Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Arborescent Orthogonal Least Squares Regression for NARMAX-based Black-box Fitting

Stéphane J.P.S. Thunus[1], Julian Parker[2], and Stefan Weinzierl[1]
[1]Technische Universität Berlin – Audio Communication Group, Berlin, 10587, Germany
[2]Native Instruments, Berlin, 10997, Germany
Corresponding author: Stéphane Thunus (Stephane@Thunus.org).

ABSTRACT This paper proposes a linear algebra-based supervised machine learning algorithm for the symbolic representation of arbitrarily non-linear and recursive systems. It introduces multiple extensions to the algorithmic class of "Forward Orthogonal Least Squares Regressions" (FOrLSR), which performs dictionary-based sparse symbolic regressions. The regression, being only provided with the system's input and output, performs variable combinations and non-linear transformations from a given dictionary of analytic expressions and selects the optimal ones to represent the unknown system. This yields a "symbolic" system representation, having the minimum number of terms to enforce sparsity, while keeping the highest possible precision. The proposed algorithm restructures the FOrLSR to be in matrix form (for large scale GPU and BLAS-like optimizations), recursive (to reduce the complexity from quadratic in model length to linear) and allows regressors to be imposed (to include user expertise and perform tree-searches). Furthermore, the dictionary search is restructured into a breadth-first arborescence traversal kept sparse by four proposed theorems, corollaries and one pruning mechanism, while adding a validation procedure for the final model selection. The arborescence scans large search-space segments, significantly increasing the probability of finding an optimal system representation, while only computing a marginal fraction of the search-space. The regression and arborescence are solvers for arbitrarily determined linear equation systems which maximize sparsity in the solution vectors.

INDEX TERMS Forward Orthogonal Least Squares Regression, Model Structure Detection, NARMAX, Non-linear System Identification, Solution Sparsity Enforcing Linear Equation System Solver, Supervised Machine Learning For Parameter Estimation, Symbolic Regression.

I. INTRODUCTION

The world becoming increasingly reliant on data-driven decision-making, the need for robust algorithms capturing intricate non-linear relationships has never been greater. NARMAX stochastic processes (Non-linear Auto-Regressive Moving Average with eXogenous inputs), characterized by their ability to encapsulate linear and nonlinear dependencies along with exogenous factors, offer a powerful framework for tackling the challenges posed by real-world stochastic phenomena. NARMAX models have been used to model spatial, temporal and spatio-temporal systems in the fields of for example space physics [1]-[3], neurology [4]-[9], industrial processes (such as diesel engine modeling [10], welding processes [11], hydrogen fuel cells [12] and well operation [13]), 3D modelling [14], genetics [15], marine ecosystem modelling [16], biology [17], control theory [18]-[20], fault and damage detection in many areas [21]-[25], economics [26], robotics [27], [28], chaotic system modelling [29], etc.

NARMAX models are applied in the three main use-cases: model identification, behavior prediction, and system approximation.

In the model identification use-case, the system's symbolic representation itself is of interest for analysis or compensation purposes. To illustrate, having an algebraic representation of a non-linear distortion might allow to trace it back to the component generating it for correction or prevention. NARMAX models are of interest, yielding interpretable and thus also verifiable [3], [25], [27] non-linear equations consisting of arbitrary terms, as opposed to

for example artificial neural networks constituted of non-interpretable coefficient matrices and non-linearities.
System behavior prediction allows, amongst other things, to prevent the system from harming other components ([21]-[25]) or compress its output for transmission or storage. To illustrate, if a NARMAX model predicts 60% of the temporal or spatial signal's variance from past or surrounding values, only the remaining 40% must be transferred or stored as error bits.
Lastly, NARMAX systems can also replace known but computationally intensive processes, as is commonly done with Taylor or Padé expansions, without requiring knowledge of the original functions. Embedded systems and real time low-latency applications with limited memory and computational resources benefit from sparse, thus light-weight models such as NARMAX expansions.

This paper introduces a new algorithm to the black-box fitting symbolic regression algorithm class named "Forward Orthogonal Least Squares Regression" (called FOrLSR, FROLS, FOLSR or OFR depending on the sources).

FOrLSR algorithms are dictionary-based symbolic least-squares regressions, as they generate (or are provided with) a "dictionary" of regressors to find the shortest possible regressor sequence describing the system with the desired precision. Common expansions such as Taylor, Padé, Fourier, RBF and Wavelet are equivalent to fitting the entirety (being non-sparse expansions) of a dictionary containing only one single function type (polynomials, oscillations, hyper-ellipsoids, etc). The FOrLSR's dictionary, however, can contain any number of arbitrary spatial and temporal functions and combinations thereof, of which the FOrLSR selects only the most relevant to describe the system.

A further advantage is the least squares framework ensuring optimal fitting for arbitrarily large subspaces, whereas Taylor- and Padé-like expansions are only optimal at their expansion point and can quickly drift off.

The remaining paper is organized as follows: Section II presents the rFOrLSR, while the new arborescent structure is introduced in Section III. Section IV provides examples and benchmarks, while Section V concludes the paper.

Appendix 1 illustrates the used notation; Appendix 2 proves equation (2)'s correctness and Appendix 3 holds Section IV-C's rational expansion's coefficient.

II. THE RECURSIVE FORWARD ORTHOGONAL LEAST SQUARES REGRESSION (rFOrLSR)

From a linear algebra perspective, expansions are projections into function spaces with basis vector sequences (BVS) depending on the used expansion type (monomials for Taylor, oscillations for Fourier, etc), whose potentially infinite dimension depends on the expansion order. Being a model structure detection algorithm, the FOrLSR searches the BVS representing the given system response vector $\underline{y}$ with the smallest mean squared error and the shortest BVS to ensure model sparsity [30]. This optimal BVS being system-dependent, thus unknown, a large dictionary D_C containing n_C candidate basis vectors (=regressors) is iteratively scanned to extract the best fitting regressor to add it to the BVS (regression). Thus, the optimal point (regression coefficients $\underline{\hat{\theta}}$) in an optimal subspace (spanned by the BVS) of D_C's vectors is searched for. To make this operation computable, one evaluates the regressors on a zero-mean uniform white noise input sequence $\underline{x}$ of length p. The regressors $\underline{\varphi}_m$ can then be any non-linear transformations and combinations of the system input $\underline{x}$, output $\underline{y}$ or internal system noise $\underline{e}$. This corresponds to working in a vector-space spanned by finite dimensional observation vectors rather than a Hilbert space of functions. This reduces the symbolic regression problem to the usual least squares regression, solved with a Gram-Schmidt-based QR-like decomposition augmented with a similarity metric iteratively selecting the most similar basis vector to the desired system's output $\underline{y} \in \mathbb{R}^p$ (see [30]).

Thus, the FOrLSR acts as a linear equation system solver (LESS) decomposing D_C into an orthogonal and an upper unitriangular matrix, both of lower dimensions, yielding a solution vector $\underline{\hat{\theta}}$ containing the original system solution's non-zero entries.

The Gram-Schmidt-based orthogonalization allows having a forward regression not required to estimate model parameters at each iteration when adding a regressor to the BVS [30], [31]. Thus, the FOrLSR re-orthogonalizes the regressor observation dictionary $D_\mathrm{C} \coloneqq [\underline{\varphi}_j]_{j=1}^{n_\mathrm{C}} \in \mathbb{R}^{p\times n_\mathrm{C}}$ with respect to all vectors (regressors) $\underline{\psi}_j$ previously added to the BVS (model). The orthogonalization is the algorithm's most expensive operation, p and n_C being usually very large. It is, thus, of interest to vectorize it and make it recursive.

A. THE ORTHONORMAL ANNIHILATOR MATRIX

A single vector $\underline{\varphi}_m$'s orthogonalization with respect to a set containing s orthogonal vectors $\{\underline{\psi}_j\}_{j=1}^{s}$ is performed by

$$\underline{\varphi}_m - \sum_{j=1}^{s} \frac{\langle \underline{\psi}_j \,;\, \underline{\varphi}_m \rangle}{\left\| \underline{\psi}_j \right\|_2^2} \underline{\psi}_j$$

Proposition: The following orthonormal annihilator matrix P_A, projecting D_C onto $\Psi \coloneqq [\underline{\psi}_j]_{j=1}^{s}$'s orthogonal complement by left multiplication, is equivalent to the FOrLSR's nested for-loop iterating over all dictionary terms $\underline{\varphi}_m$ and all s already selected orthogonal regressors $\underline{\psi}_j$:

[EDITED] (1)

Proof: [EDITED]

P_{A}, as defined in equation (1), speeds up orthogonalization by allowing parallelized computation on GPU or distributed systems, and BLAS-like optimizations on CPU.

B. THE RECURSIVE ORTHONORMAL ANNIHILATOR MATRIX

To reduce the computational burden, the orthogonalization in equation (1) is made recursive by storing and updating the orthogonalized D_{C}, defined as Ω.

[EDITED]

[EDITED] (2)

with $\left[\underline{\omega}_j\right]_{j\in\{k\}_{k=1}^{|U|-s+2}\setminus\ell}$ being Ω from the previous iteration s without the ℓ-th column ($\underline{\omega}_\ell$). Appendix 2 proves that (2) corresponds to a multiplication with P_{A} of the correct terms.

Projections on the selected $\underline{\omega}_\ell$, being taken from Ψ's orthogonal complement, are guaranteed to not reintroduce information to any $\underline{\varphi}_m$ previously eliminated by any $\underline{\psi}_j$.

The recursive orthogonalization in equation (2) reduces the algorithms' complexity from $\mathcal{O}(p\times n_{\mathrm{C}}\times n_{\mathrm{r}}^2)$ to $\mathcal{O}(p\times n_{\mathrm{C}}\times n_{\mathrm{r}})$ with n_{r} the BVS's cardinality (number of regressors in the model), as Ω is only orthogonalized with respect to the new regressor $\underline{\omega}_\ell$ at every iteration s rather than with respect to the entire BVS.

C. IMPOSING REGRESSORS

Starting the BVS with an imposed vector-set $D_{\mathrm{S}} \coloneqq \left\{\underline{\tilde{\varphi}}_j\right\}_{j=1}^{n_{\mathrm{S}}}$ of pre-selected regressors allows to incorporate user expertise and construct arborescences, as explained in Section III. Further applications are simplifying the variable selection algorithm presented in [32], which should be run as dictionary sparsification step before the FOrLSR. It eliminates unfit variables and upper-bounds the maximum lags to limit the number of terms passed through the monomial expansion and non-linearities, which strongly reduces the dictionary size, resulting in faster regressions (see Section IV-A). Imposing regressors is also necessary for the proposed arborescence's validation procedure (see Sections III-A and III-B).
Regressors are imposed by **[EDITED]**

D. THE RECURSIVE FOrLSR ALGORITHM (rFOrLSR)

The (r)FOrLSR properly functions under following conditions: Importantly, $\underline{y}$ and all regressors in D_{C} and D_{S} must be centered, such that matrix multiplications correspond to correlations rather than cosine similarities. The input $\underline{x}$ must also be centered before being passed through the system of interest. Further, $\underline{\varphi}_i \neq \underline{\varphi}_j, \forall i\neq j$ as the FOrLSR often selects all equal regressors, once one of them is selected. Finally, all regressor entries must be independent of all model parameters: $\partial_{\theta_j}\varphi_i[k]=0, \forall_{i,j=1}^{n_r}, \forall_{k=1}^{p}$ [30].

The rFOrLSR's inputs are: The system output $\underline{y}\in\mathbb{R}^p$, the imposed and candidate regressor dictionaries, respectively $D_{\mathrm{S}}\in\mathbb{R}^{p\times n_{\mathrm{S}}}$ and $D_{\mathrm{C}}\in\mathbb{R}^{p\times n_{\mathrm{C}}}$. Further, the index set U determines which regressors are accessible to the rFOrLSR, which simplifies the arborescence, see Section III. Thus, for the rFOrLSR $|U| \mathrel{\widehat{=}} n_{\mathrm{C}}$ and $\left[\underline{\varphi}_m\right]_{m\in U} \mathrel{\widehat{=}} D_{\mathrm{C}}$. As in the classical FOrLSR [30], ρ is the maximum unexplained relative output variance threshold. Higher thresholds lead to BVS with fewer terms as more error is tolerated [30]. The integer [MaxTerms] and the Boolean [Abort] allow the pruning mechanism described in Section III-B. [MaxTerms] also allows limiting expansions to a particular order (= model length, number of terms) as is usual for expansion. Finally, the Boolean [Solve] determines if the regression coefficients $\underline{\hat{\theta}}$ are to be estimated and the ordered index set $\boldsymbol{\mathcal{L}_I}$ is needed for the arborescence's OOIT-abortion described in sections III-B and III-C.

The rFOrLSR operates with the same data structures as the FOrLSR [30]: W is an ordered set containing the orthogonal regressor's regression coefficients, being static once computed due to orthogonality. $\mathcal{E}$ is an ordered set containing each term's Error Reduction Ratio (ERR), which measures each term's contribution to the regression's explained empiric variance $s_{\underline{y}}^2$ [30]. It corresponds to the squared correlation between the orthogonalized regressor $\underline{\psi}_j$ and the system output $\underline{y}$, assuming both are zero-mean. The upper unitriangular matrix A contains the projection coefficients of the non-orthogonal selected regressors $\underline{\varphi}_m$ onto the selected and orthogonalized regressors $\underline{\psi}_j$.

ALGORITHM 1: **The rFOrLSR algorithm**

[EDITED] (pseudo code)

[EDITED] (explanations)

The (r)FOrLSR functions as a lossy sparsifying QR-like equation solver, where $M\underline{x}=\underline{b}\rightarrow R\underline{x}=Q^T\underline{b}$ becomes $(D_{\mathrm{S}} \mathbin{\#} D_{\mathrm{C}})\underline{\hat{\theta}}=\underline{y}\rightarrow A\underline{\hat{\theta}}=\Psi_{\mathrm{n}}^T\underline{y}=W$. It retains a selection of D_{C}'s columns to minimize Ψ's, A's and $\underline{\hat{\theta}}$'s dimensions while keeping the mean squared error increase with respect to $\underline{y}$ minimal [30]. Equivalently, $\underline{\hat{\theta}}$ remains in $\mathbb{R}^{n_{\mathrm{C}}+n_{\mathrm{S}}}$ and is sparse, having non-zero entries at indices in $\mathcal{L}$. Ψ's columns span an orthogonal basis of a subspace of $D_{\mathrm{S}} \mathbin{\#} D_{\mathrm{C}}$, where W is the least squares optimal representation of $\underline{y}$.

III. THE ARBORESCENT ORTHOGONAL LEAST SQUARES REGRESSION (AOrLSR)

The (r)FOrLSR is a greedy algorithm since at each iteration the orthogonalized regressor the most similar to the system output is added to the model [30]-[33]. Thus, the algorithm takes the locally best choice, maximizing the increase in explained output variance without considering the global search space, which could contain shorter and/or lower validation error BVSs [33]. A common problem is the selection of a highly correlated but otherwise suboptimal first term, often $y[k-1]$, resembling the most the output $y[k]$

[31], [33], such that all following regressors are selected to compensate for that error, yielding a suboptimal model. This can, however, happen at any regression iteration [33].

Guo *et al*. [31] propose the iFOrLSR, performing a first regression and starting a new regression with each selected regressor. This mitigates greediness by forcing the FOrLSR to start in different search space locations to potentially find better BVS [31]. Traversing the entire search space would require $\mathcal{O}(n_C!)$ regressions and thus the greedy similarity-metric-based selection remains necessary [31]. Equivalently, finding an arbitrarily determined linear equation system's sparsest solution vector is considered NP-hard [34].

Guo *et al*. [31] visualize the iFOrLSR as a search tree having regressors as nodes and edges connecting two successively added terms to the BVS. They note the potential usefulness of retriggering regressions imposing more terms, which, however, becomes quickly computationally intractable and they provide no method for doing so. Both points are addressed below.

A. THE ARBORESCENCE CONSTRUCTION

This section generalizes the iFOrLSR's [31] arborescence traversal mitigating greediness and increasing the solution space exploration. Imposing regressors from later in the BVS while dropping earlier ones allows the arborescence to eliminate regressors at every level and replace them with new ones. Thus, deeper arborescences have higher probability of finding better BVSs (models). The AOrLSR explores more search space than an equivalent forward-backward regression (as in [33]), as it explores new combinations rather than just eliminating poor regressors from the current BVS.

[EDITED]

Once all nodes are processed, the shortest (for sparsity) BVSs are compared with user-defined metrics on a validation set of input sequences . The validation procedure, taking an arbitrary data-structure as input, outputs a scalar fitness measure, which the AOrLSR uses to select the best performing BVS. This allows use-case specific metrics to process arbitrary data-structures (see Section III-C).

B. THE SPARSIFICATION THEOREMS

Spanning a deep arborescence is computationally intractable due to the regression numbers' combinatorial growth (see Section IV-B). This section provides an observation yielding four sparsification theorems, four corollaries, one pruning mechanism and upper-bounds guaranteeing the algorithm's termination.

The sparsification theorems avoid computing nodes (regressions) whose results are predictable (LUT, OOIT, (generalized) PFCT) or predictably rejected by the validation method (ADTT and PM), which greatly sparsifies the arborescence without losing potential solution BVS.

Observation: [EDITED]

Predictable Free Choice Theorem (PFCT): [EDITED]

Proof: [EDITED]

Leaf Uniqueness Theorem (LUT): [EDITED]

Proof: [EDITED]

Orthogonalization Order Independence Theorem (OOIT): [EDITED]

Proof: [EDITED]

Despite the chosen vectors $\underline{\varphi}_i$ remaining identical, the orthogonalized vectors $\underline{\psi}_i$ and the QR-like decomposition's upper unitriangular matrix A change depending on vector ordering. The order being irrelevant, so is the sorting criterion, if consistent throughout the AOrLSR.

Corollary (Generalized OOIT): [EDITED]
Example: [EDITED]

Corollary (Generalized PFCT): [EDITED]

The Generalized PFCT reduces the arborescence's computational growth from permutational to combinatorial, which is a factorial order smaller.
[EDITED]

Unique Regressions Upper-bound Theorem: [EDITED]

Proof: [EDITED]

Figure 1: [EDITED]

Arborescence depth truncation theorem (ADTT): [EDITED]

Proof: [EDITED]

Corollary (Arborescence depth upper bound): [EDITED]

Proof: [EDITED]

Corollary (Guaranteed Termination): [EDITED]

Proof: [EDITED]

Pruning mechanism (PM): [EDITED]

C. THE AOrLSR ALGORITHM

The AOrLSR algorithm's inputs are the following: The data-structures $\underline{y}$, D_S and D_C are the same as for the rFOrLSR. The zero-based [MaxDepth] determines the arborescence depth (total number of levels). ρ_1 is the root regression's ERR threshold, which, as described in [31], can be set higher than the desired model precision threshold ρ_2 to span a horizontally larger arborescence by having a longer root BVS. Note that Guo *et al*. [31] use ρ_2 as offset for ρ_1, while here those are separate thresholds for simplicity. $\mathcal{F}$ and $\mathcal{V}$ correspond respectively to the validation function and data, which are application and implementation dependent (see explanations of step 7 below).

[EDITED]

ALGORITHM 2: **The AOrLSR algorithm**

[EDITED] (pseudo-code)

[EDITED] (explanations)

The last step (7) performs validation and model selection once the arborescence is traversed and returns the regression of the minimum length BVS with the highest validation score. The validation score is application dependent and should test important model characteristics. To illustrate, the validation could, additionally to the ERR, operate in the frequency domain and also penalize models based on their computational expense. The validation can also reject models based on arbitrary criteria like using negative or too large regression coefficients.

From the sparsifying linear equation system solver perspective, the AOrLSR imposes $|\mathcal{L}_I|$ non-zero entries in the solution vector and continues the iterative solving to find sparser solutions, while all theorems apply equally.

D. SUPPLEMENTARY REMARKS

The early abortion functionality, being offered by the rFOrLSR to support the AOrLSR's pruning mechanism, allows to set a maximum number of non-zero terms in the solution vector in a LESS context. By setting $\rho = 0$, the rFOrLSR runs until [MaxDepth] $\leq n_{\mathrm{C}}$ is reached. The rFOrLSR-based LESS can thus be used either with an error threshold or with a determined number of non-zero entries.

The arborescence traversal can be performed by a depth-first search (DFS), which seems attractive as from one node to its child or sibling only one imposed term must be changed, avoiding many computations. However, the rFOrLSR cannot be used since un-orthogonalizing the dictionary (D_{C}/Ω) w.r.t. a regressor is impossible and would require storing copies quickly overflowing RAM. This also trades imposing terms once at regression start, the cheapest operation, against re-orthogonalizing D_{C} at every iteration, being the most expensive operation.

The AOrLSR is a meta-algorithm triggering the (r)FOrLSR, which does not prevent the use of variations, such as for example the Ultra-orthogonal FOrLSR [35] or an information criterion-based FOrLSR [36], [37]. Different similarity metrics and data augmentation remain compatible with the (r)FOrLSR with some adjustments.

IV. EXAMPLES

Section IV-A and IV-B illustrate the speed improvement brought by the rFOrLSR and the sparsification theorems and corollaries. Section IV-C illustrates the ability to create nested expansions and Section IV-D displays both algorithms use as Linear Equation System Solver.
For Sections IV-B and IV-C, the results slightly differ depending on the white noise input sequence $\underline{x}$, thus the most representative of multiple runs was taken.

A. EXECUTION SPEED

This example compares the rFOrLSR to a modified FOrLSR implementation having a similar structure and storing the same variables for a fair comparison. The fitted system and dictionary content are irrelevant, the execution speed per regressor being independent of the vectors' content. Table I (CPU benchmark) and Table II (GPU benchmark) illustrate the fitting durations for $p \coloneqq 2'000$ in seconds averaged over 50 passes, w.r.t. dictionary size n_{C} and number of regressors (BVS length) n_{r}. The rows compare the rFOrLSR's (top numbers) linear complexity in n_{r} to the FOrLSR quadratic complexity.

TABLE 1: **rFOrLSR (top) and FOrLSR (bottom) fitting durations in seconds w.r.t. dictionary size n_C and BVS length (number of fitted regressors) n_r for vector length $p \coloneqq 2'000$ on CPU. The rFOrLSR takes a fraction of the FOrLSR fitting times in all cases.**

Table 1	$n_c = 100$	$n_c = 1k$	$n_c = 10k$	$n_c = 100k$
$n_r = 5$	0.009	0.072	0.752	9.328
	0.047	0.433	3.909	38.567
$n_r = 10$	0.019	0.155	1.594	19.488
	0.162	1.390	13.249	131.789
$n_r = 20$	0.038	0.316	3.306	39.881
	0.518	5.351	52.879	515.110
$n_r = 30$	0.055	0.500	4.948	60.834
	1.080	11.754	111.357	1085.402
$n_r = 40$	0.067	0.669	6.592	80.615
	1.738	19.489	188.725	1872.121

TABLE 2: **Same as Table 2 but benchmarked on GPU.**

Table 2	$n_c = 100$	$n_c = 1k$	$n_c = 10k$	$n_c = 100k$
$n_r = 5$	0.005	0.018	0.037	0.452
	0.190	1.871	19.574	200.086
$n_r = 10$	0.011	0.039	0.078	0.935
	0.693	5.729	59.242	595.988
$n_r = 20$	0.023	0.077	0.154	1.902
	2.001	20.254	195.572	1930.037
$n_r = 30$	0.033	0.119	0.228	2.876
	3.431	39.639	407.852	4067.117
$n_r = 40$	0.038	0.158	0.298	3.851
	5.155	67.707	699.954	6886.958

The above processing times are based on the paper's reference implementation, being uncompiled single-thread python using PyTorch, which should be considered as approximate upper bound compared to pre-compiled, multi-threaded or distributed implementations. The used computer has an AMD Ryzen 8-Core CPU with 3.3GHz/4.3GHz Turbo and an NVIDIA RTX 3080 (Laptop) GPU.

Table 1 and 2 display the rFOrLSR outperforming the FOrLSR on CPU and GPU for all n_{C} and n_{r}. This is mainly due to the linear (rFOrLSR) vs quadratic complexity in n_{r}

but also the increased CPU and GPU optimizations which can be performed on larger memory-contiguous data-structures (better caching, SIMD, GPU-parallelization, etc). The FOrLSR performs better on CPU, the algorithm comprising many scalar or small vector operations, which are slow on GPU. The rFOrLSR, however, performs much better on GPU due to the large matrix operations.

B. SPARSIFICATION

The following examples illustrate the sparsification performed by Section IV-B's theorems, corollaries and the pruning mechanism. "Naïve" stands for the total number of nodes in the arborescence per level, all of which require evaluation without the theorems. "LUT" represents nodes requiring computation applying only LUT (a), and "GPFCT" applies the OOIT-generalized PFCT corollary of which LUT is a special case. "Aborted" counts the nodes whose computation is exited due to the OOIT-prediction performed during the regression. "Ortho%" counts the percentage of orthogonalizations necessary to traverse the arborescence with the theorems compared to a naïve traversal. This is representative of the general computation expense reduction, orthogonalizations being by several orders of magnitude the most expensive operation. All mentioned numbers are cumulative sums, being the statistics for the entire arborescence up to each level. "PM" counts the aborted regressions if the respective levels were the final one. "MinLen" represents the shortest known BVS after traversing that level.

Only the last level takes into consideration the orthogonalizations prevented by the leaf-pruning mechanism PM in "Ortho%". Thus, arborescences terminated at higher levels will have slightly lower "Ortho%" numbers than indicated.

The following examples represent both extremes in (r)FOrLSR fitting difficulty and the repercussions on the arborescence's usefulness and sparsity. Furthermore, system 1 and 2 were chosen to be severely non-linear systems to illustrate the expression complexity the (r)FOrLSR and AOrLSR are capable of correctly retrieving.
The ERR tolerances are set to $\rho_1 := \rho_2 := 1\text{e}-4$, such that 0.01% of the signal $\underline{y}$'s empiric variance can be left unexplained by the model for $p := 2'000$.

Let system 1 be:

$$y[k] := 0.2x[k] + 0.3x^3[k-1] + 0.7|x[k-2]x^2[k-1]| + 0.5e^{x[k-3]x[k-2]} - 0.5\cos(y[k-1]x[k-2]) - 0.4|x[k-1]y^2[k-2]| - 0.4y^3[k-3]$$

The required dictionary is:

$$D_\text{C} := \left\{ f_i \left(\prod_{j=0}^{11} \varphi_j^{a_j} \right) \middle| a_j \in \{q\}_{q=0}^{3}, \left|\sum_{j=1}^{11} a_j\right| \leq 3, \forall j \right\} \quad (4)$$

with $\varphi_j \in \{x[k-j]\}_{j=0}^{5} \cup \{y[k-j]\}_{j=1}^{5}$ in addition to $f_i \in \{x, |x|, e^x, \cos(x)\}$. Thus, D_C contains delays up to 5 timesteps for third-order monomial expansions of x, y passed through the above functions, such that $|D_\text{C}| \triangleq n_\text{C} = 1441$.

TABLE 3: **Arborescence sparsification demonstration per level L (columns) via the total nodes, the remaining nodes after applying respectively LUT, then OOIT-augmented PFCT. Below are the number of OOIT aborted regressions, the number of regressions aborted by the pruning mechanism and the relative number of orthogonalizations. The last row contains the shortest known sequence length after that level.**

Table 3	L=0	L=1	L=2	L=3	L=4	L=5	L=6	L=7
Naïve	1	8	50	276	1'326	5'396	18'348	51'668
LUT	1	7	42	226	1'050	4'070	12'952	33'320
GPFCT	1	7	22	50	104	196	305	386
Aborted	0	5	18	43	89	159	246	306
PM	0	0	2	10	40	86	108	81
Ortho%	100	55.55	20.37	6.80	2.62	1.08	0.43	0.21
MinLen	7	7	7	7	7	7	7	7

For most input sequences, System 1 is correctly retrieved by the FOrLSR on its own (= root), thus, the minimum length (MinLen) remains constant. Thus, further arborescence levels are strongly redundant, as recognized by "GPFCT", which together prevent computing most nodes. Using all proposed theorems, the arborescence is traversed by computing only $\frac{386}{51668} \approx 0.47\%$ of all nodes before exiting the search due to ADTT. Additionally, $\frac{306}{386} \approx 79.3\%$ were OOIT-aborted and the pruning mechanism aborted 81 regressions, meaning that the arborescence was traversed with about 0.21% of the naïve computations. The rFOrLSR's further reduces computations to about $\frac{0.039}{1.39} \cdot 100 \approx 2.8\%$ (GPU rFOrLSR / CPU FOrLSR from Table 1&2 respectively at $n_r = 10$, $n_\text{C} = 1k$), yielding a total computation time of roughly 5.88e-3% of the CPU (best time for the FOrLSR) FOrLSR-based naïve arborescence.

Let system 2 be rational: $y[k] := \frac{N}{D}$ with

$$N := 0.6|x[k]| - 0.35x^3[k] - 0.3x[k-1]y[k-2] + 0.1|y[k-1]|$$

$$D := 1 - 0.4|x[k]| + 0.3|x[k-1]x[k]| - 0.2x^3[k-1] + 0.3y[k-1]x[k-2]$$

The required dictionary is:

$$D_\text{C} := \left\{ f_i \left(\prod_{j=0}^{11} \varphi_j^{a_j} \right) \middle| a_j \in \{q\}_{q=0}^{3}, \left|\sum_{j=1}^{11} a_j\right| \leq 3, \forall j \right\}$$

with φ_j as above, with however $f_i \in \{x, |x|, -yx, -y|x|\}$, as the denominator terms are created by multiplying the regressors with $-y$ to linearize the expression [38]:

$$y[k] = \frac{A}{1+B} \Leftrightarrow y[k](1+B) = A \Leftrightarrow y[k] = A - y[k]B$$

where A and B are arbitrary linear combinations.

TABLE 4: **Same content as Table 3 with System 2's data**

Table 4	L=0	L=1	L=2	L=3	L=4	L=5	L=6
Naïve	1	23	482	9'785	192'022	3.65M	67.77M
LUT	1	22	459	9'303	182'237	3.46M	64.12M
GPFCT	1	22	312	3'786	41'719	428'159	4.09M
Aborted	0	9	167	2'279	27'448	298'785	2.97M
PM	0	12	285	3'463	37'901	386'429	3.67M
Ortho%	100	67.23	37.02	19.13	9.73	4.89	2.14
MinLen	22	15	10	9	8	8	8

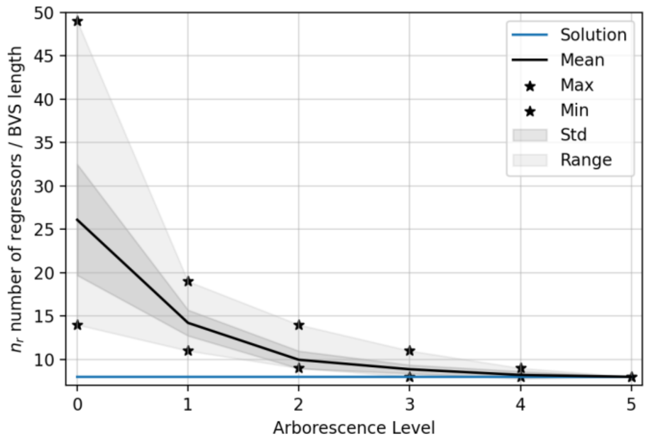

FIGURE 2: **Illustration of the arborescence convergence via the statistical properties of 150 different input random sequences $\underline{x}$ for system 2. The correct solution (blue line) has 8 regressors and the Max/Min, Mean, standard deviation (Std) and range illustrate the initial poor (r)FOrLSR performance progressively corrected by the arborescence.**

System 2 is so complex that the (r)FOrLSR on its own (root regression) performs very poorly as illustrated in the example run (Table 4) and in general (Figure 2). Depending on the used input noise sequence, the root's BVS length is between 14 and 49 (instead of 8) and the arborescence must arrive at L=3-5 to retrieve the correct equation.

System 2's arborescence levels yield progressively shorter BVSs until the correct expression is retrieved, illustrating the gain of spanning large search spaces. Furthermore, for complex systems, the (r)FOrLSR results dependent strongly on the input sequence, which the AOrLSR stabilizes by design as illustrated in Figure 2, where the range and standard deviation greatly decrease at each level. Only 18.67% of the 150 runs find the correct solution by level 3, 72% by level 4 and all by level 5.

The arborescence traversal in Table 4 requires computing 2.14% of all orthogonalizations and the pruning mechanism aborted $\frac{3.67\text{M}}{4.09} \approx 89.73\%$ of all nodes. The rFOrLSR reduces the remaining computations to a about $\frac{0.119}{11.754} \cdot 100 \approx 1\%$, (Table 1&2 respectively at $n_r = 30$, $n_C = 1k$) yielding a total computation time of roughly 0.021% of the CPU FOrLSR-based naïve arborescence.

The number of levels decreasing the BVS length depends on the fitted system and the input sequence. Empirically, for difficult to fit systems, BVS lengths decrease in the first 2-6 levels with mostly the largest decrease after the root. Some systems lose most of their regressors, while others lose almost none. For easily fitted systems such as system 1, the arborescence has few nodes and is very sparse which minimizes its cost. Indeed, for linear systems such as FIRs or IIRs, the total number of nodes is often a few hundreds. The deeper the arborescence level, the lower the percentage of computed nodes and the higher the abortion rate.

Importantly, two or three consecutive levels can yield identical shortest known BVS lengths while being followed by a level yielding a shorter one. Thus, stopping the arborescence after a level fails to find a shorter BVS is a suboptimal stopping criterion.

Even if no shorter BVS is discovered, further levels remain of interest as they often provide supplementary same length candidate BVSs for the validation procedure to choose from, to decrease the user-defined error metric.

C. ADVANCED EXPANSIONS / SYMBOLIC FITTING

As demonstrated by system 2, the requirement that the NARMAX expansions sparsely fitted by the AOrLSR be linear-in-the-parameter is quite loose. The (r)FOrLSR's advantage over other methods is that, being based on vector similarity metrics, the vectors' content is not subject to common fitting constraints such as differentiability, continuity or system causality.

This example illustrates an expansion in $|x|^j$ inside a rational non-linear expression designed to emulate $\tanh(x)$ for $x \in \mathbb{R}$. The constraints are that the approximation be a single expression, converge to ± 1 and have a small maximum error around the origin, which are hard to achieve with Padé expansions or rational systems such as system 2.

The proposed approximation has the form:

$$y = \text{sgn}(x)\left(1 - \frac{1}{1 + |x|A}\right)$$

with $A \coloneqq \sum_{j \in J} \theta_j |x|^j$ and $J \subset \mathbb{N}_0$, which linearizes to

$$-\text{sgn}(x)y + 1 = (\text{sgn}(x)y - 1)|x|A$$

Yielding

$$D_C \coloneqq \{(\text{sgn}(x)y - 1)|x|^j \,\big|\, j \in \mathbb{N}^*\}$$

This example's input sequence $\underline{x}$ is a zero-mean uniform noise of length $p \coloneqq 15'000$, with powers in $J \coloneqq \{i\}_{i=1}^{15}$, yielding a dictionary of 15 terms. The optimal noise amplitude was determined by a grid search in [2,4] for each expansion order. The AOrLSR's validation function was an L_∞ norm on the equidistantly sampled interval of $[-8,8]$, to keep the shortest BVS with the smallest maximal deviation.

The fitting errors of the respectively 5, 6 and 7 term sigmoid expansions are illustrated in Figure 3 below, while the coefficients and input noise amplitudes are provided in Appendix 3.

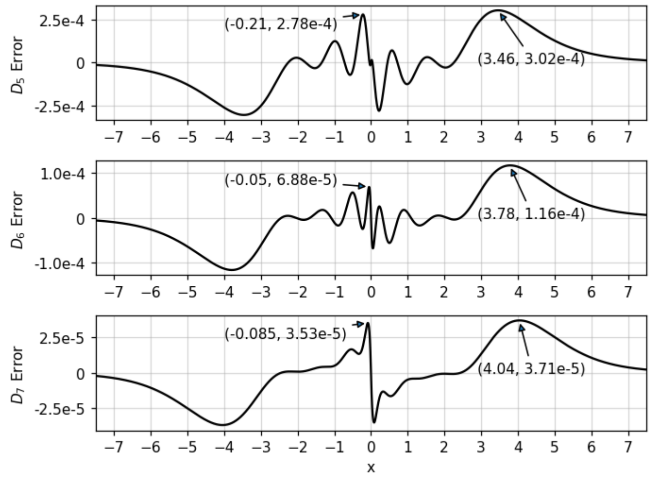

FIGURE 3: **The presented rational NARMAX expansion's fitting error (tanh(x) – 5-7 order expansions) per order (number of terms in the powers of absolute values in A) with tuples containing the respective maximal deviation coordinates of the two tallest peaks.**

A depth 3 arborescence finds the optimal BVSs for all three expansion orders. Its traversal is sparser than computing each possible regressor combination of which there are nCk (with nCk the combinatorial "from n choose k" operator).

Although level 1 and 2 suffice to find a minimum length BVS, the additional level finds lower error BVS.

For the 5th order expansion, only 80 regressions of $15C5 = 3003$ (2.66%) are computed of which 36 (45%) are early aborted. For the 6th order expansion, $100/5005 \approx 1.99\%$ of all regressions are computed of which 60 (60%) are early aborted. For the 7th order expansion, $120/6435 \approx 1.86\%$ of all regressions are computed, of which 82 (68.3%) are early aborted.

D. SPARSIFYING LINEAR EQUATION SYSTEM SOLVER

As mentioned in sections II and II-D, the (r)FOrLSR and the AOrLSR are, on an abstract level, linear equation system solvers, maximizing the solution-vector sparsity by choosing the most relevant LHS columns, while minimizing the mean squared error with respect to the RHS $\underline{y}$ [30].

The LHS is $M \coloneqq D_S \mathbin{+\!\!\!+} D_C$, where D_S allows to impose columns to the selection. M's dimensions being arbitrary, the system can be under-, well- or over-determined.

To illustrate, be the following system with $p \in \mathbb{N}^*$:

$$M\underline{x} = \underline{y} \overset{\Delta}{\Leftrightarrow} \begin{bmatrix} m_{1,1} & \cdots & m_{1,5} \\ \vdots & \ddots & \vdots \\ m_{p,1} & \cdots & m_{p,5} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix}$$

The AOrLSR with a 1% error tolerance could yield $\mathcal{L} = \{1,3\}$ and $\underline{\hat{\theta}} \in \mathbb{R}^{|\mathcal{L}|=2}$, such that $\underline{x} = \left[\hat{\theta}_1, 0, \hat{\theta}_2, 0, 0\right]^T$, while a 0.1% tolerance could yield $\mathcal{L} = \{1,3,4\}$ and $\underline{\hat{\theta}} \in \mathbb{R}^{|\mathcal{L}|=3}$, thus $\underline{x} = \left[\hat{\theta}_1, 0, \hat{\theta}_2, \hat{\theta}_3, 0\right]^T$.

The LES is thus reduced to $\left[\underline{m}_j\right]_{j\in\mathcal{L}} \underline{\hat{\theta}} = \underline{y}$, with Ψ and A matching dimension orthogonal, and unitriangular matrices, as in a QR decomposition [30].

V. CONCLUSION

This paper proposes the rFOrLSR (recursive forward orthogonal least squares regression), being a recursive matrix form of the FOrLSR [30], and the AOrLSR (arborescent orthogonal least squares regression), being a meta-algorithm triggering the (r)FOrLSR to reduce the number of regressors used to describe the desired system.

[EDITED]

The rFOrLSR and the sparsification theorems allow spanning an otherwise computationally intractable arborescence which scans a much larger search space segment than the FOrLSR. This greatly increases the probability of finding the optimal (sparser, lower error or more complex) NARMAX expansion within the given dictionary.

Thus, this paper lays the groundwork for future arborescent (r)FOrLSR algorithms using further sparsification heuristics (search-tree branch pruning methods, regressor elimination based on MSE contribution, etc).

The (r)FOrLSR / AOrLSR are also linear equation system solvers for arbitrarily determined systems which maximize solution sparsity and minimize mean square error.

The first authors' GPU-accelerated python AOrLSR library is available at https://github.com/Stee-T/rFOrLSR.

Future research could go towards further sparsification theorems and heuristics and transforming current regressor selection and elimination procedures into arborescence pruning heuristics. Further, regularization methods keeping the solution vector entries small or imposing constraints like coefficient positivity could allow using the AOrLSR for problems requiring specific sparsifying linear solvers.

Finally, the rich literature on external parameter support, such as [9], [39], [40], would allow modulating the expansion's parameters $\underline{\hat{\theta}}$ to model system changes.

Further, the orthogonalization matrix P_A will be used in the authors' upcoming work to morph regressors to adapt to the current system via infinitesimal optimization.

AKNOWLEDGMENT

The first author thanks Paolo Combes for his thorough proof-reading and feedback.

REFERENCES

[1] R. J. Boynton, M. A. Balikhin, S. A. Billings, H. L. Wei, and N. Ganushkina, "Using the NARMAX OLS-ERR algorithm to obtain the most influential coupling functions that affect the evolution of the magnetosphere," *J. Geophys. Res*., vol. 116, no. A05218, 2011. DOI: 10.1029/2010JA015505.

[2] R. Boynton, M. Balikhin, H.-L. Wei, Z.-Q. Lang, "Chapter 8 - Applications of NARMAX in Space Weather," in *Machine Learning Techniques for Space Weather*, E. Camporeale, S. Wing,

J. R. Johnson, Eds., Elsevier, 2018, pp. 203-236. DOI: 10.1016/B978-0-12-811788-0.00008-1.

[3] M. A. Balikhin et al., “Using the NARMAX approach to model the evolution of energetic electrons fluxes at geostationary orbit,” *Geophysical Research Letters*, vol. 38, no. 18, 2011. doi:10.1029/2011gl048980

[4] F. He and Y. Yang, “Nonlinear system identification of neural systems from neurophysiological signals,” *Neuroscience*, vol. 458, pp. 213–228, 2021. doi:10.1016/j.neuroscience.2020.12.001

[5] D. Florescu and D. Coca, “Identification of linear and nonlinear sensory processing circuits from spiking neuron data,” *Neural Computation*, vol. 30, no. 3, pp. 670–707, 2018. doi:10.1162/neco_a_01051

[6] R. Tian, Y. Yang, F. C. van der Helm, and J. P. Dewald, “A novel approach for modeling neural responses to joint perturbations using the NARMAX method and a hierarchical neural network,” *Frontiers in Computational Neuroscience,* vol. 12, 2018. doi:10.3389/fncom.2018.00096

[7] Y. Gu et al., “Nonlinear modeling of cortical responses to mechanical wrist perturbations using the NARMAX method,” *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 3, pp. 948–958, 2021. doi:10.1109/tbme.2020.3013545

[8] F. He et al., “Nonlinear interactions in the Thalamocortical Loop in essential tremor: A model-based frequency domain analysis,” *Neuroscience,* vol. 324, pp. 377–389, 2016. doi:10.1016/j.neuroscience.2016.03.028

[9] Y. Li, H.-L. Wei, Stephen. A. Billings, and P. G. Sarrigiannis, “Identification of nonlinear time-varying systems using an online sliding-window and common model structure selection (CMSS) approach with applications to EEG,” *International Journal of Systems Science,* vol. 47, no. 11, pp. 2671–2681, 2015. doi:10.1080/00207721.2015.1014448

[10] G. Zito and I. D. Landau, “A methodology for identification of NARMAX models applied to diesel engines,” *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 374–379, 2005. doi:10.3182/20050703-6-cz-1902.00063

[11] Y. Cao, Z. Wang, S. Hu, and W. Wang, “Modeling of weld penetration control system in GMAW-P using NARMAX methods,” *Journal of Manufacturing Processes*, vol. 65, pp. 512–524, 2021. doi:10.1016/j.jmapro.2021.03.039

[12] Z. Deng, Q. Chen, L. Zhang, and Z. Fu, “DATA DRIVEN NARMAX modeling for PEMFC Air Compressor,” *International Journal of Hydrogen Energy*, vol. 45, no. 39, pp. 20321–20328, 2020. doi:10.1016/j.ijhydene.2019.11.228

[13] D. J. Pagano, V. D. Filho, and A. Plucenio, “Identification of polinomial NARMAX models for an oil well operating by continuous gas-lift,” *IFAC Proceedings Volumes*, vol. 39, no. 2, pp. 1113–1118, 2006. doi:10.3182/20060402-4-br-2902.01113

[14] E. Perracchione, “Rational RBF-based partition of unity method for efficiently and accurately approximating 3D objects,” *Computational and Applied Mathematics*, vol. 37, no. 4, pp. 4633–4648, 2018. doi:10.1007/s40314-018-0592-8

[15] K. Krishnanathan, S. R. Anderson, S. A. Billings, and V. Kadirkamanathan, “A data-driven framework for identifying nonlinear dynamic models of genetic parts,” *ACS Synthetic Biology*, vol. 1, no. 8, pp. 375–384, 2012. doi:10.1021/sb300009t

[16] A. M. Marshall et al., “Quantifying heterogeneous responses of fish community size structure using novel Combined Statistical Techniques*,” Global Change Biology*, vol. 22, no. 5, pp. 1755–1768, 2016. doi:10.1111/gcb.13190

[17] S. L. Kukreja, H. L. Galiana, and R. E. Kearney, “Narmax representation and identification of ankle dynamics,” *IEEE Transactions on Biomedical Engineering,* vol. 50, no. 1, pp. 70–81, 2003. doi:10.1109/tbme.2002.803507

[18] J. Bernat, J. Kołota, and P. Superczyńska, “NARMAX approach for the identification of a dielectric electroactive polymer actuator,” *International Journal of Control, Automation and Systems*, vol. 21, no. 9, pp. 3080–3090, 2023. doi:10.1007/s12555-022-0518-5

[19] G. Triantos and A. T. Shenton, “Narmax structure selection for Powertrain Control,” *IFAC Proceedings Volumes*, vol. 37, no. 22, pp. 279–285, 2004. doi:10.1016/s1474-6670(17)30357-9

[20] J. S.-H. Tsai et al., “A NARMAX model-based state-space self-tuning control for Nonlinear Stochastic Hybrid Systems,” *Applied Mathematical Modelling*, vol. 34, no. 10, pp. 3030–3054, 2010. doi:10.1016/j.apm.2010.01.011

[21] L. Aggoun and Y. Chetouani, “Fault detection strategy combining NARMAX model and Bhattacharyya distance for process monitoring,” *Journal of the Franklin Institute,* vol. 358, no. 3, pp. 2212–2228, 2021. doi:10.1016/j.jfranklin.2021.01.001

[22] H. Huang et al., “Study of cumulative fatigue damage detection for used parts with nonlinear output frequency response functions based on Narmax modelling,” *Journal of Sound and Vibration,* vol. 411, pp. 75–87, 2017. doi:10.1016/j.jsv.2017.08.023

[23] Z. K. Peng, Z. Q. Lang, C. Wolters, S. A. Billings, and K. Worden, “Feasibility Study of structural damage detection using NARMAX modelling and nonlinear output frequency response function based analysis,” *Mechanical Systems and Signal Processing*, vol. 25, no. 3, pp. 1045–1061, 2011. doi:10.1016/j.ymssp.2010.09.014

[24] A. J. Wootton, J. B. Butcher, T. Kyriacou, C. R. Day, and P. W. Haycock, “Structural Health Monitoring of a footbridge using Echo State Networks and NARMAX,” *Engineering Applications of Artificial Intelligence,* vol. 64, pp. 152–163, 2017. doi:10.1016/j.engappai.2017.05.014

[25] Z. Wei, L. H. Yam, and L. Cheng, “NARMAX model representation and its application to damage detection for multi-layer Composites,” *Composite Structures*, vol. 68, no. 1, pp. 109–117, 2005. doi:10.1016/j.compstruct.2004.03.005

[26] C. McHugh, S. Coleman, and D. Kerr, “Hourly electricity price forecasting with Narmax,” *Machine Learning with Applications*, vol. 9, p. 100383, 2022. doi:10.1016/j.mlwa.2022.100383

[27] O. Akanyeti, I. Rañó, U. Nehmzow, and S. A. Billings, “An application of Lyapunov stability analysis to improve the performance of Narmax models,” *Robotics and Autonomous Systems*, vol. 58, no. 3, pp. 229–238, 2010. doi:10.1016/j.robot.2009.11.001

[28] I. M. Yassin et al., “Binary particle swarm optimization structure selection of nonlinear autoregressive moving average with exogenous inputs (NARMAX) model of a flexible robot arm,” *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 5, p. 630, 2016. doi:10.18517/ijaseit.6.5.919

[29] P. Han, S. Zhou, and D. Wang, “A multi-objective genetic programming/ NARMAX approach to chaotic systems identification,” *2006 6th World Congress on Intelligent Control and Automation*, pp. 1735–1739, Oct. 2006. doi:10.1109/wcica.2006.1712650

[30] S. A. Billings, “Model Structure Detection And Parameter Estimation”, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains.* s.l.:John Wiley & Sons, Ltd.. 2013, pp. 64-84.

[31] Y. Guo, L. Z. Guo, S. A. Billings, and H.-L. Wei, “An iterative orthogonal forward regression algorithm,” *International Journal of Systems Science*, vol. 46, no. 5, pp. 776–789, 2014. doi:10.1080/00207721.2014.981237

[32] H. L. Wei, S. A. Billings, and J. Liu, “Term and variable selection for non-linear system identification,” International Journal of Control, vol. 77, no. 1, pp. 86–110, 2004. doi:10.1080/00207170310001639640

[33] L. Piroddi and W. Spinelli, “A pruning method for the identification of polynomial NARMAX models,” *IFAC Proceedings Volumes*, vol. 36, no. 16, pp. 1071–1076, 2003. doi:10.1016/s1474-6670(17)34901-7

[34] S. Jokar and M. E. Pfetsch, “Exact and approximate sparse solutions of underdetermined linear equations,” *SIAM Journal on Scientific Computing*, vol. 31, no. 1, pp. 23–44, 2008. doi:10.1137/070686676

[35] Y. Guo, L. Z. Guo, S. A. Billings, and H.-L. Wei, “Ultra-orthogonal forward regression algorithms for the identification of Non-Linear Dynamic Systems,” *Neurocomputing*, vol. 173, pp. 715–723, 2016. doi:10.1016/j.neucom.2015.08.022

[36] H.-L. Wei , S. A. Billings, “Sparse Model Identification Using a Forward Orthogonal Regression Algorithm Aided by Mutual Information”. *IEEE Transactions on Neural Networks,* 18(1), pp. 306 - 310, 2007, DOI: 10.1109/TNN.2006.886356.

[37] [1] H. L. Wei and S. A. Billings, “Model structure selection using an integrated forward orthogonal search algorithm assisted by squared correlation and mutual information,” *International Journal of Modelling, Identification and Control*, vol. 3, no. 4, p. 341, 2008. doi:10.1504/ijmic.2008.020543

[38] Q. M. Zhu, “An implicit least squares algorithm for nonlinear rational model parameter estimation,” *Applied Mathematical Modelling*, vol. 29, no. 7, pp. 673–689, 2005. doi:10.1016/j.apm.2004.10.008

[39] Y. Gu and H.-L. Wei, “A robust model structure selection method for small sample size and multiple datasets problems,” *Information Sciences*, vol. 451–452, pp. 195–209, 2018. doi:10.1016/j.ins.2018.04.007

[40] A. Kadochnikova, Y. Zhu, Z.-Q. Lang, and V. Kadirkamanathan, “Integrated identification of the nonlinear autoregressive models with exogenous inputs (NARX) for Engineering Systems Design,” *IEEE Transactions on Control Systems Technology*, vol. 31, no. 1, pp. 394–401, 2023. doi:10.1109/tcst.2022.3171130

APPENDIX 1

This appendix describes the notation used throughout the paper. Vectors are denoted with underlined lower-case letters ($\underline{a}$), matrices and sets with uppercase letters (A) and scalars with lower-case letters (a). Set and tensor constructors are $\{a_j\}_{j=1}^{n} \coloneqq \{a_1, \dots, a_n\}$, $[a_j]_{j=1}^{n} \coloneqq [a_1, \dots, a_n]$ (row vectors) and $[\underline{a}_j]_{j=1}^{n}$(matrices). They also support index sets: $[\underline{a}_j]_{j \in S} \triangleq [\underline{a}_{s_1}, \dots, \underline{a}_{s_{|S|}}]$ with $S \triangleq \{s_j\}_{j=1}^{|S|}$ and $|S|$ the set cardinality. Be further $[a_j]_{j=1}^{n,\downarrow} \coloneqq \left([a_j]_{j=1}^{n}\right)^T$ a vertical ($\downarrow$) tensor constructor and the horizontal concatenation operator $\Vert$. For disambiguation, elementwise or broadcasted tensor arithmetic operation symbols are in a circle such as $\underline{a} \oslash \underline{b} \coloneqq [a_j/b_j]_{j=1}^{\dim(\underline{a})=\dim(\underline{b}),\downarrow}$, $\underline{a} \oslash b \coloneqq [a_j/b]_{j=1}^{\dim(\underline{a}),\downarrow}$ and $\underline{a}^{\odot n} \coloneqq [a_j^n]_{j=1}^{\dim(\underline{a}),\downarrow}$. Tensor slicing is denoted with vector indexation such as $A\left[[j]_{j=1}^{s}, s\right] \coloneqq [a_{j,s}]_{j=1}^{s,\downarrow}$. Matrix columns are denoted by lower-casing the matrix name and adding their index such as $\underline{\omega}_j \in \Omega$ and $\underline{d}_{\mathrm{S},j} \in D_{\mathrm{S}}$. Set union and set differences are respectively denoted with $\mathrm{A} \cup \mathrm{B}$ and $\mathrm{A} \setminus \mathrm{B}$.

APPENDIX 2

This appendix proves that Equation (2)’s recursive form corresponds to the desired Gram-Schmidt orthogonalization or equivalently to a multiplication by the proposed orthonormal annihilator matrix in Equation (1).

First, it must be proven that $\Omega^{(s)}$’s columns always correspond to those in $[\underline{\varphi}_m]_{m \in U^{(s)}}$ (**A**), and secondly that the correct operations are performed (**B**).

Without loss of generality, this proof assumes $D_{\mathrm{S}} = \emptyset$. $D_{\mathrm{S}} \neq \emptyset$ only changes the number of terms in the orthogonalization sum and the number of columns in Ψ and Ψ_{n}.

Further, all indices ($\_^{(s)}$) represent the data-structures’ state after respectively algorithm 1’s step 2.h) and 3.l).

$\underline{\psi}_s$ are always orthogonal w.r.t. all previous $\underline{\psi}$, even if not directly taken from Ω.

A) [EDITED]
B) [EDITED]

APPENDIX 3

This appendix contains the respective expansions denominators, being the best results of multiple runs on different input random sequences. The expansions have the property that higher order terms have tendentially increasingly smaller coefficients, which increases the numerical stability around the origin.

The optimal input noise sequence $\underline{x}$’s amplitude determined by a grid search is respectively 2.36, 2.51, and 2.45 for the orders 5, 6, and 7.

[EDITED]

STÉPHANE J.P.S. THUNUS received his B.S. in sound engineering from SAE Institute, Glasgow, UK in 2017 with distinction. He received his M.S. from the audio communication group at the Technical University of Berlin, Germany in 2022 also with distinction. His research interests include digital signal processing, machine learning, linear algebra, and non-linear stochastic processes.

JULIAN PARKER
Has been Parkering Julians since ever.

STEFAN WEINZIERL
Has been Weinzierling Stefans since ever.