On-line Generation of Trajectories Using Singular Spectrum Analysis: A Non-parametric Approach

Emerson Alves da Silva [*] **Leonardo Amaral Mozelli** [**]
Víctor Costa da Silva Campos [**]

[*] *Graduate Program in Electrical Engineering - Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG, Brazil*
[**] *Department of Electronics Engineering - UFMG*
(eas2011@ufmg.br, lamoz@ufmg.br, victor@cpdee.ufmg.br)

Abstract: Complex control applications, such as robotics and aerospace engineering, often require nonlinear control strategies. In the backstepping, feedback linearization, and sliding mode control, online trajectory generation is a major requirement in improving the controller performance, leading to high precision tracking. Its purpose is to generate continuous and bounded trajectories that are derivatives of a rough input reference signal, such as steps and pulses. In this context, this paper proposes the use of a non-parametric filter based on the real-time Singular Spectrum Analysis (SSA) method for online trajectory generation. The SSA is highly adaptive to the behavior of signals, including non-stationary ones, through its spectral decomposition. Additionally, is more selective than a simple Finite Impulse Response (FIR) filter, commonly used for generating trajectories. It can identify and extract components, smooth, and denoise a signal. Some experimental results show that SSA can be used as trajectory filter, by successfully generating bounded derivatives from discontinuous input signals. Moreover, empirical adjustment of the fixed filter parameters resulted in similar responses as those obtained for a parametric trajectory filter. These findings provide a potential mechanism for further researches regarding complex and non-stationary signals.

Keywords: Online Trajectory Generation; SSA Filter; Causal SSA Filter; Non-parametric Filtering; FIR Filter; Filter Bank; Bounded Derivatives.

1. INTRODUCTION

Motion control and industrial robot control problems usually require online computation of smooth and bounded trajectories and their derivatives from a rough input signal. They are also required in the design of nonlinear control laws such as feedback linearization, backstepping and Active Disturbance Rejection Control (ADRC) techniques. Additionally, abrupt changes in an input signal often lead to vibration, stress, and wear of mechanical components. Therefore, the smoothness of trajectory signals can impact the overall performance of control systems.

A common way to generate smooth and bounded trajectories is using a filter-based approach. In this approach, a rough input signal and its derivatives are smoothed by successive applying Finite Impulse Response (FIR) Moving Average (MA) filters. They are simple, efficient, and capable of complying with kinematic constraints imposed on the shape of the trajectories.

Another approach is to use data-driven methods. Those methods are called non-parametric filtering and differ from the parametric filter-based approach by not assuming a fixed distribution for the data, e.g., Gaussian. This way those methods can better deal with model nonlinearities, by adapting its output accordingly with the structure of the data.

Given this, this paper investigates the use of a real-time version of the non-parametric Singular Spectrum Analysis (SSA) filter, known as Causal Singular Spectrum Analysis (CSSA), as a novel online trajectory generating approach. The method is based on a cascade of CSSA filters that successively smooth an initially rough reference input signal such as step, pulse, and sawtooth, and its derivatives to generate smooth and bounded trajectories.

This paper is organized as follows. The trajectory filter approach and the SSA method are briefly reviewed in section 2. Section 3 details the cascaded FIR trajectory filter approach and the SSA filter, its interpretation through linear filtering as a Filter Bank (FB) of FIR filters and its real-time version, the CSSA. The experimental results on the proposed non-parametric SSA trajectory filter are presented in section 4. Finally, concluding remarks are given in section 5.

2. LITERATURE REVIEW

This section briefly describes traditional methods for computing trajectories in control applications, as well as the main aspects of the non-parametric SSA filter.

2.1 Trajectory Filter

Trajectory schemes aim at producing a set of signal profiles from a rough reference input by computing its derivatives.

Usually, the reference input represents the position, and the trajectory scheme generates bounded velocity, acceleration, jerk, snap, and so on, which also satisfy kinematic constraints (Biagiotti and Melchiorri, 2012). Gerelli and Bianco (2010) state that controlled robotics and mechatronics applications require smooth signals produced by trajectory generators to improve system performance.

Sudden changes in the control signal can lead to mechanical vibration, wear of mechanical components, induced noise, and disturbances (Lu, 2008). Zanasi et al. (2000); Kim et al. (1994); Zheng et al. (2009) emphasize that the availability of smooth reference trajectories with bounded derivatives is fundamental in getting accurate and quick tracking performance in complex motion control systems. Gerelli and Bianco (2010) stress that smooth reference signals are the most relevant requirement in the control of robotic applications. The bounds on the derivatives, i.e., minimum and maximum velocity and acceleration, are a direct consequence of physical constraints of actuators, such as voltage and current limits (Gerelli and Bianco, 2010; Zanasi et al., 2000).

Several techniques for trajectory generation are available in the literature. In industrial robotic and mechanical systems, smooth trajectories are usually obtained offline by computing optimal profiles (Zanasi et al., 2000; Zheng et al., 2009). Another offline approach is the selection of polynomial functions (Jeon and Ha, 2000; Zheng et al., 2009). These techniques are not suited for real-time applications due to their high computational cost (Zanasi et al., 2000; Jeon and Ha, 2000). According to Gerelli and Bianco (2010), an online alternative is to generate trajectories by nonlinear filtering through a feedback controller. Studies that explore this approach can be found in Zanasi et al. (2000), Zanasi and Morselli (2003), Lu (2008) and Gerelli and Bianco (2010).

Zanasi et al. (2000) proposed a nonlinear discrete-time trajectory filter based on a Variable Structured Controller (VSC), sliding mode control with a chain of integrators, capable of providing a minimum-time response without overshoot and symmetrical bounds on the first and second derivatives. Zanasi and Morselli (2003) presented a generalization of this method for higher-order derivatives and a broader class of input signals. In Lu (2008) a time-optimal jerk-constrained trajectory generator with disturbance rejection capabilities was proposed for speed control of electrical drives. Zheng et al. (2009) used a modified nonlinear tracking differentiator, robust to noise, to generate smooth trajectories from rough input signals such as steps. In Gerelli and Bianco (2010) a VSC, based on an algebraic nonlinear control law, in series with a chain of three integrators, was used to produce continuous trajectories for velocity and acceleration profiles.

Although trajectory filter generators with nonlinear controllers can be used in robotic applications successfully, they are rather complex and computational demanding (Biagiotti and Melchiorri, 2012). A simpler online alternative can be obtained through linear filtering by FIR filters (Biagiotti and Melchiorri, 2012; Besset and Béarée, 2017; Zheng et al., 2009). Studies that use this filter-based approach are Kim et al. (1994); Olabi et al. (2010); Biagiotti and Melchiorri (2012); Besset and Béarée (2017). In Kim et al. (1994), a velocity trajectory is generated through convolution with a digital filter for the control of industrial robots and Computer Numerical Control (CNC) machining tools, together with a PID controller and a notch filter for vibration reduction. Olabi et al. (2010) proposed a method of jerk-limited trajectory planning for continuous machining robots that consist of convolving a trapezoidal velocity-shaped signal with a FIR MA filter. Biagiotti and Melchiorri (2012) used a cascade of FIR MA filters, combined with the features of multi-segment polynomial trajectories, to generate time-optimal profiles with kinematic constraints on velocity, acceleration, and jerk. Finally, in Besset and Béarée (2017), a low computational complex acceleration-limited signal is convolved with a FIR MA filter to generate jerk-constrained trajectories for robot applications.

2.2 Singular Spectrum Analysis

The SSA is a technique for time series analysis and prediction that includes classic elements of series analysis, statistics, multivariate geometry, dynamic systems, and signal processing (Golyandina et al., 2001; Alexandrov, 2009). It decomposes the original time series in terms of its internal components, which are additive and interpretable, such as trend, periodical components, and noise (Golyandina and Zhigljavsky, 2013b).

This technique can be applied in trend analysis, detection, and extraction of quasi-periodic components, noise attenuation, and point change detection (Alexandrov, 2009). According to Hassani (2007), other applications include identification of trends and patterns with different resolutions and complexity, smoothing, extraction of seasonal components, cycles, and periodic signals with different amplitudes. Concerning distinct time-scales or non-stationary signals, Leles et al. (2018) proposed a new algorithm that introduces a moving subseries approach, addressing the problem of how to combine distinct decompositions into a single one.

Additionally, it does not require any statistical assumptions to be fulfilled for the time series or its components (Golyandina et al., 2001). It is not necessary also a parametric model that characterizes its statistical distribution, nor non-stationarity conditions (Zhigljavsky, 2010; Hassani, 2007). It is simple to use, requires few parameters to be adjusted, and allows trend extraction in the presence of noise and oscillating components (Alexandrov, 2009). Because of these advantages, Zhang and Wang (2016) adopted jointly the SSA and ADRC to resolve the complex problem of attitude regulation of a liquid-filled spacecraft model. The adaptive features of SSA show usefulness to separate relevant modes from an undesirable signal, without relying on much information from the system.

Despite being more selective and adaptive, there are some drawbacks in using SSA as a filter. The necessity to perform the Singular Value Decomposition (SVD) for each new sample increased the computational cost. This could be dealt with by the use of a recursive version of the SSA method. A discussion on the computational cost of the SSA method can be found in Leles et al. (2018). Since is a non-parametric filter, SSA requires tuning of its parameters to be done by empirical rules. This poses a challenge in

finding the right degree of smoothness and is an open problem in the signal processing field.

3. METHODS

This section describes the parametric trajectory filter based on a cascade of FIR filters and the non-parametric SSA method. In addition, it also addresses its interpretation as a bank of FIR filters as well as the CSSA, its adapted version for real-time processing.

3.1 Cascaded FIR Trajectory Filter

A trajectory filter based on a cascade of FIR filters, as described in Biagiotti and Melchiorri (2012), is depicted in Fig. 1. It generates a trajectory with the desired smoothness from a rough input signal such as steps and pulses. Successive filtering of the rough input signal f_0 and its derivatives by a MA filter H_i results in a bounded and smooth trajectory.

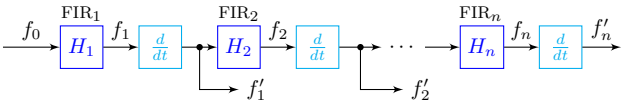

Figure 1. Trajectory generating filter based on a cascade of FIR filters.

The impulse response $h_i(t)$ and the i-th MA filter $H_i(s)$ are given in (1) and (2), for $i = 1, 2, \ldots, n$. It performs an averaging operation on the samples of the input signal from $t=0$ to $t=T_i$. An example of how this process works, for a scaled unit step input $f_0 = au(t)$, for $a>0$, is presented in Figs. 2 and 3.

$$h_i(t) = \begin{cases} \frac{1}{T_i}, & 0 \leqslant t < T \\ 0, & \text{otherwise} \end{cases} \quad (1) \qquad H_i(s) = \frac{1}{T_i}\frac{1-e^{-sT_i}}{s} \quad (2)$$

Assuming, for example, that $f_0(t)$ represents the position and the application requires the knowledge of the velocity trajectory from it. This information can be obtained simply by taking the derivative of $f_0(t)$, resulting in $f'_0(t) = a\delta(t)$. As shown in Fig. 2 (b), it is not a bounded signal, it has an area equals to a but infinite amplitude. Therefore can not be used for trajectory tracking.

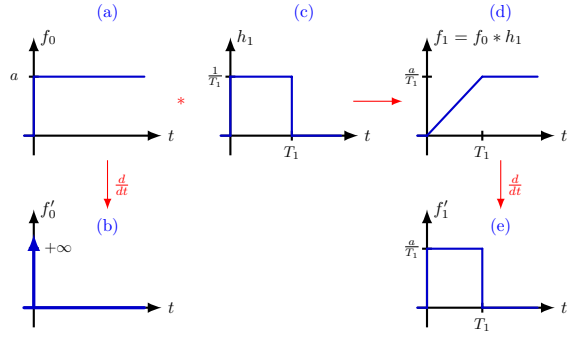

Figure 2. Effect of filtering a step input signal $f_0(t) = au(t)$ by a MA filter $h_1(t)$.

Instead, before taking the derivative, the input signal $f_0(t)$ is convolved with the MA filter $h_1(t)$, shown in Fig. 2 (c), resulting in a smoother version given by $f_1(t) = f_0(t) * h_1(t)$ in Fig. 2 (d), where $*$ stands for the convolution operation. Its derivative now, given in Fig. 2 (e), is a bounded signal and can be shaped according to the choice of the filter parameter T_1 and the amplitude a of the input.

Supposing that the acceleration profile is also required, to avoid getting an unbounded derivative again, such as $f''_1(t)$ in Fig. 3 (f), $f'_1(t)$ is convolved with another MA filter $h_2(t)$, shown in Fig. 3 (g), resulting in a signal $f_2(t) = f'_1(t) * h_2(t)$ (Fig. 3 (i)). Its derivative, $f'_2(t)$ in Fig. 3 (j), is now a bounded signal. The smoothing effect achieved by successive filtering of the original input signal can be seen by taking the integral of $f_2(t)$, as shown in Fig. 3 (h).

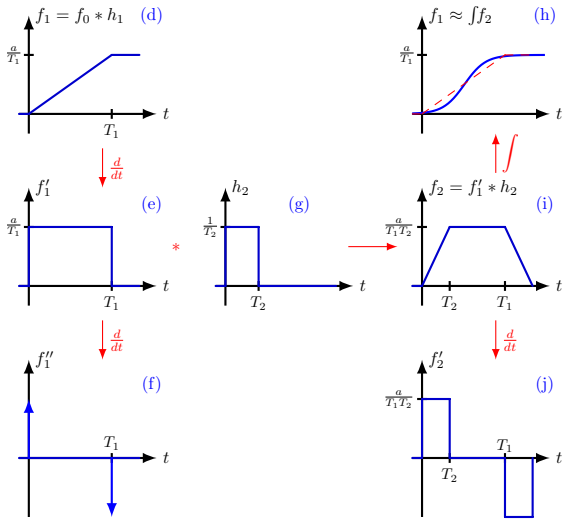

Figure 3. Further filtering f'_1 by another MA filter $h_2(t)$ to get a bounded derivative for f'_2.

This process repeats as higher-order derivatives are required. By the use of n FIR filters, as shown in Fig. 1, the first n–1 derivatives are continuous and the n-th derivative is a piece-wise constant signal.

3.2 SSA Algorithm

The non-parametric SSA filter, in its original form, is applied to a real-valued time series X_N of length N.

$$X_N = (x_0, x_1, \ldots, x_n, \ldots, x_{N-2}, x_{N-1}) \quad (3)$$

The filtering process consists of four steps, divided in two stages. The first is the Decomposition stage, which comprises the Embedding and SVD steps. The second is the Reconstruction stage, which performs the Grouping and the Diagonal Averaging steps. These steps are described as follows.

1. Decomposition

1.1. Embedding: This step maps the original 1-dimensional time series, given in (3), into a K-dimensional series $\boldsymbol{X_m}$, called the trajectory matrix, shown in (4).

$$\boldsymbol{X_m} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_K], \quad \boldsymbol{X}_k = [x_{k-1}, \ldots, x_{k+L-2}]^\top \quad (4)$$

This trajectory matrix is made of a set of K column vectors of length L that are subsets of the original time series, with $K = N-L+1$. It is a Hankel matrix, given that the elements in its anti-diagonals are the same. The filter parameter $L \in \mathbb{Z}$ is called the window

length or the embedding dimension and is constrained to $2 \leq L \leq N/2$. It represents the maximum number of components into which the original time series is decomposed.

1.2. Singular Value Decomposition:
The second step aims to decompose the trajectory matrix in (4) in terms of its singular vectors $\boldsymbol{U} \in \mathbb{R}^{L\times L}$ and $\boldsymbol{V} \in \mathbb{R}^{K\times K}$, and singular values $\boldsymbol{\Sigma} \in \mathbb{R}^{L\times K}$, by the use of the SVD method, resulting in (5).

$$\boldsymbol{X_m} = \boldsymbol{U\Sigma V}^T \tag{5}$$

where the singular vectors and values are in a decreasing order of magnitude: $\sigma_{11} \geq \sigma_{22} \geq \ldots \geq \sigma_{LK}$.

From the representation in (5), the trajectory matrix is then split into $q = \max\{i|\sigma_{ii} > 0\} \leq L$ elementary matrices, as illustrated in (6).

$$\boldsymbol{X_m} = \sum_{i=1}^{q\leq L} \sigma_i U_i V_i^T \tag{6}$$

resulting in a set of matrices $X_{\mathcal{D}}$ that represents the trajectory matrix of each component.

$$X_{\mathcal{D}} = \{\boldsymbol{X_{m_1}}, \boldsymbol{X_{m_2}}, \ldots, \boldsymbol{X_{m_q}}\}, \quad \boldsymbol{X_{m_i}} \in \mathbb{R}^{L\times K} \tag{7}$$

1.3. Eigendecomposition:
An alternative to the SVD method is the eigendecomposition of the covariance matrix, given in (8).

$$\boldsymbol{C_m} = \left(\boldsymbol{X_m X_m^\top}\right)/K, \qquad \boldsymbol{C_m} \in \mathbb{R}^{L\times L} \tag{8}$$

which results in a representation of the original covariance matrix in terms of its eigenvalues $\boldsymbol{S} \in \mathbb{R}^{L\times L}$ and eigenvectors $\boldsymbol{U} \in \mathbb{R}^{L\times L}$, as shown in (9).

$$\boldsymbol{C_m} = \boldsymbol{USU^T} \tag{9}$$

From this representation, the original trajectory matrix can be decomposed into $q = \max\{i|\lambda_{ii} > 0\} \leq L$ matrices, by the projection of each column in $\boldsymbol{X_m}$ onto the respective eigenvector in $\boldsymbol{U}$, as shown in (10).

$$\boldsymbol{X_m} = \sum_{i=1}^{q\leq L} U_i (X_i^\top U_i)^\top = \sum_{i=1}^{q\leq L} U_i X_{pc_i}^\top \tag{10}$$

where $\boldsymbol{X_{pc_i}} \in \mathbb{R}^{K\times L}$ are the time series principal components. This process results in the same set of decomposed matrices $X_{\mathcal{D}}$ as the one obtained in (7).

2. Reconstruction

2.1. Grouping:
In this step, the set of matrices $X_{\mathcal{D}}$, from the decomposition stage, are merged into disjoint groups. Defining $\mathcal{I} = \{I_1, I_2, \ldots, I_i, \ldots, I_p\}$ such that a given set of indices $I_i = \{i_1, i_2, \ldots, i_{p_i}\}$ corresponds to the matrix indices that belongs to the i-th group. Then, the grouping step is perform as in (11), by summing up all the matrices that belongs to that group.

$$\boldsymbol{X}_{I_i} = \sum_{i\in I_i} \boldsymbol{X}_{mi} = \boldsymbol{X}_{mi_1} + \boldsymbol{X}_{mi_2} + \cdots + \boldsymbol{X}_{mi_{p_i}} \tag{11}$$

Extending to all sets of indices, results in (12), a set of grouped matrices.

$$X_{\mathcal{I}} = \{\boldsymbol{X}_{I_1}, \boldsymbol{X}_{I_2}, \ldots, \boldsymbol{X}_{I_p}\} \tag{12}$$

This step relies on prior knowledge about the separability of the components, the magnitude of the eigenvalues, the existence of patterns between the eigenvectors, among others. The reader is referred to Golyandina and Zhigljavsky (2013a, p. 43) and Golyandina and Korobeynikov (2014, p. 12-15) for a more detailed discussion about these aspects.

2.2. Diagonal Averaging:
The step seeks to map each matrix resulting from the grouping step, back into a 1-dimensional time series, $\tilde{X}_N$, with the same length N as the original one.

$$\tilde{X}_N^{(k)} = (\tilde{x}_0^{(k)}, \tilde{x}_1^{(k)}, \ldots, \tilde{x}_n^{(k)}, \ldots, \tilde{x}_{N-2}^{(k)}, \tilde{x}_{N-1}^{(k)}) \tag{13}$$

Let $\boldsymbol{X_{I_i}} = \boldsymbol{A} = (a_{ij})_{i,j=1}^{L,K}$ represent one of the matrices obtained from the grouping step. The reconstructed time series in (13) is obtained by performing an average operation on each anti-diagonal of $\boldsymbol{A}$. This operation is shown in (14), and it is repeated for $n = 0, 1, \ldots, N-1$.

$$\tilde{x}_n^{(k)} = \begin{cases} \dfrac{1}{n+1}\displaystyle\sum_{i=1}^{n+1} a^*_{ij'}, & 0 \leqslant n < L^* - 1 \\ \dfrac{1}{L^*}\displaystyle\sum_{i=1}^{L^*} a^*_{ij'}, & L^* - 1 \leqslant n < K^* \\ \dfrac{1}{N-n}\displaystyle\sum_{i=n-K^*+2}^{N-K^*+1} a^*_{ij'}, & K^* \leqslant n < N \end{cases} \tag{14}$$

where $j' = n - i + 2$, $L^* = \min(L, K)$, $K^* = \max(L, K)$, $N = L + K - 1$, $a^*_{ij} = a_{ij}$ for $L < K$, and $a^*_{ij} = a_{ji}$ for $L \geq K$.

Extending this procedure for each component $k = 1, 2, \ldots, q \leq L$, the original time series is approximated by a reconstructed one, $\tilde{X}_N^R$, as shown in (15).

$$\tilde{X}_N^R = \sum_{k=1}^{R} \tilde{X}_N^{(k)} \tag{15}$$

where R is a parameter that controls the number of components used to approximate the time series, with $1 \leq R \leq L$. If $R = L$, then $\tilde{X}_N^R = X_N$, the reconstructed series is equal to the original one, if $R < L$, then $\tilde{X}_N^R$ is an approximation of X_N considering the first R most important components.

The SSA filtering steps are summarized in the pseudo-algorithm shown in 1.

Algorithm 1: SSA Algorithm

Input: $X_N \in \mathbb{R}^N$: original time series
$L \leq N/2 \in \mathbb{Z}$: embedding dimension
$R \leq L \in \mathbb{Z}$: components for reconstruction
Output: $\tilde{X}_N^R \in \mathbb{R}^N$: reconstructed time series
Function `SSA`(X_N, L, R)**:**
  $\boldsymbol{X_m} \leftarrow$ Embedding(X_N, L) # Trajectory matrix
  $X_{\mathcal{D}} \leftarrow$ Decomposition$(\boldsymbol{X_m})$ # SVD
  $X_{\mathcal{I}} \leftarrow$ Grouping$(X_{\mathcal{D}}, \mathcal{I})$
  $\tilde{X}_N^R \leftarrow$ DiagonalAveraging$(X_{\mathcal{I}}, R)$
end

3.3 SSA as a FIR Filter Bank

The SSA method can also be interpreted in terms of a linear filtering process. The projection operation that

results in the set of decomposed matrices $X_{\mathcal{D}}$ through (6) or (10), in the decomposition and the diagonal averaging operation, lead to the FB description of the SSA shown in Fig. 4.

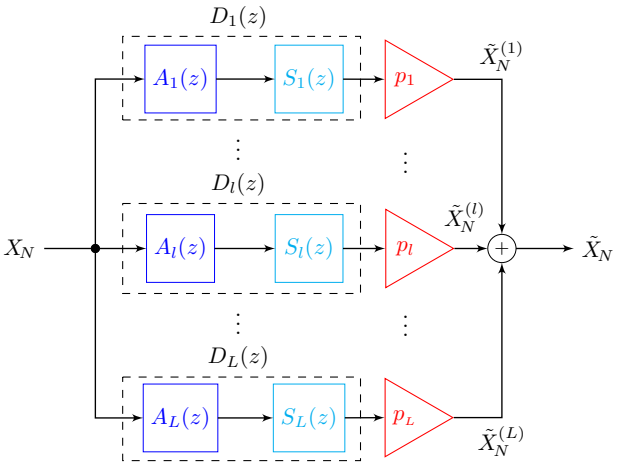

Figure 4. Block diagram of the SSA as a FIR FB.

The eigenvectors obtained from the decomposition of the trajectory matrix form the coefficients of the filters $A_l(z)$ and $S_l(z)$, for each component $l = 1, 2, \ldots, L$. Equation $A_l(z)$, given in (16), is a causal filter that represents the projection operation that results in the principal components, called forward filtering. Meanwhile, $S_l(z)$, given in (17), is an anti-causal filter and is responsible for the inverse projection and the diagonal averaging step by normalizing it by L, performing reverse filtering.

$$A_l(z) = \sum_{k=0}^{L-1} u_k^{(l)} z^{-k} \quad (16) \qquad S_l(z) = \frac{1}{L} \sum_{k=0}^{L-1} u_k^{(l)} z^{k} \quad (17)$$

where $\boldsymbol{u}^{(l)} = \left[u_0^{(l)}\, u_1^{(l)} \ldots u_k^{(l)} \ldots u_{L-1}^{(l)}\right]^\top$ are the eigenvectors of the l-th component.

The binary gain $p_l \in \{0, 1\}$ in Fig. 4 indicates whether or not a given component is selected for the reconstruction of the time series. The l-th branch of the FB, for $p_l = 1$, relates the l-th reconstructed component $\tilde{X}_N^{(l)}$ with the original time series X_N through the transfer function of a FIR filter $D_l(z)$ given by (18).

$$\begin{aligned} D_l(z) &= \frac{\tilde{X}_N^{(l)}(z)}{X_N(z)} = A_l(z) S_l(z) \\ &= \left(\sum_{k=0}^{L-1} u_k^{(l)} z^{-k}\right)\left(\frac{1}{L}\sum_{k=0}^{L-1} u_k^{(l)} z^{k}\right) \\ &= \sum_{k=-(L-1)}^{L-1} d_k^{(l)} z^k = \sum_{k=1-L}^{L-1} \left(\frac{1}{L}\sum_{i=|k|}^{L-1} u_i^{(l)} u_{|k|-i}^{(l)}\right) z^k \end{aligned} \quad (18)$$

According to Tomé et al. (2018), the filter coefficients $d_k^{(l)}$ are the entries of the matrix $\boldsymbol{D}_l = \boldsymbol{u}^{(l)} \boldsymbol{u}^{(l)\top}$ along the k-th diagonal. With the main diagonal given by $k = 0$, and above and below the main diagonal by $k > 0$ and $k < 0$, respectively. $\boldsymbol{D}_l$ is a symmetric matrix, $d_{-k} = d_k$, and the sum of each matrix component results in an identity matrix as shown in (19).

$$\sum_{l=1}^{L} \boldsymbol{D}_l = \sum_{l=1}^{L} \boldsymbol{u}^{(l)} \boldsymbol{u}^{(l)\top} = I_{L\times L} \implies \sum_{l=1}^{L} d_k^{(l)} = \delta(k) \quad (19)$$

where $\delta(k)$ is a unit impulse signal.

This implies that if the reconstruction is done with all the components, $p_l = 1$, $\forall_{l=1,2,\ldots,L}$, then the original time series is exactly recovered. Equivalent to convolving the series with an impulse in the time-domain.

In addition, the symmetry property leads the frequency response of the filter $D_l(z)$ to be expressed in a closed-form as in (20).

$$D_l(e^{j\omega}) = d_0^{(l)} + 2\sum_{k=1}^{L-1} d_k^{(l)} \cos(k\omega), \quad \forall_{l=1,2,\ldots,L} \quad (20)$$

Which is of a real-valued and zero-phase filter, implying that each component is not distorted and is in phase with the original time series.

Therefore, regarding the reconstruction stage, the SSA can be seen as a linear filtering process performed through a bank of FIR filters generated by the spectral decomposition of the time series in terms of its eigenvectors. Another characterization of SSA as eigenvector filters is provided by Leles et al. (2016).

3.4 Real-time SSA

The SSA filter requires all the samples of the time series to be available when applying the method. In order to use it in a real-time application, a modified version of it, was adopted in Leles et al. (2017, p. 4), called CSSA.

The method collects samples from a continuous time signal, $x_n = x(nT_s)$, with T_s the sampling period, to form a time series X_N of fixed length N. After that, as new samples arrive, the old ones are discarded, as in a sliding window.

Besides, for each incoming sample, x_n, instead of performing the diagonal averaging on each component's matrices, its reconstruction, $\tilde{x}_n$, is done directly from the updated eigenvectors $\boldsymbol{U}$ of the decomposition step, according to (21), which performs the same operation as (10).

$$\tilde{x}_n^{(k)} = \sum_{i=0}^{L-1} u_{L-1}^{(k)} u_i^{(k)} x_{n-L+i}, \quad n = 0, 1, 2, \ldots \quad (21)$$

where $u^{(k)}$ are the eigenvectors of the k-th component.

The reconstructed sample considering then the first R components of the time series is obtained by (22).

$$\tilde{x}_n^R = \sum_{k=1}^{R} \tilde{x}_n^{(k)} \quad (22)$$

The pseudo-algorithm of this filter is presented in 2.

In the CSSA algorithm, n and X_N represent global variables (maintain its state between function calls) that are initialized once. The filter executes as long as new samples arrive. For $n < N$, the filter is inactive. When $n = N$, the minimum number of samples has been reached, then the SSA filter is executed on X_N. After that, for $n > N$, the series is updated with the new sample, the eigendecomposition is performed on the new covariance matrix and the

Algorithm 2: CSSA Algorithm

Input: $x_n \in \mathbb{R}$: sample of the original time series
$N \in \mathbb{Z}$: minimum number of samples
$L \leq N/2 \in \mathbb{Z}$: embedding dimension
$R \leq L \in \mathbb{Z}$: components for reconstruction
Output: $\tilde{x}_n^R \in \mathbb{R}$: reconstructed sample
Function `CausalSSA`(x_n, N, L, R)**:**
  $n \leftarrow 0$, $X_N \leftarrow 0$
  while *new sample to process* **do**
    if $n < N$ **then**
      $X_N \leftarrow \text{AddSample}(X_N, x_{ns})$
      $\tilde{x}_n^R \leftarrow x_n$
    else if $n == N$ **then**
      $\tilde{X}_N^R \leftarrow \text{SSA}(X_N, L, R)$ # SSA filter
    else if $n > N$ **then**
      $X_N \leftarrow \text{UpdateTimeSeries}(X_N, x_n)$
      $\boldsymbol{X_m} \leftarrow \text{Embedding}(X_N, L)$
      $\boldsymbol{U} \leftarrow \text{Eigendecomposition}\left(\left(\boldsymbol{X_m X_m^\top}\right)/K\right)$
      $\tilde{x}_n^R \leftarrow \text{Reconstruction}(X_N, \boldsymbol{U})$ # (21)-(22)
    end
    $n \leftarrow n + 1$
  end
end

operations in (21) and (22) are executed to compute the reconstructed sample.

4. EXPERIMENTAL RESULTS

The proposed method consists of replacing the FIR filter of the trajectory generating scheme in Fig. 1 by the real-time version of the non-parametric SSA filter, the CSSA, as shown in Fig. 5.

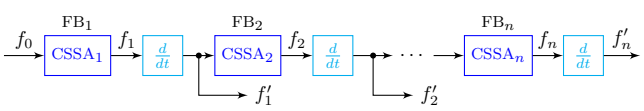

Figure 5. Block diagram of a non-parametric trajectory filter composed of a cascade of CSSA FIR FB.

Instead of a single parameter, as was T_i for the moving average filter, the CSSA requires three parameters to be tuned: (i) the sliding window length N; (ii) the embedding dimension L; and (iii) the number of components for reconstructing the time series R. Consequently, to investigate the response of this trajectory filter, the experiments carried out sought to vary one of the parameters, L or R, while the other remained unchanged.

The proposed method was compared to a parametric trajectory filter in state-space form, as described in Farrell and Polycarpou (2006) and shown in (23)-(25). In this approach, the low-pass filter in (23) is represented in its controllable canonical form in (24)-(25), which returns, as states, the $n-1$ desired derivatives of the input.

$$G = \frac{Y}{U} = \frac{a_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} \tag{23}$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \vdots \\ x^{(n-1)} \\ x^{(n)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & \dots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \vdots \\ x^{(n-2)} \\ x^{(n-1)} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ a_0 \end{bmatrix} u \tag{24}$$

$$y = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \vdots \\ x^{(n-1)} \end{bmatrix} \tag{25}$$

Figs. 6, 7, and 8 show the results of the CSSA and the parametric trajectory filters applied to the following signals: pulse, ramp and sawtooth, respectively. These signals and their filtered versions for the CSSA and the FIR filter are shown in Figs. 6-8 (a). While its first and second filtered derivatives are shown in Figs. 6-8 (b) and (c), respectively.

The beginning of the simulations in which the filter was not yet active but only accumulating samples to reach the sliding window length N are not shown. The simulations in Figs. 6 and 7 were performed for $N = 150$ samples, with sampling period $T_s = 0.01\,\text{s}$, signal's decomposition into $L = 100$ components, and reconstructing with only the first, $R = 1$. While in Fig. 8, were used $N = 50$, $T_s = 0.1\,\text{s}$, $L = 40$. For the parametric filter were chosen $a_0 = 10^2$, $a_1 = 120$ and $a_2 = 21$.

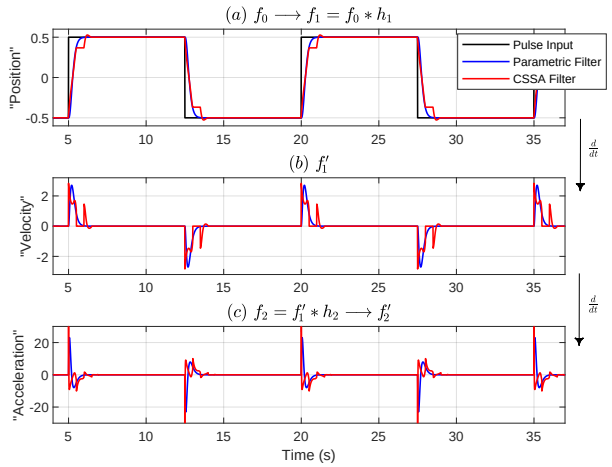

Figure 6. CSSA and a parametric trajectory filter for the first two derivatives of a discontinuous pulse signal: $N = 150$, $T_s = 0.01\,\text{s}$, $L = 100$, $R = 1$, $a_0 = 10^2$, $a_1 = 120$ and $a_2 = 21$.

The results in Figs. 6, 7 and 8 show that the parametric and non-parametric trajectory filters obtained similar results, with the non-parametric resulting in signals with a slightly larger amplitude. Nonetheless, they returned bounded derivatives from an initial discontinuous input signal, at least for Fig. 6 and 8. The results in Fig. 7 show that if the input is not discontinuous, then filtering it does not distort the signal derivatives significantly.

To further analyze and compare the non-parametric trajectory filtering response, the values of L and R were varied. The results are shown in Fig. 9, for L in $[10\!:\!5\!:\!45]$ for each value of $R = [1, 2, 3, 4]$, and Fig. 10, for R varying

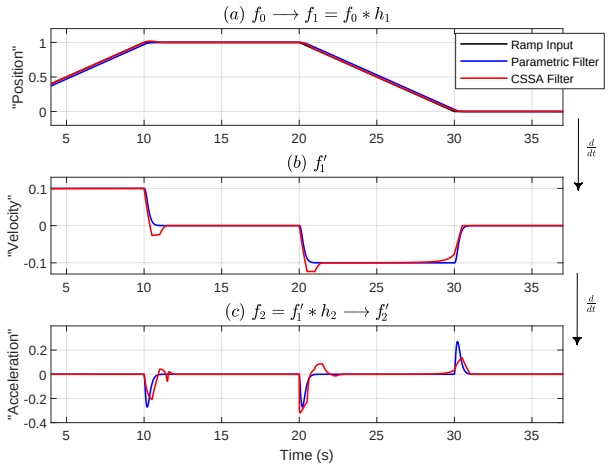

Figure 7. CSSA and a parametric trajectory filter for the first two derivatives of a continuous ramp signal: $N = 150$, $T_s = 0.01\,\text{s}$, $L = 100$, $R = 1$, $a_0 = 10^2$, $a_1 = 120$ and $a_2 = 21$.

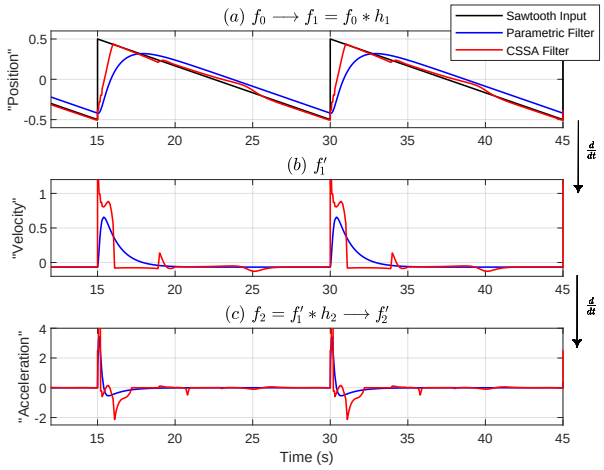

Figure 8. CSSA and a parametric trajectory filter for the first two derivatives of a continuous ramp signal: $N = 50$, $T_s = 0.1\,\text{s}$, $L = 40$, $R = 1$, $a_0 = 10^2$, $a_1 = 120$ and $a_2 = 21$.

in $[1, 2, \ldots, 10]$ for each value of $L = [10, 20, 30, 40]$. The simulation used $N = 50$, $T_s = 0.1\,\text{s}$, $a_0 = 10^3$, $a_1 = 10^2$ and $a_2 = 30$.

In Fig. 9 the best results were obtained for $R = 1$, the lowest possible value. In this case, by reconstructing the signal with its most important component, the method provides a ramp-like signal that was faster than the ramp obtained by the parametric filter. The exception was for $L = 25$. All the other values of $R > 1$, regardless of the values of L, provided faster signals, in comparison with the parametric filter. However, it resulted in a distorted signal with some overshoot or undershoot. Once again, the case for $L = 25$ exhibited the worst performance.

The fastest response is verified for $R = 1$ and $L = 45$, both in its extreme limits since $N = 50$ is an upper bound for L and R can not be lower than 1. Additionally, in all cases, the pair of values of $L = (10, 40)$, $(15, 35)$ and $(20, 30)$ showed a similar response. Indicating that lower values of L could achieve a similar smoothness degree. But with a lower computational cost, given that the decomposition occurs in a lower dimension.

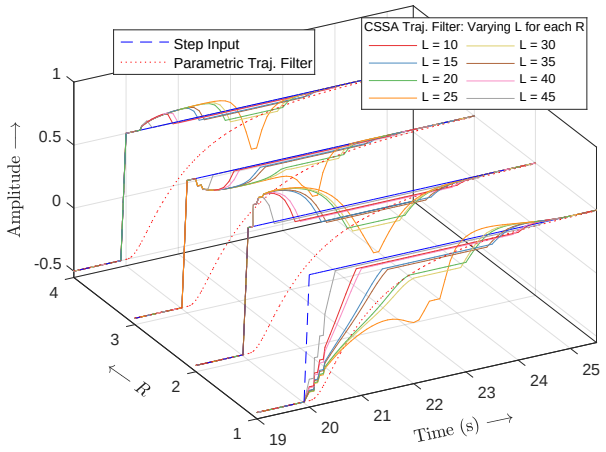

Figure 9. CSSA trajectory filter for a discontinuous step signal for varying L for each value of R: $N = 50$, $T_s = 0.1\,\text{s}$, $L = [10{:}5{:}45]$, $R = [1, 2, 3, 4]$, $a_0 = 10^3$, $a_1 = 10^2$ and $a_2 = 30$.

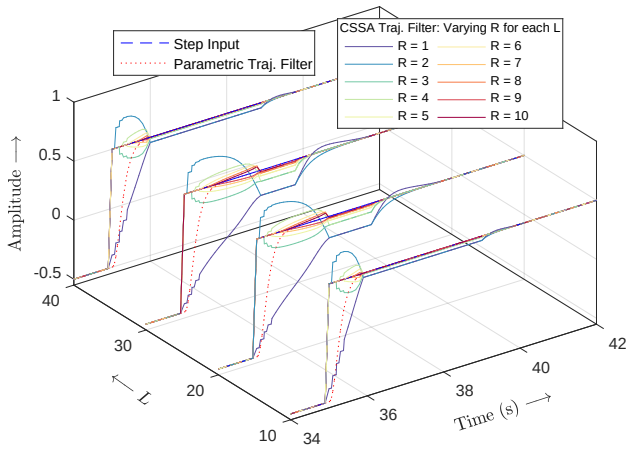

Figure 10. CSSA trajectory filter for a discontinuous step signal for varying R for each value of L: $N = 50$, $T_s = 0.1\,\text{s}$, $L = [10, 20, 30, 40]$, $R = [1, 2, \ldots, 10]$, $a_0 = 10^3$, $a_1 = 10^2$ and $a_2 = 30$.

Similar results were verified considering the inverse process, i.e., by varying R for each constant value of L, as shown in Fig. 10. For any value of L considered, only for $R = 1$, the non-parametric trajectory filter was able to smooth the step signal to some degree. The reconstruction of the original signal with any other component besides the first and most important one does not result in a smoother signal, however, it still results in bounded and fast changes, approximating the step signal.

5. CONCLUSIONS AND FUTURE PERSPECTIVES

This paper investigated the use of CSSA as a novel trajectory filter approach, suitable for nonlinear control methods. The exploratory results showed that the proposed non-parametric CSSA filter could be employed in an online trajectory generating scheme. The simulations returned

bounded derivatives from initially discontinuous input signals. The proposed filter followed the same principle of a cascade of FIR filters, which is the successive filtering of the input signal and its derivatives. These results are encouraging, since the frequency analysis for nonlinear system is a far more complex task than the linear counterpart.

One of the advantages of CSSA are the additional degrees of freedom provided by the parameters available. The proposed trajectory filter could also be used for more complex input signals, with non-stationary characteristics. For instance, to identify and extract specific components and attenuate noise. In addition, it is an adaptive filter, whose parameters could be automatically updated according to structural changes in the signals, in future researches. A disadvantage is the lack of systematic methods for SSA parameters selection, which is a open problem in the signal processing field. In general, there are some rules of thumb, depending on the kind of application.

Further studies could test its application on more complex signals and apply it to real control problems. Deriving rules for tuning the parameters based on the behavior of some well-known input signals is another possibility.

ACKNOWLEDGMENTS

This work has been supported by the Brazilian agencies CAPES and CNPq.

REFERENCES

Alexandrov, T. (2009). A method of trend extraction using singular spectrum analysis. *Revstat–Statistical Journal*, 7(1), 1–22.

Besset, P. and Béarée, R. (2017). FIR filter-based online jerk-constrained trajectory generation. *Control Engineering Practice*, 66, 169–180.

Biagiotti, L. and Melchiorri, C. (2012). FIR filters for online trajectory planning with time-and frequency-domain specifications. *Control Engineering Practice*, 20(12), 1385–1399.

Farrell, J.A. and Polycarpou, M.M. (2006). *Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches*. John Wiley & Sons.

Gerelli, O. and Bianco, C.G.L. (2010). A discrete-time filter for the on-line generation of trajectories with bounded velocity, acceleration, and jerk. In *2010 IEEE International Conference on Robotics and Automation*, 3989–3994. IEEE.

Golyandina, N. and Korobeynikov, A. (2014). Basic singular spectrum analysis and forecasting with R. *Computational Statistics & Data Analysis*, 71, 934–954.

Golyandina, N., Nekrutkin, V., and Zhigljavsky, A.A. (2001). *Analysis of time series structure: SSA and related techniques*. CRC press.

Golyandina, N. and Zhigljavsky, A. (2013a). *Singular Spectrum Analysis for Time Series*. Springer Berlin/Heidelberg.

Golyandina, N. and Zhigljavsky, A. (2013b). SSA for forecasting, interpolation, filtration and estimation. In *Singular Spectrum Analysis for Time Series*, 71–119. Springer.

Hassani, H. (2007). Singular spectrum analysis: Methodology and comparison. *Journal of Data Science*, 5, 239–257.

Jeon, J.W. and Ha, Y.Y. (2000). A generalized approach for the acceleration and deceleration of industrial robots and CNC machine tools. *IEEE transactions on industrial electronics*, 47(1), 133–139.

Kim, D.I., Jeon, J.W., and Kim, S. (1994). Software acceleration/deceleration methods for industrial robots and CNC machine tools. *Mechatronics*, 4(1), 37–53.

Leles, M.C.R., Mozelli, L.A., and Guimarães, H.N. (2017). A new trend-following indicator: Using SSA to design trading rules. *Fluctuation and Noise Letters*, 16(02), 1750016.

Leles, M.C., Cardoso, A.S., Moreira, M.G., Guimaraes, H.N., Silva, C.M., and Pitsillides, A. (2016). Frequency-domain characterization of singular spectrum analysis eigenvectors. In *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 22–27. IEEE.

Leles, M.C., Sansão, J.P.H., Mozelli, L.A., and Guimarães, H.N. (2018). A new algorithm in singular spectrum analysis framework: The overlap-ssa (ov-ssa). *SoftwareX*, 8, 26–32.

Lu, Y.S. (2008). Smooth speed control of motor drives with asymptotic disturbance compensation. *Control Engineering Practice*, 16(5), 597–608.

Olabi, A., Béarée, R., Gibaru, O., and Damak, M. (2010). Feedrate planning for machining with industrial six-axis robots. *Control Engineering Practice*, 18(5), 471–482.

Tomé, A.M., Malafaia, D., Teixeira, A.R., and Lang, E.W. (2018). On the use of singular spectrum analysis. *arXiv preprint arXiv:1807.10679*.

Zanasi, R., Bianco, C.G.L., and Tonielli, A. (2000). Nonlinear filters for the generation of smooth trajectories. *Automatica*, 36(3), 439–448.

Zanasi, R. and Morselli, R. (2003). Discrete minimum time tracking problem for a chain of three integrators with bounded input. *Automatica*, 39(9), 1643–1649.

Zhang, H. and Wang, Z. (2016). Attitude control and sloshing suppression for liquid-filled spacecraft in the presence of sinusoidal disturbance. *Journal of Sound and Vibration*, 383, 64–75.

Zheng, C., Su, Y., and Müller, P.C. (2009). Simple online smooth trajectory generations for industrial systems. *Mechatronics*, 19(4), 571–576.

Zhigljavsky, A. (2010). Singular spectrum analysis for time series: Introduction to this special issue. *Statistics and its Interface*, 3(3), 255–258.