Table 1
Relevance of the confidence intervals: Ratios of the instants where the reconstruction error is within the thresholds $\sigma/2$, σ, 2σ or 3σ.

	$d=1$	$d=2$	$d=3$	$d=4$
$\sigma/2$	0.35	0.48	0.40	0.45
σ	0.61	0.73	0.65	0.70
2σ	0.87	0.92	0.87	0.91
3σ	0.95	0.97	0.95	0.97

6. Conclusion

In this paper a new **tuning-free** algorithm for the reconstruction of high derivatives of noisy time-series is proposed. The algorithm provides, in addition to providing precise derivatives reconstruction, a consistent confidence interval that can be used in the selection of the windows over which the results can be kept for later use in the identification and/or characterization of the normality among many possible tasks. The algorithm is implemented in a publicly available `python` module via: `pip install ML_derivatives` and a full description of use is provided at https://github.com/mazenalamir/ml_derivatives.

References

Bergweiler, W. (2001). Normality and exceptional values of derivatives. *Proceedings of the American Mathematical Society*, *129*(1), 121–129.

Brunton, S. L., & Kutz, J. N. (2022). Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press.

Davies, E. B. (1995). Spectral theory and differential operators, vol. 42. Cambridge University Press.

de Souza, J., & Ferreira, F. J. (2012). On the use of derivatives for interpreting magnetic anomalies due to dyke-like bodies: qualitative and quantitative analysis. In *Istanbul 2012-international geophysical conference and oil & gas exhibition* (pp. 1–4).

Evensen, G., Vossepoel, F. C., & Van Leeuwen, P. J. (2022). Data assimilation fundamentals: A unified formulation of the state and parameter estimation problem. Springer Nature.

Khodarahmi, M., & Maihami, V. (2023). A review on Kalman filter models. *Archives of Computational Methods in Engineering*, *30*(1), 727–747.

Krishnan, S. R., & Seelamantula, C. S. (2012). On the selection of optimum Savitzky-Golay filters. *IEEE Transactions on Signal Processing*, *61*(2), 380–391.

Levant, A., & Yu, X. (2018). Sliding-mode-based differentiation and filtering. *IEEE Transactions on Automatic Control*, *63*(9), 3061–3067.

Ljung, L. (2010). Perspectives on system identification. *Annual Reviews in Control*, *34*(1), 1–12.

Mojallizadeh, M. R., Brogliato, B., & Acary, V. (2021). Time-discretizations of differentiators: Design of implicit algorithms and comparative analysis. *International Journal of Robust and Nonlinear Control*, *31*(16), 7679–7723.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Mazen Alamir is research director at CNRS, France. He graduated in Mechanics (Grenoble, 1990) and Avionics (Toulouse 1992). He received his Ph.D. in Nonlinear Model Predictive Control in 1995 from Grenoble Institute of Technology. He served as head of the Nonlinear Systems and Complexity research group at the Control System Department, University of Grenoble-Alpes. His main research topics are Nonlinear model predictive control, Moving-horizon Estimation and Blind Fault Detection in industrial equipments via invariant-based normality characterization. He was a member of the IFAC technical committee on Nonlinear Systems as well as the IEEE Conference Editorial Board and served as Associate Editor of the IEEE Transaction on Automatic Control. He organized the first IFAC workshop on NMPC for Fast Systems.