Efficient Volterra Systems Identification Using Hierarchical Genetic Algorithms

Laura S. de Assis, Jurair R. de P. Junior, Luis Tarrataca, Diego B. Haddad

Federal Center of Technological Education of Rio de Janeiro - CEFET/RJ - RJ, Brazil

Abstract

The Volterra series consists of a powerful method for the identification of non-linear relationships. However, the identification of the series active basis sets requires intense research in order to reduce the computational burden of such a procedure. This is a result of a large number of measurements being required in order to produce an adequate estimate, due to overparameterization issues. In this work, we present a robust hierarchical evolutionary technique which employs a heuristic initialization and provides robustness against noise. The advanced solution is based on a genetic algorithm which improves on the computational complexity of existing methods without harming the identification accuracy. The impact of the parameters calibration is evaluated for different signal-to-noise levels and several nonlinear systems considered in the literature.

Keywords: Volterra Series, System Identification, Genetic Algorithms.

1. Introduction

The identification task can be interpreted as recovering the system hidden states by exploiting its input-output relationship data in the digital domain. Important applications of such a data-driven representation task are: (i) acoustic echo cancellation [1]; (ii) mitigation of intersymbolic interference by pre-distortion techniques [2]; (iii) spec-

URL: laura.assis@cefet-rj.br (Laura S. de Assis), jurair.junior@cefet-rj.br (Jurair R. de P. Junior), luis.tarrataca@cefet-rj.br (Luis Tarrataca), diego.haddad@cefet-rj.br (Diego B. Haddad)

tral regrowth analysis [3]; and (iv) active noise control [4]. Note that some of these examples require the identification of nonlinearities, which is especially important when their impact on the single-valued output is not negligible or the input signal range is large [5]. Nonlinearities effects can be caused by (sometimes complex) physical nonlinear phenomena, such as operation near the saturation region [6], intermodulation distortion [7], and diffusion capacitance [8]. In practice, nonlinear systems can be accurately modeled with limited prior knowledge by Volterra[1] series [9, 10], which are essentially a flexible (and always stable [11]) functional series expansion of a nonlinear time-invariant system. These series have the advantage of taking into account memory effects, in contrast to static nonlinear models [12, 13]. The generality of Volterra models can be shown either by interpreting them as discrete-time systems with fading memory or by the application of the Stone-Weierstrass theorem to the approximation of input/output finite-memory mappings [11, 14, 15]. The result published in [16] deserves mention, since it states the existence of a locally convergent power-series-like expansion of a large class of systems that contain an arbitrary (although finite) number of nonlinear elements.

It is noteworthy that Volterra modeling is an enduring research problem due to its wide range of applications [17]. Such models describe with conceptual simplicity the system output as a sum of a first-order operator, a second-order operator and higher-order operators, generalizing the convolution concept from linear time-invariant systems [18, 19, 5]. As the memory (or delays) and orders become larger, the number of the Volterra coefficients (each of them unequivocally associated to a kernel or basis waveform) increases geometrically. This makes the identification task a very challenging one, especially when there is a lack of knowledge about the operating principle and/or the structure of the device to be identified [20, 5]. In optical transmissions systems, for example, as the transmission capacity increases, the computational burden

[1] Such a name derives from the work of the Italian mathematician Vito Volterra (1860-1940).

required for standard techniques to model the communication system may be unacceptable [21]. Due to these facts, it is important to identify such systems with a low computational burden. Furthermore, robustness against the ubiquitous noise is crucial.

The global search feature of evolutionary algorithms avoids the local minima trapping phenomenon in non-convex or NP-hard optimization problems, providing an effective search procedure for different research areas [22]. Such properties enable them to become a natural choice for the selection of proper basis Volterra sets. In general terms, an evolutionary algorithm deals with individuals, aiming to encounter a proper point that conveniently addresses the inherent trade-off between the exploration and exploitation abilities of the stochastic search [22]. Each individual is unequivocally mapped into a candidate solution of an objective function defined for optimization purposes. The value of such a function, evaluated using a properly mapped individual as an argument, is employed as a fitness evaluation of the candidate solution.

Several families of evolutionary schemes have been advanced, such as particle swarm optimization [23], multiobjective decomposition-based algorithm [24], genetic programming [25], reaction optimisation [26], indicator-based algorithms [27], firefly algorithms [28], artificial bee or ant colony algorithms [29], differential evolution [30], learning automata-based selection [31]. To our knowledge, none of these evolutionary techniques was ever employed in order to address the identification of Volterra systems.

The focus of this paper is on genetic algorithms [32], which can be regarded as a nature inspired meta heuristics that also enforces an evolutionary strategy. Accordingly, we propose a genetic algorithm that efficiently takes into account the idiosyncrasies of Volterra-based identification tasks. The first attempt to use genetic algorithms (GAs) for the identification of Volterra systems was devised in [33], which encoded the active kernels by binary chromosome representation. This paper (as well as the very similar approach of [34]) assumed a multi-objective performance criterion, combining both mean squared error (*i.e.*, the ℓ_2-norm) and the maximum error (*i.e.*, the ℓ_∞-norm of the

error). Work [35] proposed the usage of the least squares procedure for the estimation of the coefficients of the supposed-to-be active kernels. Reference [36] encoded the location of active kernels using B bits, which requires precautions against non-factible locations. The employment of genetic algorithms was also proposed by [37], which aims to capture the nonlinear relationships of functional link networks, consisting of high-order perceptrons that may be equivalently rewritten as Volterra models. The floating-point genetic algorithm presented in [38] combines the kernels selection and coefficients identification steps in one single evolutionary step. In [39] an adaptive variable-length GA was proposed whose chromosomes encode the selected candidate's coefficients. The initialization procedure of this solution assumed to be active the basis functions where the correlation magnitude with the output was large.

This paper proposes an efficient genetic algorithm-based solution for the identification of time-domain Volterra series, suited to get a representation of complex nonlinear systems when a physically-based model is not available [5]. The memory length is assumed to be finite and upper bounded by a known value. The proposed algorithm takes into account the sparsity property that Volterra systems often present in practice [40, 41]. This avoids the need to estimate all kernel coefficients in each step, since often only a small number of them may contribute significantly to the output signal [35, 42]. Furthermore, an initialization procedure that chooses the most promising kernels ith higher probabilities is adopted. Sparsity-aware Volterra identification methods typically require a judicious pruning in order to reduce the basis set size [43], and the proposed method is not an exception.

The data structures of the proposed GA-based methodology were suitably selected to allow a customized hierarchical search of proper solutions in practical systems, spending little computational time for such an identification task. This hierarchical feature presents the potential to address large problems in an efficient way [44].

This paper is structured as follows. Section 2 presents the theoretical modelling

regarding the Volterra series. Section 3 presents our advanced GA approach towards identifying the basis sets of time-domain Volterra series. Section 4 discusses the experimental setup and respective results obtained. Section 5 presents the main conclusions of this work.

2. Volterra Series Identification Model

This paper focuses on the identification of single input single output nonlinear systems. In the case of continuous-time systems, one may write the output $y(t)$ as a sum of response components $x_n(t)$ [45]:

$$y(t) = \sum_{n=1}^{\infty} x_n(t), \tag{1}$$

where the n-th component is described by:

$$x_n(t) \triangleq \underbrace{\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty}}_{n\times} \overline{h}_n(\tau_1, \dots, \tau_n) \prod_{i=1}^{n} u(t-\tau_i) d\tau_1 \dots d\tau_n, \tag{2}$$

where $\overline{h}_n(\tau_1, \dots, \tau_n)$ is the n-th order Volterra kernel (or the n-th-order impulse response of the non-linear system [46]) and $u(t)$ is the system input. Since the continuous-time modelling of (1)-(2) presents several limitations in practice [45], henceforth a discrete-time counterpart with finite support is assumed.

The output $y[k] \in \mathbb{R}$ of a double-truncated discrete-time Volterra filter whose input is denoted by $x[k] \in \mathbb{R}$ is modeled as:

$$y[k] = \sum_{p=1}^{M} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \cdots \sum_{n_p=0}^{N-1} h_p[n_1, \cdots, n_p] \prod_{l=1}^{p} x[n-n_l] + \nu[k], \tag{3}$$

where $N \in \mathbb{N}$ is the memory depth, $h_p[n_1, \cdots, n_p] \in \mathbb{R}$ denotes the coefficients of the p-th order kernel (or polynomial basis function) and $M \in \mathbb{N}$ is the maximum nonlinearity order [47], which can be estimated with experimental or theoretical methods [48, 49].

The Volterra model (11) is assumed to be symmetric, so that $h_p[n_1,\cdots,n_p]$ maintains its value irrespective of the $p!$ possible permutations of the indices $n_1,\cdots,n_p$ [46, 42]. The measurement process of the output gives place to a measured output $d[k]$ described as:

$$d[k] = y[k] + \nu[k], \tag{4}$$

where $\nu[k] \in \mathbb{R}$ accounts for measurement noise, interferences and/or error modeling errors. Assuming the availability of L output samples, one may model the input-output mapping as:

$$\boldsymbol{X}\boldsymbol{w}^{\star} \approx \boldsymbol{d}, \tag{5}$$

where[2] the approximation derives from the existence of the noise $\nu[k]$, $\boldsymbol{X} \in \mathbb{R}^{L\times R}$, $\boldsymbol{w}^{\star} \in \mathbb{R}^{R}$, $\boldsymbol{d} \in \mathbb{R}^{L}$, and R depends on N and M through [36]:

$$R = \sum_{i=1}^{M} \binom{N+i-1}{i}. \tag{6}$$

More specifically, vector $\boldsymbol{d}$ can be defined as:

$$\boldsymbol{d} \triangleq \begin{bmatrix} d[0] & d[1] & \dots & d[L-1] \end{bmatrix}^{T}, \tag{7}$$

and the (unknown) vector $\boldsymbol{w}^{\star}$ contains the R ideal coefficients one intends to estimate. Notice that R, defined in Eq. (6), depends on M and N, and is also the number of columns of $\boldsymbol{X}$. In the case $N = M = 2$, one has $R = 5$ and $\boldsymbol{X}$ assumes the following

[2] All vectors of this paper are of column-type.

structure:

$$\boldsymbol{X} = \begin{bmatrix} x[0] & x[-1] & x^2[0] & x^2[-1] & x[-1]x[0] \\ x[1] & x[0] & x^2[1] & x^2[0] & x[0]x[1] \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ x[L-1] & x[L-2] & x^2[L-1] & x^2[L-2] & x[L-1]x[L-2] \end{bmatrix}. \tag{8}$$

Please note that the i-th column of $\boldsymbol{X}$ is associated to the i-th kernel (or basis set), so that a system whose output is written as:

$$y[k] = 0.5x[k-1] - 0.3x^2[k] + 0.1x[k]x[k-1] \tag{9}$$

120 presents an ideal vector[3] $\boldsymbol{w}^\star$ given as:

$$\boldsymbol{w}^\star = \begin{bmatrix} 0 & 0.5 & -0.3 & 0 & 0.1 \end{bmatrix}^T, \tag{10}$$

whose zero elements correspond to inactive kernels (i.e., $x[k]$ and $x^2[k-1]$). An estimate $\hat{\boldsymbol{w}}$ of $\boldsymbol{w}^\star$ emulates the actual system output by performing the following evaluation:

$$\hat{y}[k] = \hat{\boldsymbol{w}}^T \boldsymbol{x}[k], \tag{11}$$

where $\boldsymbol{x}[k]$ is a column-vector that contains the elements of the k-th row of matrix $\boldsymbol{X}$. The discrepancy between the measured output and the estimated output is incorporated into the error signal $e[k]$, defined as:

$$e[k] \triangleq d[k] - \hat{y}[k], \tag{12}$$

whose magnitude consists of a stochastic assessment of the identification procedure [50]. The vector $\boldsymbol{e} \in \mathbb{R}^L$ (or residue vector) collects the error samples of a specific candidate

[3]Note that still it is assumed here that $N = P = 2$.

$\hat{\boldsymbol{w}}$:

$$\boldsymbol{e}\left[\hat{\boldsymbol{w}}\right] \triangleq \begin{bmatrix} e[0] & e[1] & \dots & e[L-1] \end{bmatrix}^T. \tag{13}$$

A naive approach to estimate $\boldsymbol{w}^\star$ can be implemented by the least squares (LS) method, which minimizes the ℓ_2-norm of the residue vector, providing the following closed-form solution [51, 52]:

$$\hat{\boldsymbol{w}}_{\mathrm{LS}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{d}, \tag{14}$$

where $\hat{\boldsymbol{w}}_{\mathrm{LS}}$ is the LS estimate of $\boldsymbol{w}^\star$. It is possible to regard the LS regression as a special case of the more general method of maximum likelihood [53]. Since the number of parameters of a Volterra system is large even for models of moderate size, the corresponding least-squares computation routines become prone to numerical errors. This requires that the number of measurements should be much larger than the number of model parameters [9]. Such a fact motivates the usage of alternative approaches (such as the one advanced in this paper) that present robustness against overfitting or overparameterization.

In this paper, an accurate estimation of the active kernels (associated with columns of matrix $\boldsymbol{X}$) is the main goal. A candidate solution should indicate which columns of $\boldsymbol{X}$ are to be included in the nonlinear identification model. The nonzero kernel coefficients associated with such a solution may be computed through

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{X}_{\mathrm{part}}^T\boldsymbol{X}_{\mathrm{part}}\right)^{-1}\boldsymbol{X}_{\mathrm{part}}^T\boldsymbol{d}, \tag{15}$$

where $\boldsymbol{X}_{\mathrm{part}}$ is obtained by removing the columns of $\boldsymbol{X}$ whose kernels are estimated as inactive.

3. Proposed Approach

In this section we propose a method for the identification of time-domain Volterra series using genetic algorithms (GA), in order to conveniently address the existence of multiple local optima solutions. In brief, genetic algorithms operate on a population of individuals, where each individual is a potential solution to a problem. Frequently, GA-based approaches ensure higher convergence rates of the search procedure, when compared against conventional gradient-based techniques [54]. In this paradigm, each individual is typically encoded as a fixed-length binary string. After defining an individual, the next step is to generate a population (randomly or heuristically). Three genetic operators are then applied to the population, sequentially and interactively, namely: (i) selection, (ii) crossover and (iii) mutation, which puts forth a new generation. Such operators tend to improve the quality of the entire population, in a statistical sense [55]. The following sections are organized as follows: Section 3.1 describe the core concepts behind GAs; Section 3.2 characterizes the hierarchical structure employed; and Section 3.3 presents proposed enhancements such as a randomized constructive heuristic and a constructive heuristic oriented by benefit.

3.1. Main Genetic Algorithm Concepts

3.1.1. Solution Encoding

A solution encoding scheme to find the best number and positions of Volterra kernels can be represented by a chromosome composed of a binary array representing which kernel is part of the solution. The value (*allele*) assigned to each position (*locus*) indicates whether the i-th kernel was chosen to be active in the identification procedure. A locus with allele equal to 1 means that the kernel represented by this position is part of the system. Conversely, an allele with value 0 symbolizes that the kernel was not selected to be part of the identification system. Figure 1 exemplifies the encoding for the system shown in Eq. (16), using the indexing of a Volterra series when: (i) the

memory length is $N = 3$ and (ii) terms up to second order are present (i.e., $M = 2$; see Table 1).

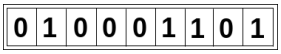


Figure 1: Chromosome representation.

$$\begin{aligned} d[k] &= w_1^\star x[k-1] + w_5^\star x[k]x[k-2] + w_6^\star x^2[k-1] \\ &\quad + w_8^\star x^2[k-2], \end{aligned} \tag{16}$$

where $w_i^\star$ is the i-th coefficient of the ideal impulse response.

Table 1: Example of a Volterra serie indexing.

Index	Component
0	$x[k]$
1	$x[k-1]$
2	$x[k-2]$
3	$x^2[k]$
4	$x[k]x[k-1]$
5	$x[k]x[k-2]$
6	$x^2[k-1]$
7	$x[k-1]x[k-2]$
8	$x^2[k-2]$

3.1.2. Fitness

Each chromosome is evaluated and assigned to a fitness value. As in other typical optimization problems, the fitness function adopted in this work is the objective (fitness) function $\mathcal{F}$, defined as:

$$\mathcal{F}\left[\hat{\boldsymbol{w}}\right] \triangleq \mathbb{E}\left\{\exp\left[-e^2(k)\right]\right\}, \tag{17}$$

where $\mathbb{E}[\cdot]$ is the expectation operator, which in practice is estimated by the average procedure in a realization of the underlying stochastic process. The exponentiation function is employed in (17) in order to avoid that outlier observations degrade significantly the estimation procedure [56, 57]. This procedure can be motivated by the correntropy, which is a similarity measure that generalizes the linear correlation function to nonlinear spaces [58]. Such a criterion is more suitable for nonlinear signal processing problems [59]. Note that the error $e(k)$ in Equation (17) is evaluated using (15) and (11)-(13).

3.1.3. Selection operation

The selection operator is crucial for the convergence characteristics of GA. In this work, the selection of which individuals participate of the crossover is performed in accordance with population structure (see Section 3.2). Accordingly, each pair of individuals, leader and subordinate, of the population, is selected for a crossover operation.

3.1.4. Crossover operation

After the pairs of individuals have been selected, the crossover operation is performed in order to generate new individuals in the population. During the reproductive process, a permutation of the parents genetic material is performed. Namely, a one-point crossover is used, where the two parents are cut once at specific randomly chosen point selected to split their chromosomes. This procedure is illustrated in Figure 2.

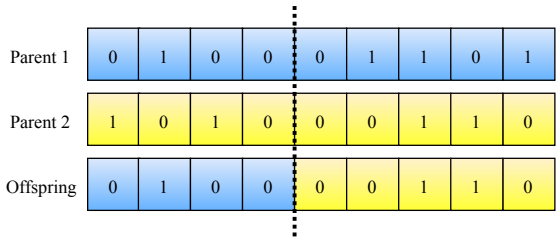


Figure 2: Example of crossover operation.

Algorithm 1 shows the pseudocode for the crossover operation. (I_l, I_s), are the

leader and supporter individuals, respectively; c_s is the size of the chromosome and I_o is the offspring resulting from the crossover. Procedure *Random(0, 1)* is a function that provides a random number in the interval $[0, 1]$. Please note that Algorithm 1 executes in $\Theta(c_s)$ time.

Algorithm 1 Crossover(I_l, I_s, c_s)

```
1: p_cut ← Random(1, c_s)
2: for i ← 1 to p_cut do
3:     I_o[i] ← I_l[i]
4: end for
5: for i ← p_cut + 1 to c_s do
6:     I_o[i] ← I_s[i]
7: end for
8: return I_o
```

3.1.5. Mutation operation

The mutation operator is very significant to avoid premature convergence when most individuals of the population present genetic information that is very similar. Considering a mutation rate, this operation aims to diversify the genetic material in the new generation of individuals. When a locus is selected for mutation, the method considers two possibilities:

- *Locus without kernel:* the i-th kernel is set (allele transitions from $0 \rightarrow 1$);
- *Locus with kernel:* the i-th kernel is removed from the system (allele transitions from $1 \rightarrow 0$).

The pseudocode of the mutation operation is presented in Algorithm 2. I_o is the offspring resulting from the crossover, m_p is the probability of the mutation and c_s is the size of the chromosome. Similarly to the crossover pseudocode, Algorithm 2 also executes in $\Theta(c_s)$ time.

Algorithm 2 Mutation(I_o, m_p, c_s)

```
I_m ← I_o
for i ← 1 to c_s do
    R ← Random(0,1)
    if R > m_p then
        if I_o[i] == 0 then
            I_m[i] ← 1
        else
            I_m[i] ← 0
        end if
    end if
end for
return I_m
```

3.2. Hierarchical Population Structure

In the proposed approach we adopted a hierarchical structure, in order to reduce the computational burden associated with calculating solutions. Previous experiences in solving combinatorial optimization problems using genetic algorithms show that a hierarchically structured population leads to performance improvements over non-structured population approaches. Accordingly, in this work, the population of individuals is organized as a ternary tree, composed of four clusters with four individuals; each one composed of a leader and three subordinate individuals [60]. As illustrated in Figure 3, the individuals' hierarchy is established by their fitness, where a *leader* individual resides above (and has a better fitness than) its three *subordinates* individuals. As a result, the best solution is always placed at the upper cluster (*i.e.*, at the root node of the tree).

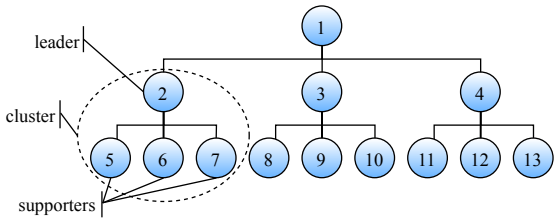


Figure 3: Example of the individuals' hierarchy in terms of their fitness.

In evolutionary algorithms approaches, employing a structured population has the benefit of using a small number of individuals when compared to typical non-structured populations. The computational effort is reduced, since fewer fitness evaluations are required. Empirical studies have shown that population structure and hierarchy among individuals overcome the requirement for a larger population that is needed to maintain the exploration effectiveness of the genetic algorithm [60, 61]. The number of individuals in the population is the number of nodes in a complete ternary tree, *i.e.*, $(3^h - 1)/2$, where h is the height of the tree [61]. In this work, the population has 13 individuals, resulting from using a ternary tree with three levels.

3.3. Initial Population

Two population initialization procedures are proposed and compared: (i) a randomized constructive heuristic (RCH) and (ii) a constructive heuristic oriented by benefit (CHOB), inspired by [38]. Both procedures begin with an empty population and generate a feasible initial population (see Section 3.4). Algorithm 3 illustrates the ideas behind these procedures. The main difference concerns the probability of choosing a specific kernel.

For both RCH and CHOB, each main iteration (lines 2-15) constructs one feasible individual. Each individual starts without active kernels (lines 3). The inner loop (lines 5-12), attempt to position a kernel in the i-th locus with probability τ_i.

The RCH represents a purely random choice which sets $\tau_i = 0.5$ for $i \in \{0, 1, \ldots, c_s - 1\}$, *i.e.*, the probability of a kernel composing the solution or not is 50%. In order to improve the initial solutions, the alternative CHOB was proposed, which considers the benefit of a specific kernel for system identification. For this procedure $\tau_i = B_i$, where

Algorithm 3 GenerateInitialPopulation()

```
1:  P ← {}
2:  while (|P| ≤ popSize) do
3:     I ← 0
4:     while (!isFeasibleIndividual(I)) do
5:        for (i = 0; i < c_s; i++) do
6:           R ← Random(0,1)
7:           if R ⩽ τ_i then
8:              I_i ← 1
9:           else
10:             I_i ← 0
11:          end if
12:       end for
13:    end while
14:    P ← P ∪ {I}
15: end while
16: return P
```

B_i is the benefit of choosing the i-th kernel, which can be calculated by:

$$B_i = \left| \sum_{l=0}^{L-1} \frac{(x[l,i] - \bar{x}_i) \times (d[l] - \bar{d})}{\sqrt{\sum_{l=0}^{L-1} (x[l,i] - \bar{x}_i)^2 \times \sum_{l=0}^{L-1} (d[l] - \bar{d})^2}} \right|, \tag{18}$$

where $x[l,i]$ is the element of matrix X located at the (l,i) position, L is the number of samples, x is a one-dimensional vector, and $\bar{x}_i$ and $\bar{d}$ are the average of the i-th column of matrix $\boldsymbol{X}$ and of vector $\boldsymbol{d}$, respectively. B_i gives a measure of the benefit achievable by each possible kernel allocation. Thus the probability of selecting the kernel with a better benefit is greater than the probability of selecting a kernel with a worse benefit. Note that Equation (18) computes the bivariate linear correlation B_i between the i-th column of matrix $\boldsymbol{X}$ and the measured system response $\boldsymbol{d}$, where $0 \leq B_i \leq 1$. It is noteworthy that the evaluation of B_i is required only once, since it does not depend on the current population.

The CHOB generates feasible solutions faster than the naiver approach RCH. Generally, constructive heuristics generate higher quality initial populations [62, 63].

In Algorithm 3 *popSize* represents the number of individuals present in the population and has value $(3^h - 1)/2$; I is a chromosome that represents a solution, as described in 3.1.1. The algorithm executes in time $O(3^h \lambda c_s)$, where λ is an upper bound on the number of iterations that need to be performed to obtain a feasible individual.

3.4. Feasibility

In order to overcome overfitting issues, there are constraints regarding the number of active kernels in the chromosome. Namely, an individual is considered unfeasible whether if it has zero active kernels or presents an excessive number of active basis sets. $R_{\max} + 1$ is a lower-bound on the number of active kernels required for an individual to be considered unfeasible. It is important to mention that $R_{\max}$ is a user-defined parameter. In practice, the choice of the $R_{\max}$ value should be oriented by prior information about the specific problem the identification procedure intends to solve.

3.5. Genetic Algorithm

Algorithm 4 presents the pseudo-code for the method proposed in this work. The procedure is based on a constructive heuristic used within a genetic algorithm, which includes the previously mentioned selection, crossover, and mutation operators as well as the hierarchical population structure (P). In order to introduce diversity, the main iteration (lines 1-19) allows more than one restart in the population of individuals. The core of the GA is observed in the loop shown in lines 3-18, where the structure of the population is hierarchically reorganized (this is performed through the `SortTree()` method, see Section 3.2). Furthermore, each pair of individuals, incorporating a leader and subordinate, is selected for the crossover operation, and each generate offspring can mutate according to a certain probability. If the generated offspring is unfeasible, a procedure to make it feasible is performed. Also, if it has a better fitness than its subordinate parent then the offspring occupies the parent's position. Otherwise, the offspring dies. The algorithm executes in $O\left(GA_{\text{Resets}}(3^h C_s \lambda + \varphi(3^h \log 3^h + |P|C_s))\right)$, where φ represents the upper bound of generations.

Algorithm 4 Genetic algorithm with a structured population

```
for number of GA resets do
    P ← GenerateInitialPopulation();
    while (numGenerations < limitGen) do
        P ← SortTree()
        for each pair (leader, subordinate) ∈ P do
            offspring ←Crossover(leader, subordinate)
            R ← Random(0,1)
            if R ⩽ T then
                offspring ←Mutation(offspring)
            end if
            if Unfeasible(offspring) == true then
                offspring ←MakeFeasible(offspring)
            end if
            if Fitness(offspring) > Fitness(subordinate) then
                subordinate ← offspring
            end if
        end for
    end while
end for
```

4. Experimental Results

In the forthcoming simulations, the measurement noise signal $\nu[k]$ is assumed to be a white Gaussian signal. Its variance is chosen accordingly to the considered signal-to-noise ratio (SNR), in dB, defined as:

$$\text{SNR (dB)} \triangleq 10.\log_{10}\frac{\mathbb{E}[y^2[k]]}{\mathbb{E}[\nu^2[k]]}. \tag{19}$$

The identification of the following three distinct nonlinear systems will be analysed:

$\star$ System I (considered in [64]):

$$d[k] = 0.6x[k] + 1.2x^2[k-1] + 0.8x[k-1]x[k-2] + \nu[k] \tag{20}$$

⋆ System II (considered in [65, 66]):

$$d[k] = x[k-2] + 0.08x^2[k-2] - 0.04x^3[k-1] + \nu[k] \tag{21}$$

⋆ System III (considered in [64]):

$$\begin{aligned} d[k] \quad = \quad & 0.3528x[k] + 0.2393x[k-1] + 0.1199x[k-2] \\ & -0.0025x[k-3] - 0.1248x[k-4] + 0.3461x^2[k] \\ & +0.2923x[k]x[k-1] + 0.2312x[k]x[k-2] \\ & +0.2434x^2[k-1] + 0.1886x[k-1]x[k-2] \\ & +0.1644x[k]x[k-3] + 0.0936x[k]x[k-4] \\ & +0.1291x[k-1]x[k-3] + 0.0664x[k-1]x[k-4] \\ & +0.1413x^2[k-2] + 0.0905x[k-2]x[k-3] \\ & +0.0376x[k-2]x[k-4] + 0.0498x^2[k-3] \\ & +0.0078x[k-3]x[k-4] - 0.0222x^2[k-4] + \nu[k]. \end{aligned} \tag{22}$$

Unless stated otherwise, the proposed GA method employed a mutation rate of 0.1. As stated before, a candidate individual to a population (a candidate solution) is considered unfeasible if the cardinality of the estimated active basis set is larger than the adjustable parameter $R_{\max}$. Due to the fact that the nonlinear system parameters are not known, it is expected that the choice of $R_{\max}$ guarantees a sufficient safety margin. For example, System II (see Eq. (21)) presents three active kernels. In this case, a successful identification procedure should employ a parameter $R_{\max}$ larger than 3. Henceforth, otherwise stated on the contrary, one assumes that the chromosome maximum size is 55 and the constraint concerning to number maximum of kernels ranges from 8 to 32 being for the Systems I and II $R_{\max} \in \{8, 9, \ldots, 15\}$ and for the System III $R_{\max} \in \{25, 26, \ldots, 32\}$. The following sections evaluate the impact of several identifi-

cation procedure parameters.

4.1. Comparison between CHOB versus RCH strategies

Tables 2 and 3 present, for Systems I and II (respectively), the total execution time of the GA with the Random Constructive Heuristic (RCH) in order to run the GA-based identification procedure with the following parameters: 100 generations, $N = 4$, $M = 3$, and $L = 1000$ measurements. In the case of System III, the total execution time is 2s for all $R_{\max}$ values. Note that the SNR parameter does not have a significant impact on the average execution time. Also, it is important to say the smaller the $R_{\max}$ value, the higher the execution average time because the constraint gets tighter. This is due to the time required to address unfeasible candidates, which occurs frequently. Employing the CHOB for a 100 Monte-Carlo trials, each using 100 GA generations, always required less than one second for all configurations. Tables 2 and 3 present the total execution time for Systems I and II, required by RCH. These results show that the CHOB requires less computational burden due to its ability to find feasible solutions with less computational effort than the RCH.

Table 2: Total execution time (with 100 independent Monte Carlo trials) for the GA-based identification with RCH strategy for System I.

$R_{\max}$	SNR = 10 dB	SNR = 15 dB	SNR = 20 dB
15	00m17s	00m17s	00m17s
14	00m51s	00m50s	00m50s
13	02m47s	02m44s	02m30s
12	09m35s	09m45s	09m25s
11	10m08s	10m01s	10m05s
10	10m14s	11m00s	10m30s
9	12m18s	12m32s	10m05s
8	46m24s	53m59s	45m13s

Due to lack of space, we opted not to include all the results values (fitness, chromosome, number of active kernels, and coefficients) obtained for each $R_{\max}$ and SNR case test. The full set of results can be obtained through the GitLab repository for this work [67]. In addition, in the set of results that follows, we opted to describe the best

Table 3: Total execution time (with 100 independent Monte Carlo trials) for the GA-based identification with RCH strategy for System II.

$R_{\max}$	SNR = 10 dB	SNR = 15 dB	SNR = 20 dB
15	00m22s	00m18s	00m20s
14	00m55s	00m53s	01m11s
13	02m51s	02m46s	03m21s
12	10m01s	10m00s	10m01s
11	10m06s	10m12s	10m01s
10	10m04s	10m36s	11m04s
9	11m36s	12m22s	12m41s
8	46m50s	52m34s	43m12s

system estimates in terms of different values of $R_{\max}$ and SNR. We chose to employ different values for each estimate on purpose, in order to show that a specific favourable configuration was not adopted. In doing so, we aim to show that the model is robust to diverse configurations.

The best estimates of System I for both CHOB and RCH strategies for $R_{\max} = 8$ and under an SNR of 10 dB (with the remaining parameters assuming the same values that generated Table 2) after running the proposed GA whilst varying the number of generations between $[10, 20, \cdots, 100]$ are:

$$\begin{aligned} y_{\mathrm{I}}^{(\mathrm{RCH})}(k) \quad = \quad & \mathbf{1.2252}x^2[k-1] + \mathbf{0.7775}x[k-1]x[k-2] \\ & -0.0111x^2[k-2] + 0.1212x[k]x^2[k-3] \\ & +0.1978x[k]x^2[k-4] - 0.0076x^2[k-1]x[k-4] \\ & +0.0075x^2[k-2]x[k-4] + 0.0079x^3[k-3], \end{aligned} \tag{23}$$

$$\begin{aligned} y_{\mathrm{I}}^{(\mathrm{CHOB})}(k) \quad = \quad & \mathbf{0.5692}x[k] + 0.0091x^2[k] + 0.0284x[k]x[k-2] \\ & -0.0268x[k]x[k-4] + \mathbf{1.1831}x^2[k-1] \\ & +\mathbf{0.8092}x[k-1]x[k-2] + 0.0292x[k-1]x^2[k-2] \\ & +0.0399x[k-2]x^2[k-3], \end{aligned} \tag{24}$$

where the proposed CHOB approach has successfully identified the three active kernels (indicated in boldface) in contrast to the RCH that identified, in the best case, two active kernels. It is important to draw attention to the fact that the higher the $R_{\max}$ value is, the easier it is the identification of the active kernels. This is justified by the smaller number of possible combinatorial solutions.

Tables 4 and 5 present a summary of the computational experiments for System I with the number of generations ranging from 10 to 100 and the same parameters for solutions in Equations (23) and (24). For each of these generation tests a hundred (100) resets were performed. In addition, the best solution was recorded. These tables present the best solution found for each number of GA generation ($\#gen$), where vector $\mathcal{J}$ contains the indices of the active kernels present in the respective best estimate that was recorded and vector $\hat{\boldsymbol{w}}$ contains the coefficients found for each corresponding index. The last line shows the best solution ($S^{\star}$) which refers to the systems considered (Eq. (20)). For each solution, the active kernels found that match with the ones in the best solution are underlined, as well as their respective weights values.

From the results presented in Tables 4 and 5, it is possible to observe that the CHOB is more efficient than RCH, not only in terms of computational time but also in identifying the active kernels. The RCH in some cases has identified only one kernel. In other cases, two kernels have been identified, in contrast with CHOB, which correctly has found the three correct active kernels for all test scenarios. The solution in boldface in both tables (4 and 5) represent solutions with the best fitness and they are depicted in Equations (23) and (24). Due to the random nature of the algorithm, the times described in Table 4 do not grow linearly with the number of generations. This is due to the number of unfeasible solutions that are generated and the respective amount of time that is required to address them.

The same case study was performed for the identification of System II, using $R_{\max} = 10$ and an SNR of 15 dB, with the same parameters of Table 3. The best

Table 4: Best solutions found when running the GA for System I with: RCH, resets = 100, SNR (dB) = 10, $R_{\max} = 8$.

#gen	Active kernels ($\mathcal{J}$) Coefficients ($\hat{w}$)	#gen	Active kernels ($\mathcal{J}$) Coefficients ($\hat{w}$)
10	$\mathcal{J}$ = [2 5 <u>10</u> <u>11</u> 12 21 24 47] $\hat{w}$ = [0.0017 -0.0123 <u>1.2315</u> <u>0.7353</u> -0.0297 -0.0074 -0.0336 0.0394] *time:* 53m31s *fitness:* 0.4184	60	$\mathcal{J}$ = [1 <u>10</u> 28 37 40 47 51 53] $\hat{w}$ = [0.0065 <u>1.2134</u> 0.0957 -0.0578 -0.0009 -0.0205 0.0014 -0.0276] *time:* 64m08s *fitness:* 0.2416
20	$\mathcal{J}$ = [2 <u>10</u> 18 22 32 41 44 48] $\hat{w}$ = [0.0167 <u>1.2055</u> 0.0084 -0.0549 0.1912 0.1058 0.0083 0.0398] *time:* 44m34s *fitness:* 0.2555	70	$\mathcal{J}$ = [5 6 <u>10</u> 17 27 32 33 48] $\hat{w}$ = [-0.0018 -0.0029 <u>1.1964</u> -0.0329 0.0004 0.1955 0.0579 0.0348] *time:* 47m48s *fitness:* 0.2519
30	$\mathcal{J}$ = [1 <u>10</u> <u>11</u> 13 22 25 27 49] $\hat{w}$ = [-0.0061 <u>1.1939</u> <u>0.7944</u> 0.0047 0.0089 0.2202 -0.0167 0.0517] *time:* 36m21s *fitness:* 0.4917	80	$\mathcal{J}$ = [<u>10</u> 16 20 28 45 46 47 52] $\hat{w}$ = [<u>1.1883</u> -0.0002 0.1239 0.0481 0.0137 -0.0603 -0.0323 0.0046] *time:* 46m13s *fitness:* 0.2594
40	$\mathcal{J}$ = [2 <u>10</u> <u>11</u> 15 22 41 46 52] $\hat{w}$ = [0.0427 <u>1.1980</u> <u>0.7372</u> 0.0314 -0.0342 -0.0098 -0.0088 -0.0053] *time:* 51m45s *fitness:* 0.4071	**90**	$\mathcal{J}$ = [**10 11** 14 32 34 38 47 51] $\hat{w}$ = [**1.2252 0.7775** -0.0111 0.1212 0.1977 -0.0076 0.0075 0.0078] *time:* 58m05s *fitness:* **0.5150**
50	$\mathcal{J}$ = [<u>0</u> 9 <u>10</u> 25 31 32 41 42] $\hat{w}$ = [<u>0.6042</u> 0.0128 <u>1.2125</u> -0.0673 -0.0171 0.0077 0.0826 -0.0059] *time:* 70m08s *fitness:* 0.3328	100	$\mathcal{J}$ = [2 <u>10</u> 21 25 34 41 46] $\hat{w}$ = [0.0372 <u>1.2039</u> 0.0171 0.0589 0.2190 0.0520 -0.0748] *time:* 46m24s *fitness:* 0.2540
$S^\star$	$\mathcal{J}^\star$ **= [0 10 11]** $\hat{w}^\star$ **= [0.6 1.2 0.8]**		

estimates for both strategies can be described as:

$$
\begin{aligned}
y_{\mathrm{II}}^{(\mathrm{RCH})}(k) \;=\; & \mathbf{1.0021}x[k-2] + 0.0065x[k]x[k-2] \\
& -0.0074x[k-1]x[k-3] + \mathbf{0.0861}x^2[k-2] \\
& -0.0015x[k-2]x[k-4] + 0.0112x[k]x[k-1]x[k-4] \\
& +0.0029x[k]x^2[k-2] + 0.0108x[k]x[k-2]x[k-3] \\
& -\mathbf{0.0395}x^3[k-1] - 0.0055x[k-1]x^2[k-2],
\end{aligned}
\tag{25}
$$

Table 5: Best solutions found when running the GA for System I with: CHOB, resets = 100, SNR (dB) = 10, $R_{\max} = 8$.

#gen	Active kernels ($\mathcal{J}$) Coefficients ($\hat{w}$)	#gen	Active kernels ($\mathcal{J}$) Coefficients ($\hat{w}$)
10	$\mathcal{J}$ = [0 10 11 25 35 38 46 50] $\hat{w}$ = [0.5623 1.1855 0.8190 -0.0193 -0.0050 -0.0079 0.0117 0.0381] *time:* 00m00s *fitness:* 0.5971	60	$\mathcal{J}$ = [0 5 10 11 12 25 33 45] $\hat{w}$ = [0.6371 0.0303 1.2035 0.8056 -0.0288 -0.0347 -0.0236 0.0029] *time:* 00m00s *fitness:* 0.5945
20	$\mathcal{J}$ = [0 2 4 10 11 18 25 48] $\hat{w}$ = [0.6116 -0.0183 -0.0293 1.2233 0.8051 0.0668 -0.0184 0.0134] *time:* 00m00s *fitness:* 0.5951	**70**	$\mathcal{J}$ = [**0** 5 7 9 **10 11** 39 48] $\hat{w}$ = [**0.5692** 0.0091 0.0284 -0.0268 **1.1831 0.8092** 0.0292 0.0399] *time:* 00m00s *fitness:* **0.5972**
30	$\mathcal{J}$ = [0 10 11 16 19 40 41 51] $\hat{w}$ = [0.5590 1.1899 0.7724 0.0172 0.0158 0.0458 -0.0059 0.0193] *time:* 00m00s *fitness:* 0.5961	80	$\mathcal{J}$ = [0 3 10 11 21 43 52 53] $\hat{w}$ = [0.6185 0.0567 1.2129 0.7875 -0.0132 0.0389 0.0225 -0.0144] *time:* 00m00s *fitness:* 0.5945
40	$\mathcal{J}$ = [0 6 10 11 16 32 37 54] $\hat{w}$ = [0.5896 -0.0681 1.2142 0.8417 0.0335 0.0199 -0.0008 -0.0080] *time:* 00m00s *fitness:* 0.5962	90	$\mathcal{J}$ = [0 2 3 5 10 11 29 54] $\hat{w}$ = [0.6154 0.0050 -0.0409 0.0350 1.1959 0.7803 -0.0128 -0.0094] *time:* 00m00s *fitness:* 0.5945
50	$\mathcal{J}$ = [0 2 10 11 13 26 40 49] $\hat{w}$ = [0.5984 -0.0363 1.2058 0.7526 -0.0350 0.0206 0.0333 0.0483] *time:* 00m00s *fitness:* 0.5958	100	$\mathcal{J}$ = [0 4 10 11 20 23 52 54] $\hat{w}$ = [0.5226 -0.1020 1.1944 0.8136 0.0094 -0.0220 -0.0074 0.0292] *time:* 00m00s fitness: 0.5966
$S^\star$	$\mathcal{J}^\star$ **= [0 10 11]** $\hat{w}^\star$ **= [0.6 1.2 0.8]**		

$$
\begin{aligned}
y_{\mathrm{II}}^{(\mathrm{CHOB})}(k) \;=\; & \mathbf{1.0038}x[k-2] - 0.0110x[k]x[k-2] \\
& -0.0076x[k]x[k-3] + \mathbf{0.0824}x^2[k-2] \\
& -0.0164x[k-2]x[k-3] + 0.0018x^2[k]x[k-2] \\
& -0.0081x[k]x[k-2]x[k-4] - \mathbf{0.0368}x^3[k-1] \\
& +0.0070x^2[k-1]x[k-2] - 0.0008x^3[k-2],
\end{aligned}
\tag{26}
$$

where the RCH strategy sometimes identify two of the three kernels. In the other cases all active basis set were identified. The CHOB correctly identify the active basis sets for all tests. Both strategies assign small weights to the inactive basis set.

Tables 6 and 7 present a summary of the computational experiments for System II with number of generations ranging from 10 to 100 and the same parameters for the

solutions in Equations (25) and (26). These tables present the best solution found for each number of GA generations (#*gen*) and with a hundred (100) resets respectively performed. The results are shown following the same pattern as Tables 4 and 5.

Table 6: Best solutions found when running the GA for System II with: RCH, resets = 100, SNR (dB) = 15, $R_{\max} = 10$.

#gen	**Active kernels ($\mathcal{J}$)** **Coefficients ($\hat{w}$)**	**#gen**	**Active kernels ($\mathcal{J}$)** **Coefficients ($\hat{w}$)**
10	$\mathcal{J}$ =[2 3 5 10 34 35 39 41 45 50] $\hat{w}$ = [1.0100 0.0054 0.0298 0.0126 -0.0025 -0.0404 0.0048 0.0080 -0.0085 0.0041] *time:* 10m41s fitness: 0.9503	60	$\mathcal{J}$ =[1 2 3 7 11 14 20 25 35 53] $\hat{w}$ = -0.0085 0.9971 0.0104 0.0010 -0.0014 0.0805 0.0031 -0.0028 -0.0395 0.0025] *time:* 10m28s fitness: 0.9676
20	$\mathcal{J}$ = [2 6 10 11 12 17 18 34 35 52] $\hat{w}$ = [1.0010 -0.0019 0.0194 0.0096 -0.0113 0.0139 0.0032 -0.0053 -0.0414 0.0063] *time:* 10m03s fitness: 0.9512	70	$\mathcal{J}$ =[2 5 14 15 18 25 30 35 40 44] $\hat{w}$ = 0.9924 -0.0059 0.0838 0.0051 0.0017 -0.0022 0.0032 -0.0435 -0.0020 0.0081] *time:* 10m34s fitness: 0.9677
30	$\mathcal{J}$ = [**2** 7 12 **14** 16 28 29 30 **35** 39] $\hat{w}$ = [**1.0021** 0.0065 -0.0074 **0.0861** -0.0015 0.0112 0.0029 0.0108 **-0.0395** -0.0056] *time:* 10m07s fitness: **0.9678**	80	$\mathcal{J}$ =[2 8 10 14 18 33 35 42 45 52] $\hat{w}$ =[0.9824 -0.0016 0.0056 0.0836 -0.0033 -0.0042 -0.0369 0.0021 0.0027 -0.0052] *time:* 10m26s fitness: 0.9678
40	$\mathcal{J}$ = [1 2 4 14 23 29 31 39 41 54] $\hat{w}$ = [-0.1337 1.0023 -0.0061 0.0804 0.0002 -0.0023 0.0088 0.0068 -0.0006 0.0032] *time:* 10m07s fitness: 0.9574	90	$\mathcal{J}$ = [1 2 8 13 14 19 29 40 48] $\hat{w}$ = [-0.1300 1.0043 0.0042 -0.0054 0.0793 -0.0074 0.0049 0.0060 -0.0013] *time:* 10m22s fitness: 0.9580
50	$\mathcal{J}$ = [2 4 14 22 30 33 35 41 42 45] $\hat{w}$ = [1.0011 0.0103 0.0769 -0.0057 -0.0093 0.0026 -0.0402 -0.0021 -0.0028 -0.0006] *time:* 10m34s fitness: 0.9676	100	$\mathcal{J}$ = [2 3 4 10 14 17 25 35 41 43] $\hat{w}$ = [1.0022 -0.0070 -0.0051 -0.0036 0.0819 0.0009 -0.0005 -0.0389 -0.0042 0.0059] *time:* 10m36s fitness: 0.9674
$S^{\star}$	$\mathcal{J}^{\star}$ **= [2 14 35]** $\hat{w}$ **= [1.0 0.08 -0.04]**		

For System II identification the best solutions are exhibited in Tables 6 and 7 and the results show that CHOB again demands less computational effort than RCH and that it successfully has identified all active kernels in all case studies.

From the identification point of view, System III is more challenging, since it has many small-magnitude coefficients. Eqs. (27)-(28) present the best estimated Volterra system using $R_{\max} = 25$ and under an SNR of 10 dB, for RCH (Eq. (27)) and CHOB (Eq. (28)). Note that both strategies correctly identify most active kernels. Tables that present more details about algorithmic performance (similar to Tables 4-5) are not presented, since it will require a large amount of information. This is due to the higher

Table 7: Best solutions found when running the GA for System II with CHOB, resets = 100, SNR (dB) = 15, $R_{\max} = 10$.

#gen	Active kernels ($\mathcal{J}$) Coefficients ($\hat{w}$)	#gen	Active kernels ($\mathcal{J}$) Coefficients ($\hat{w}$)
10	$\mathcal{J}$ =[2 5 14 22 28 35 37 45 48 50] $\hat{w}$ = [1.0007 0.0085 0.0759 -0.0094 -0.0013 -0.0412 -0.0008 0.0019 0.0023 -0.0023] *time:* 00m00s fitness: 0.9666	60	$\mathcal{J}$ =[2 3 14 27 33 35 36 40 44 49] $\hat{w}$ = [0.9970 -0.0026 0.0820 -0.0020 -0.0104 -0.0401 0.0056 0.0005 0.0059 -0.0102] *time:* 00m00s fitness: 0.9666
20	$\mathcal{J}$ = [2 6 14 17 34 35 43 45 47 48] $\hat{w}$ = [1.0301 0.0120 0.0787 0.0050 0.0046 -0.0414 -0.0058 -0.0033 0.0031 -0.0108] *time:* 00m00s fitness: 0.9668	70	$\mathcal{J}$ = [0 2 3 4 14 21 23 26 35 45] $\hat{w}$ = [-0.0011 0.9785 0.0145 -0.0124 0.0812 -0.0074 -0.0063 0.0120 -0.0401 0.0038] *time:* 00m00s fitness: 0.9669
30	$\mathcal{J}$ = [**2** 7 8 **14** 15 22 31 **35** 36 45] $\hat{w}$ = [**1.0038** -0.0110 -0.0076 **0.0824** -0.0164 0.0018 -0.0081 **-0.0368** 0.0070 -0.0008] *time:* 00m00s fitness: **0.9670**	80	$\mathcal{J}$ = [2 14 15 35 36 40 41 45 46 48] $\hat{w}$ = [0.9913 0.0763 0.0137 -0.0409 -0.0081 0.0153 0.0083 0.0037 0.0053 0.0063] *time:* 00m00s fitness: 0.9669
40	$\mathcal{J}$ = [2 7 9 14 23 32 35 36 48 52] $\hat{w}$ = [1.0042 0.0031 -0.0096 0.0753 0.0087 0.0103 -0.0393 -0.0023 -0.0007 0.0010] *time:* 00m00s fitness: 0.9669	90	$\mathcal{J}$ = [0 2 4 5 14 16 35 36 47 51] $\hat{w}$ = [-0.0000 1.0107 0.0118 -0.0036 0.0830 0.0129 -0.0401 -0.0072 -0.0062 0.0014] *time:* 00m00s fitness: 0.9666
50	$\mathcal{J}$ = [2 12 14 22 31 34 35 36 46 50] $\hat{w}$ = [1.0140 -0.0073 0.0771 0.0030 -0.0074 -0.0025 -0.0391 -0.0020 0.0028 -0.0065] *time:* 00m00s fitness: 0.9666	100	$\mathcal{J}$ = [2 7 11 14 22 26 35 36 42 47] $\hat{w}$ = [0.9970 -0.0123 -0.0119 0.0772 -0.0036 0.0053 -0.0397 0.0010 -0.0095 0.0039] *time:* 00m00s fitness: 0.9668
$S^\star$	$\mathcal{J}^\star$ **= [2 14 35]** $\hat{w}$ **= [1.0 0.08 -0.04]**		

complexity of System III.

$$
\begin{aligned}
y_{\mathrm{III}}^{(\mathrm{RCH})}(k) \quad = \quad & \mathbf{0.3743}x[k] + \mathbf{0.2468}x[k-1] + \mathbf{0.1092}x[k-2] \\
& -\mathbf{0.1152}x[k-4] + \mathbf{0.3463}x^2[k] + \mathbf{0.2881}x[k]x[k-1] \\
& +\mathbf{0.2253}x[k]x[k-2] + \mathbf{0.1594}x[k]x[k-3] \\
& +\mathbf{0.0867}x[k]x[k-4] + \mathbf{0.2521}x^2[k-1] \\
& +\mathbf{0.1635}x[k-1]x[k-2] + \mathbf{0.1311}x[k-1]x[k-3] \\
& +\mathbf{0.0540}x[k-1]x[k-4] + \mathbf{0.1368}x^2[k-2] \\
& +\mathbf{0.0979}x[k-2]x[k-3] + \mathbf{0.0326}x^2[k-3] \\
& +0.0139x^2[k]x[k-1] + 0.0072x^2[k]x[k-2] \\
& -0.0060x^2[k]x[k-4] - 0.0065x[k]x[k-1]x[k-4] \\
& -0.0186x[k]x^2[k-3] + 0.0106x[k-1]x^2[k-2] \\
& -0.0278x[k-1]x^2[k-3] - 0.0110x[k-2]x^2[k-4] \\
& -0.0186x^2[k-3]x[k-4],
\end{aligned}
\tag{27}
$$

$$
\begin{aligned}
y_{\mathrm{III}}^{(\mathrm{CHOB})}(k) \;=\; & \mathbf{0.3918}x[k]+\mathbf{0.2414}x[k-1]+\mathbf{0.1461}x[k-2] \\
& +\mathbf{0.0265}x[k-3]-\mathbf{0.1435}x[k-4]+\mathbf{0.3413}x^2[k] \\
& +\mathbf{0.3038}x[k]x[k-1]+\mathbf{0.2509}x[k]x[k-2] \\
& +\mathbf{0.1759}x[k]x[k-3]+\mathbf{0.0724}x[k]x[k-4] \\
& +\mathbf{0.2303}x^2[k-1]+\mathbf{0.1814}x[k-1]x[k-2] \\
& +\mathbf{0.1367}x[k-1]x[k-3]+\mathbf{0.0639}x[k-1]x[k-4] \\
& +\mathbf{0.1395}x^2[k-2]+\mathbf{0.0967}x[k-2]x[k-3] \\
& +\mathbf{0.0590}x^2[k-3]+0.0105x^2[k]x[k-4] \\
& -0.0123x[k]x[k-2]x[k-3]-0.0200x[k]x^2[k-3] \\
& +0.0015x^2[k-1]x[k-2]-0.0029x[k-1]x^2[k-2] \\
& +0.0137x^2[k-2]x[k-4]-0.0128x[k-3]x^2[k-4].
\end{aligned}
\tag{28}
$$

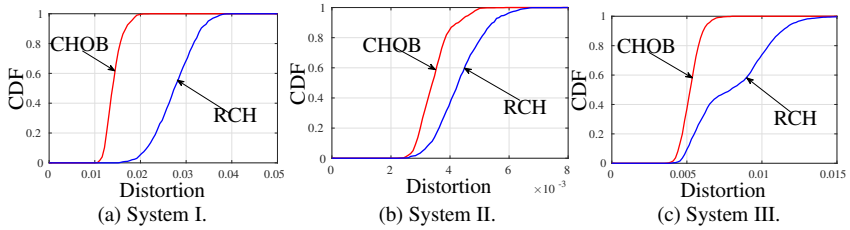
Intensive tests were performed, varying the two parameters presented in Tables 2 and 3. Our data indicates, when considering the systems in study, that the CHBO always finds good quality solutions whilst requiring less time than RCH.

Furthermore, we also analyzed the impact of using CHOB method, by comparing it to the naive RCH strategy in the identification of the three considered systems. The employed configuration uses a SNR of 15 dB, L = 1000, and $R_{\max} = 15$ (for Systems I and II) and $R_{\max} = 32$ (for System III). Fig. 4 depicts the cumulative density function (CDF) of the distortion, evaluated through 1000 independent Monte Carlo trials. Such a distortion is evaluated through Eq. (29).

$$
\mathrm{Distortion}[\hat{\boldsymbol{w}}] \triangleq \|\hat{\boldsymbol{w}}-\boldsymbol{w}^{\star}\|. \tag{29}
$$

The CDF of an estimator distortion indicates a more accurate estimate the faster

it approaches unity. Note that the proposed CHOB method outperforms the RCH in all considered scenarios. Due to this fact, only the CHOB method will be assessed henceforth.

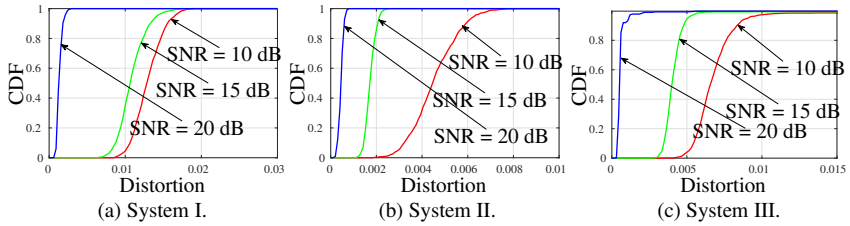


(a) System I. (b) System II. (c) System III.

Figure 4: Cumulative density function of the distortion of the CHOB (in red) and RCH (in blue) methods.

4.2. Impact of the SNR

In this scenario, three different SNR values (in dB) are considered: 10, 15 and 20. For all systems, the mutation rate is 0.1, the crossover rate is 0.3, $L = 1000$, $R_{\max}$ is 12 (for Systems I and II) and 30 (for System III). Fig. 5 presents the CDF for the three identification problems, where one can confirm that the higher the SNR value, the better is the system identification procedure.

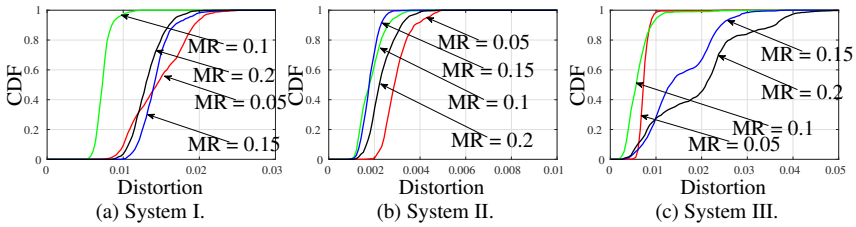


(a) System I. (b) System II. (c) System III.

Figure 5: Cumulative density function of the distortion of the CHOB method for different SNR values (in dB).

4.3. Impact of the Mutation Rate

In this experiment, the mutation rate (MR) is varied in the set $\{0.05, 0.1, 0.15, 0.2\}$. For all considered systems, the SNR is 10 dB, the crossover rate is 0.3, $L = 1000$, $R_{\max}$



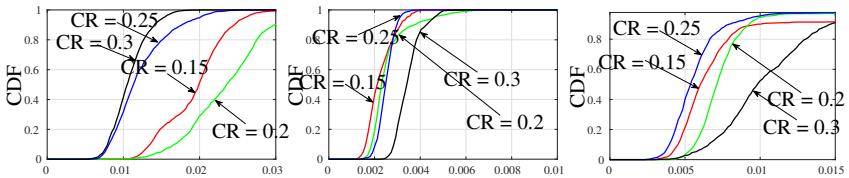
(a) System I. (b) System II. (c) System III.

Figure 6: Cumulative density function of the identification procedure distortion for different mutation rate values.

is 14 (for System I), 15 (for System II) and 28 (for System III). Fig. 6 presents the results. Such a figure allows one to conclude that the choice of a mutation rate of 0.1 tends to be competitive in the different scenarios. These results show that it is necessary to maintain diversity of the population during the generations. However, inserting a lot of diversity can lead to poor quality solutions.

4.4. Impact of the Crossover Rate

In this experiment, the crossover rate (CR) is varied in the set $\{0.15, 0.2, 0.25, 0.3\}$. For all considered systems, the SNR is 15 dB, the mutation rate is 0.1, $L = 1000$, and $R_{\max}$ is the same of Experiment C. Fig. 7 presents the results achieved. One can conclude that the choice of a crossover rate of approximately 0.25 tends to be competitive in the different scenarios. Furthermore, it should be noted that in practice it is not possible to have an accurate prior knowledge about its optimum value.



(a) System I. (b) System II. (c) System III.

Figure 7: Cumulative density function of the identification procedure distortion for different crossover rate values.

4.5. Impact of the R_{max}

In this experiment, the $R_{\max}$ parameter varies in the set $\{12, 13, 14, 15\}$. For this experiment, Systems I and II are considered. Due to its size, System III is not considered in this scenario. The SNR is 10 dB, the mutation rate is 0.1, $L = 500$ and the crossover rate is 0.3. Table 8 presents the mean squared distortion, for 1000 independent Monte Carlo trials. Note that in average the estimation of System I presents worse results than that of System II. Furthermore, increasing $R_{\max}$ does not necessarily imply more accuracy in the identification process, as expected.

Table 8: Mean squared distortion for different values of $R_{\max}$ for Systems I and II.

$R_{\max}$	System I	System II
12	0.0198	0.0049
13	0.0272	0.0095
14	0.0313	0.0035
15	0.0239	0.0053

4.6. Comparison with Alternative Approaches

This section assesses the performance of the proposed algorithm when compared against: (i) Recursive Least Squares (RLS) with forgetting factor parameter λ [68] and Least Squares (LS, see Eq. (14)). Consider the identification of System II with L = 1000. The advanced genetic algorithm solution is implemented with $R_{\max} = 15$, SNR = 10 dB, a mutation rate of 0.1 and a crossover of 0.3. Table 9 shows that the LS algorithm has unpredictable results, therefore, it will not be considered in the comparison.

Figures 8 and 9 show smoothed density distribution and boxplots of the root mean square error (RMSE) of the remaining five algorithms, respectively. In both images, it can be clearly seen that both CHOB and RCH algorithms perform much better than RLS versions with $\lambda \in \{0.95, 0.98\}$. RLS with $\lambda = 0.99$, although, seems to have a good performance. Figure 10 show the same boxplot, with only these three algorithms, and it shows that RLS has the worst performance.

Table 9: Mean, standard deviation and maximum root mean square error of algorithms CHOB, RCH, RLS (with different λ factors) and LS, for 1000 independent Monte Carlo trials.

Algorithm	**Mean**	**St. Dev.**	**Max**
CHOB	1.01	0.04	1.15
RCH	1.01	0.04	1.15
RLS ($\lambda = 99$)	1.13	0.04	1.28
RLS ($\lambda = 98$)	2.11	0.26	3.03
RLS ($\lambda = 95$)	2.16	0.50	4.87
LS	19,701,725.11	587,906,028.36	18,583,390,122.44

Analysis of variance (ANOVA) has been run on these three RMSE sets. The adopted null hypothesis states that the means of the groups are all the same, whilst alternative hypothesis says that at least one group has a different average.

The returned F-value is 3009, leading to a p-value smaller than $2e^{-16}$. Consequently, one has strong statistical evidence that at least one set has a different mean. To find out which one, we compute Tukey HSD. Table 10 presents the results obtained. Adjusted p-values show clearly, on any significance level, that RLS is different from both CHOB and RCH, while these do not have much statistical significant difference between them. Such an equivalence between CHOB and RCH was somewhat expected since both converge at some point, however, CHOB converges faster.

Table 10: Tukey HSD test for the different pairs of algorithms.

Algorithms	Difference	$p-$ value
RCH-CHOB	0.0002	0.9904
RLS 99-CHOB	0.1174	0.0000
RLS 99-RCH	0.1172	0.0000

5. Final Remarks

This work has focused on developing a solution approach to identify in an efficient manner Volterra Systems. The proposed solution framework is based on genetic algorithms (GA) concepts, enhanced by two distinct heuristics and a hierarchical structured population.

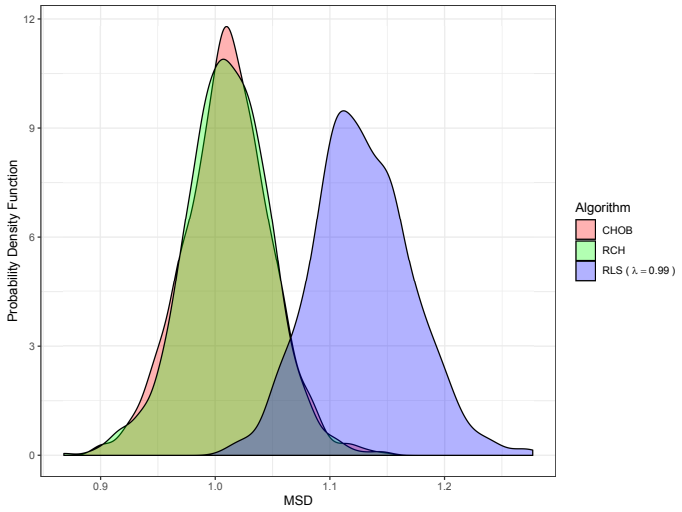


Figure 8: Estimated probability density function of the RMSE of algorithms CHOB, RCH and RLS (with $\lambda = 0.99$).

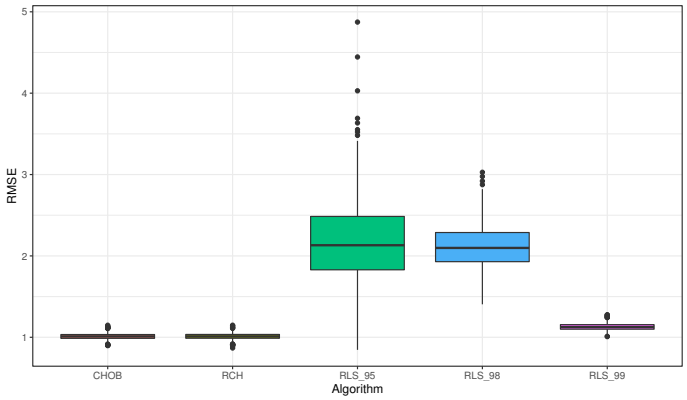


Figure 9: Boxplot of the RMSE of the CHOB, RCH, and RLS algorithms. The RLS algorithm was tested with $\lambda \in \{0.99, 0.98, 0.95\}$.

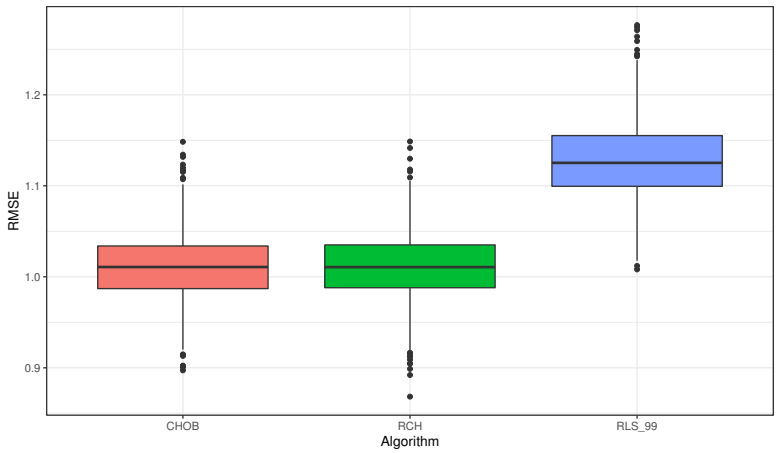


Figure 10: Boxplot of the RMSE of the CHOB, RCH, and RLS algorithms. The RLS algorithm was tested with $\lambda = 0.99$.

Some features presented in this work are: (i) an efficient methodology for identification of Volterra series based on genetic algorithms; (ii) the introduction of two constructive heuristics; (iii) the reduction of the overall computational burden by the usage of a hierarchical evolutionary technique; (iv) robustness against noise; (v) lack of necessity for a judicious parameters adjustment process and (vi) a repository containing several experiments, performed with three systems used in the literature, which evaluates the impact of: (i) the GA parameters; (ii) the levels of signal-to-noise; (iii) the number of samples; (iv) the maximum limits of active kernels; and (v) the two proposed heuristics. The tests concerning the comparison of the two proposed heuristics showed that the CHOB requires less time to find quality solutions than the RCH approach. Beyond time, CHOB is also able to find better quality solutions. Experiments regarding the signal-to-noise levels were fulfilled and the solution methodology was capable of achieving good results even in a scenario with high-noise in a reasonable computational time. Evidences are given about the impact of calibrating GA parameters as well. The results obtained demonstrate the effectiveness of the proposed methodology in producing high-quality solutions for the identification of Volterra systems, in an acceptable

computational time.

Acknowledgments

This work has been supported by CNPq, FAPERJ and CAPES.

References

[1] L. C. Resende, D. B. Haddad, and M. R. Petraglia, "A variable step-size NLMS algorithm with adaptive coefficient vector reusing," in *2018 IEEE International Conference on Electro/Information Technology*, May 2018, pp. 1–6.

[2] N. Liu, C. Ju, and X. Chen, "Nonlinear ISI cancellation in VSSB Nyquist-SCM system with symbol pre-distortion," *Optics Communications*, vol. 338, pp. 492–495, 2015.

[3] C. Crespo-Cadenas, J. Reina-Tosina, M. J. Madero-Ayora, and J. Munoz-Cruzado, "A new approach to pruning Volterra models for power amplifiers," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2113–2120, April 2010.

[4] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1667–1676, Aug 2001.

[5] M. Faifer, C. Laurano, R. Ottoboni, M. Prioli, S. Toscani, and M. Zanoni, "Definition of simplified frequency-domain Volterra models with quasi-sinusoidal input," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 5, pp. 1652–1663, May 2018.

[6] W. Saabe, J. Sombrin, E. Ngoya, G. Soubercaze-Pun, and L. Lapierre, "Volterra-based modeling of traveling wave tube amplifier for system level simulation,"

in *2017 Eighteenth International Vacuum Electronics Conference (IVEC)*, April 2017, pp. 1–2.

[7] S. M. Serunjogi, M. A. Sanduleanu, and M. S. Rasras, "Volterra series based linearity analysis of a phase-modulated microwave photonic link," *Journal of Lightwave Technology*, vol. 36, no. 9, pp. 1537–1551, May 2018.

[8] J. T. Valliarampath and S. Sinha, "Designing linear pas at millimeter-wave frequencies using Volterra series analysis," *Canadian Journal of Electrical and Computer Engineering*, vol. 38, no. 3, pp. 232–237, Summer 2015.

[9] P. Śliwiński, A. Marconato, P. Wachel, and G. Birpoutsoukis, "Non-linear system modelling based on constrained Volterra series estimates," *IET Control Theory Applications*, vol. 11, no. 15, pp. 2623–2629, 2017.

[10] R. Lin and T.-Y. Ng, "Identification of Volterra kernels for improved predictions of nonlinear aeroelastic vibration responses and flutter," *Engineering Structures*, vol. 171, pp. 15–28, 2018.

[11] P. M. S. Burt and J. H. de Morais Goulart, "Efficient computation of bilinear approximations and Volterra models of nonlinear systems," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 804–816, Feb 2018.

[12] T. Kamalakis and G. Dede, "Nonlinear degradation of a visible-light communication link: A Volterra-series approach," *Optics Communications*, vol. 417, pp. 46–53, 2018.

[13] G. Feng, H. Li, J. Dong, and J. Zhang, "Face recognition based on Volterra kernels direct discriminant analysis and effective feature classification," *Information Sciences*, vol. 441, pp. 187–197, 2018.

[14] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlin-

ear operators with Volterra series," *IEEE Transactions on Circuits and Systems*, vol. 32, no. 11, pp. 1150–1161, Nov 1985.

[15] A. Carini and G. L. Sicuranza, "A new class of flann filters with application to nonlinear active noise control," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 1950–1954.

[16] I. W. Sandberg, "Expansions for nonlinear systems," *The Bell System Technical Journal*, vol. 61, no. 2, pp. 159–199, Feb 1982.

[17] Z. A. Khan, E. Zenteno, P. Handel, and M. Isaksson, "Multitone design for third order MIMO volterra kernels," in *2017 IEEE MTT-S International Microwave Symposium (IMS)*, June 2017, pp. 1553–1556.

[18] H. Shrimali, V. K. Sharma, J. N. Tripathi, and R. Malik, "Nonlinear modeling and analysis of buck converter using Volterra series," in *2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2017, pp. 222–226.

[19] T. Nguyen, J. E. Schutt-Aine, and Y. Chen, "Volterra kernels extraction from frequency-domain data for weakly non-linear circuit time-domain simulation," in *2017 IEEE Radio and Antenna Days of the Indian Ocean (RADIO)*, Sept 2017, pp. 1–2.

[20] S. Zhang, J. Zhang, and Y. Pang, "Pipelined set-membership approach to adaptive Volterra filtering," *Signal Processing*, vol. 129, pp. 195–203, 2016.

[21] H. Ying, M. Zhu, J. Zhang, X. Yi, Y. Song, and K. Qiu, "Sparse Volterra model based on optical single side-band NPAM-4 direct-detection system," *Optical Fiber Technology*, vol. 40, pp. 180–184, 2018.

[22] W. Du, M. Zhang, W. Ying, M. Perc, K. Tang, X. Cao, and D. Wu, "The net-

worked evolutionary algorithm: A network science perspective," *Applied Mathematics and Computation*, vol. 338, pp. 33–43, 2018.

[23] M. R. AlRashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 913–918, Aug 2009.

[24] C. Guerrero, I. Lera, and C. Juiz, "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures," *Future Generation Computer Systems*, vol. 97, pp. 131 – 144, 2019.

[25] R. Bruns, J. Dunkel, and N. Offel, "Learning of complex event processing rules with genetic programming," *Expert Systems with Applications*, vol. 129, pp. 186 – 199, 2019.

[26] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 381–399, June 2010.

[27] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature - PPSN VIII*, X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 832–842.

[28] C.-F. Wang and W.-X. Song, "A novel firefly algorithm based on gender difference and its convergence," *Applied Soft Computing*, vol. 80, pp. 107 – 124, 2019.

[29] G. Lindfield and J. Penny, "Chapter 7 - artificial bee and ant colony optimization," in *Introduction to Nature-Inspired Optimization*, G. Lindfield and J. Penny, Eds. Boston: Academic Press, 2017, pp. 119–140.

[30] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, Feb 2011.

[31] W. Li, E. Özcan, and R. John, "A learning automata-based multiobjective hyper-heuristic," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 59–73, Feb 2019.

[32] L. S. de Assis, J. F. V. González, F. L. Usberti, C. Lyra, C. Cavellucci, and F. J. V. Zuben, "Switch allocation problems in power distribution systems," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 246–253, Jan 2015.

[33] G. P. Liu and V. Kadirkamanathan, "Learning with multi-objective criteria," in *1995 Fourth International Conference on Artificial Neural Networks*, Jun 1995, pp. 53–58.

[34] ——, "Multiobjective criteria for neural network structure selection and identification of nonlinear systems using genetic algorithms," *IEE Proceedings - Control Theory and Applications*, vol. 146, no. 5, pp. 373–382, Sep 1999.

[35] L. Yao, "Genetic algorithm based identification of nonlinear systems by sparse Volterra filters," in *Emerging Technologies and Factory Automation, 1996. EFTA '96. Proceedings., 1996 IEEE Conference on*, vol. 1, Nov 1996, pp. 327–333 vol.1.

[36] ——, "Genetic algorithm based identification of nonlinear systems by sparse Volterra filters," *IEEE Transactions on Signal Processing*, vol. 47, no. 12, pp. 3433–3435, Dec 1999.

[37] A. Sierra, J. A. Macias, and F. Corbacho, "Evolution of functional link networks," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 54–65, Feb 2001.

[38] H. M. Abbas and M. M. Bayoumi, "Volterra system identification using adaptive genetic algorithms," *Applied Soft Computing*, vol. 5, no. 1, pp. 75–86, 2004.

[39] ——, "An adaptive evolutionary algorithm for Volterra system identification," *Pattern Recognition Letters*, vol. 26, no. 1, pp. 109–119, 2005.

[40] D. Korpi, M. Turunen, L. Anttila, and M. Valkama, "Modeling and cancellation of self-interference in full-duplex radio transceivers: Volterra series-based approach," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[41] R. A. d. Prado, F. d. R. Henriques, and D. B. Haddad, "Sparsity-aware distributed adaptive filtering algorithms for nonlinear system identification," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–8.

[42] L. Yao and C. c. Lin, "Identification of nonlinear systems by the genetic programming-based Volterra filter," *IET Signal Processing*, vol. 3, no. 2, pp. 93–105, March 2009.

[43] R. N. Braithwaite, "Digital predistortion of an RF power amplifier using a reduced Volterra series model with a memory polynomial estimator," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 10, pp. 3613–3623, Oct 2017.

[44] R. Kumar, K. Izui, Y. Masataka, and S. Nishiwaki, "Multilevel redundancy allocation optimization using hierarchical genetic algorithm," *IEEE Transactions on Reliability*, vol. 57, no. 4, pp. 650–661, Dec 2008.

[45] J. Prawin and A. R. M. Rao, "Nonlinear identification of mdof systems using Volterra series approximation," *Mechanical Systems and Signal Processing*, vol. 84, pp. 58–77, 2017.

[46] M. Rudko and D. Weiner, "Volterra systems with random inputs: A formalized

approach," *IEEE Transactions on Communications*, vol. 26, no. 2, pp. 217–227, Feb 1978.

[47] H. Enzinger, K. Freiberger, G. Kubin, and C. Vogel, "Fast time-domain Volterra filtering," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, Nov 2016, pp. 225–228.

[48] G. Tomlinson, G. Manson, and G. Lee, "A simple criterion for establishing an upper limit to the harmonic excitation level of the duffing oscillator using the Volterra series," *Journal of Sound and Vibration*, vol. 190, no. 5, pp. 751–762, 1996.

[49] L. M. Li and S. A. Billings, "Piecewise volterra modeling of the duffing oscillator in the frequency-domain," *Mechanical Systems and Signal Processing*, vol. 26, pp. 117–127, 2012.

[50] D. B. Haddad, M. R. Petraglia, and A. Petraglia, "A unified approach for sparsity-aware and maximum correntropy adaptive filters," in *24th European Signal Processing Conference (EUSIPCO)*, Aug 2016, pp. 170–174.

[51] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear estimation*. Prentice Hall, 2000, no. EPFL-BOOK-233814.

[52] D. B. Haddad, L. O. Nunes, W. A. Martins, L. W. P. Biscainho, and B. Lee, "Closed-form solutions for robust acoustic sensor localization," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct 2013, pp. 1–4.

[53] T. Strutz, *Data fitting and uncertainty: A practical introduction to weighted least squares and beyond*. Vieweg and Teubner, 2010.

[54] R. Alipour-Sarabi, Z. Nasiri-Gheidari, F. Tootoonchian, and H. Oraee, "Improved winding proposal for wound rotor resolver using genetic algorithm and winding

function approach," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1325–1334, Feb 2019.

[55] P. Salza and F. Ferrucci, "Speed up genetic algorithms in the cloud using software containers," *Future Generation Computer Systems*, vol. 92, pp. 276–289, 2019.

[56] R. He, B. Hu, W. Zheng, and X. Kong, "Robust principal component analysis based on maximum correntropy criterion," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1485–1494, June 2011.

[57] L. Resende, N. Siqueira, R. Pimenta, D. Haddad, and M. R Petraglia, "Lms algorithm with reuse of coefficients and robustness against impulsive noise," 10 2018.

[58] B. Chen and J. C. Principe, "Maximum correntropy estimation is a smoothed map estimation," *IEEE Signal Processing Letters*, vol. 19, no. 8, pp. 491–494, Aug 2012.

[59] L. Shi and Y. Lin, "Convex combination of adaptive filters under the maximum correntropy criterion in impulsive interference," *IEEE Signal Processing Letters*, vol. 21, no. 11, pp. 1385–1388, Nov 2014.

[60] A. Mendes, P. M. França, C. Lyra, C. Pissarra, and C. Cavellucci, "Capacitor placement in large-sized radial distribution networks." *IEE Proceedings Generation, Transmission and Distribution*, vol. 152, no. 4, pp. 496–502, July 2005.

[61] P. M. França, A. Mendes, and P. Moscato, "A memetic algorithm for the total tardiness single machine scheduling problem." *European Journal of Operational Research*, vol. 132, pp. 224–242, April 2001.

[62] A. Ladj, F. B. Tayeb, and C. Varnier, "Tailored genetic algorithm for scheduling jobs and predictive maintenance in a permutation flowshop," in *23rd IEEE Inter-*

national Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1, Sept 2018, pp. 524–531.

[63] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.

[64] X. Xia, J. Zhou, J. Xiao, and H. Xiao, "A novel identification method of volterra series in rotor-bearing system for fault diagnosis," *Mechanical systems and signal processing*, vol. 66, pp. 557–567, 2016.

[65] L. Lu and H. Zhao, "Adaptive volterra filter with continuous lp-norm using a logarithmic cost for nonlinear active noise control," *Journal of Sound and Vibration*, vol. 364, pp. 14–29, 2016.

[66] Z. Qizhi and J. Yongle, "Active noise hybrid feedforward/feedback control using neural network compensation," *Journal of vibration and acoustics*, vol. 124, no. 1, pp. 100–104, 2002.

[67] J. R. Paula Junior, L. S. Assis, L. D. T. J. Tarrataca, and D. B. Haddad, "Volterra Identification Code," https://gitlab.com/jurairr/volterra-series-public, 2018.

[68] P. S. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 2nd ed. Norwell, MA, USA: Kluwer Academic Publishers, 2002.