Grammar-based representation and identification of dynamical systems

Grammar-based Representation and Identification of Dynamical Systems

Dhruv Khandelwal, Maarten Schoukens and Roland Tóth

Abstract— **In this paper we propose a novel approach to identify dynamical systems. The method estimates the model structure and the parameters of the model simultaneously, automating the critical decisions involved in identification such as model structure and complexity selection. In order to solve the combined model structure and model parameter estimation problem, a new representation of dynamical systems is proposed. The proposed representation is based on Tree Adjoining Grammar, a formalism that was developed from linguistic considerations. Using the proposed representation, the identification problem can be interpreted as a multi-objective optimization problem and we propose an Evolutionary Algorithm-based approach to solve it. A benchmark example is used to demonstrate the proposed approach. The achieved performance of the proposed method, without making use of knowledge of the system description, was comparable to that obtained by state-of-the-art non-linear system identification methods that do take advantage of correct selection of model structure and complexity based on a priori information.**

I. INTRODUCTION

The problem of inferring models from data has been well studied in many research domains including systems and control. Modelling of complex systems using first principle laws and relations is often too cumbersome to be accomplished in practice. This led to the development of several data-driven grey-box and black-box modelling approaches [1]. Each of these methods have been developed and tuned to identify a well specified class of dynamical systems described by the so-called *model class* associated with the method [1]. However, they are, in general, not well-equipped to identify systems that do not belong to the assumed model class. As a result, most identification procedures require an expert practitioner to make several critical choices and ensure the validity of key assumptions, including model class, complexity and noise structure, to successfully complete a data-driven modelling task [1].

Due to these critical user choices, even for a skilled practitioner, the task of modelling complex systems remains arduous, making automation of these choices highly challenging. In this contribution, we develop a framework for system identification that can function across different model classes and automatically explores varying levels of complexity. In order to realize a method that can work across different classes of models, we propose a new representation for stochastic dynamical models. The proposed representation uses Tree Adjoining Grammar (TAG) [2] - a tree-generating methodology originating from computational linguistics. As we show, the use of TAG-based representation allows one to express a dynamic model in terms of a set of fundamental building blocks called *elementary trees*. These building blocks can be combined in specific ways in order to build more complex models. The advantage of using TAG-based representations is that a given set of elementary trees may be used to generate models that belong to different classes of dynamical systems. This allows the proposed identification framework to function across different model classes.

In the context of unknown model structure and complexity, the problem of inferring models from data can be divided into two components: i) the search of the *appropriate* model structure and complexity, and ii) optimization of the model parameters. The former is a combinatorial optimization problem, while the latter is a continuous optimization problem, which may or may not be convex, depending on the structure of the model [1]. In order to solve the combinatorial problem, we use an algorithm based on Genetic Programming (GP) and TAG. TAG makes the combinatorial search more efficient by restricting the search space explored by GP. Moreover, since the combinatorial search is formulated on the set of elementary trees of a TAG and *not* a particular model structure, the GP-based algorithm for structure estimation runs independent of the model structure.

In order to effectively illustrate the idea, in this paper, we consider the problem of identifying models across model structures that belong to the superset of SISO (Single-Input Single-Output) polynomial NARX (Non-linear Auto-Regressive with eXogenous inputs) model class [3]. The polynomial NARX class contains a number of commonly-used model structures such as FIR, ARX and truncated Volterra series. We propose a TAG that generates polynomial NARX models. We show that the proposed TAG can be scaled down to restrict the scope of the identification task. Similarly, it is also possible to scale up the proposed TAG. This task will be taken up in future research.

Evolutionary Algorithms (EAs) have been previously used for structure determination in non-linear System Identification (SI) [4]–[9]. However, these methods are developed for pre-specified model structures, and EAs are used to estimate the appropriate complexity. This makes it difficult to automate the aforementioned approaches for different model structures. In contrast to the existing literature, our proposed EA-based SI method uses TAG-based representations. TAG enables EA-based structure determination across multiple

This research is supported by the Dutch Organization for Scientific Research (NWO, domain TTW, grant: 13852) which is partly funded by the Ministry of Economic Affairs.

All authors are with the Control Systems group, Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands.

Corresponding author: D.Khandelwal@tue.nl

model classes, thereby automating the SI process. TAG also allows the user to incorporate prior information, if any, within EA-based SI.

Previous attempts to use TAG in a data-driven modelling context has been made in [10] to estimate static models for a curve-fitting problem. However, the extension of the idea to dynamical systems, as proposed in this paper, is far from trivial. For instance, the presence of noise in the data may introduce bias in the estimates if the noise contributions are not treated carefully. In this paper, we extend TAG representation to stochastic dynamical models.

II. Tree Adjoining Grammar of Dynamical Models

A. Preliminaries

TAG [2] is a tree generating system that was initially developed in order to capture features of natural languages that could not be captured by Context Free Grammars (CFG). Several formal properties can be attributed to TAG and can be found in [11]. To develop our contribution, we briefly introduce some key ingredients of TAG. For a more detailed treatment of TAG, see [11] and [12].

Let $\gamma = \langle V, E, r\rangle$ denote a *finite tree*, where V is the set of vertices, E is the set of edges, and $r \in V$ is the root node. A vertex of a finite tree with out-degree 0 (i.e., number of outgoing edges) is called a *leaf*. Introduce the labelling function of a tree $l : V \rightarrow A$ that maps from the set of vertices V of a tree to an alphabet (i.e., set of symbols) A.

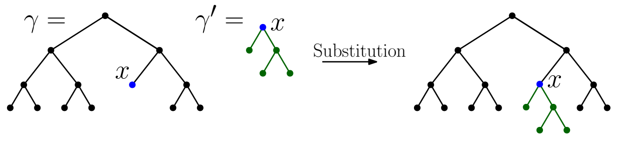
Definition 1 (Tree Adjoining Grammar): A Tree Adjoining Grammar G is a tuple $\langle N, T, S, I, A\rangle$, where

- N is an alphabet of non-terminal symbols;
- T is an alphabet of terminal symbols;
- S is a specific start symbol in N;
- I is a set of initial trees. An initial tree $\gamma = \langle V, E, r\rangle$ has $l(v_{\text{int}}) \in N$ for all internal vertices $v_{\text{int}} \in V$ and $l(v_{\text{leaf}}) \in (N \cup T) \setminus \{l(r)\}$ for all leaves $v_{\text{leaf}} \in V$;
- A is a set of *auxiliary trees*. A tree $\gamma = \langle V, E, r\rangle$ is an auxiliary tree iff $l(v_{\text{int}}) \in N$ for all internal vertices $v_{\text{int}} \in V$ and $l(v_{\text{leaf}}) \in (N \cup T)$ for all leaves $v_{\text{leaf}} \in V$ and there is a unique leaf $f \in V$ with $l(f) = l(r)$. The node f is called the *foot node*. The auxiliary tree is denoted as $\langle V, E, r, f\rangle$.

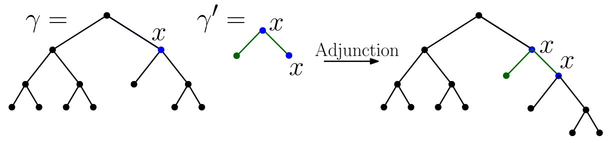
The set of trees $I \cup A$ is called *elementary trees*. A *syntactic tree* is a tree $\gamma = \langle V, E, r\rangle$ that satisfies $l(r) \in S$, $l(v_{\text{int}}) \in N$ for all internal vertices $v_{\text{int}} \in V$ and $l(v_{\text{leaf}}) \in (N \cup T)$ for all leaves $v_{\text{leaf}} \in V$. A syntactic tree is said to be *saturated* if all leaves $v_{\text{leaf}} \in V$ satisfy $l(v_{\text{leaf}}) \in T$. The set of all finite saturated trees of a TAG G is called the *tree language* $L_{\text{T}}(G)$ of G. The *yield* of a saturated tree is the string of labels of the leaves of the tree. The set of yields of the trees in $L_{\text{T}}(G)$ is called the string language $L(G)$.

The TAG framework provides two operations, *substitution* and *adjunction*, that can be used to generate or modify tree structures with a TAG G.

- *Substitution*: substitute a leaf v_{leaf} in a syntactic tree $\gamma = \langle V, E, r\rangle$ with an initial tree $\gamma' = \langle V', E', r'\rangle \in I$ iff $l(r') = l(v_{\text{leaf}})$. See Fig. 1a.



(a) TAG substitution operation.



(b) TAG adjunction operation.

Fig. 1: Illustration of the TAG operations.

- *Adjunction*: insert an auxiliary tree $\gamma' = \langle V', E', r', f'\rangle \in A$ at an internal vertex v_{int} of a syntactic tree $\gamma = \langle V, E, r\rangle$. The adjunction operation is defined iff $l(v_{\text{int}}) = l(r')$. Adjunction takes place in three steps. First, detach the subtree $\gamma'' = \langle V'', E'', v_{\text{int}}\rangle$ starting at the internal node v_{int}. Subsequently, substitute the foot node f' with the tree $\langle V'', E'', v_{\text{int}}\rangle$. This is valid since $l(v_{\text{int}}) = l(r') = l(f')$. Finally, insert the new syntactic tree in the original tree in place of the the internal vertex v_{int}. See Fig. 1b.

B. TAG representation of polynomial NARX models

The discrete-time polynomial NARX model class can be represented as

$$y_k = \sum_{i=1}^{p} \theta_i \prod_{j=0}^{n_u} u_{k-j}^{b_{i,j}} \prod_{m=1}^{n_y} y_{k-m}^{a_{i,m}} + \xi_k, \tag{1}$$

where p is the number of model terms, $u_k, y_k \in \mathbb{R}$ are values of input and output signals at time instant k, ξ_k is a white noise process, θ_i are the model parameters, and $a_{i,m}, b_{i,j} \in \mathbb{Z}_{\geq 0}$ are the exponents of the signal values. In (1), we can observe a hierarchy in the structure of the Right Hand Side (RHS) of the model expression, summarized as follows

- a model expression consists of a sum of *terms*, i.e., parts of the expression that are connected by addition. Addition and subtraction are "equivalent" operators.
- each term is a multiplication of *factors*, i.e., parts of the expression that are connected by multiplication. These factors may be real parameters, input or output signals,
- each input and output factor may contain delays.

We assign the labels `expr0`, `expr1`, `expr2` for each of the three levels of hierarchy. Furthermore, we use `op`, `aff` and `par` as labels for operators, the affine noise term ξ_k and real coefficients θ_i, respectively. For convenience, the time index k is dropped. This should not lead to ambiguity since delays will be explicitly denoted by the backward shift operator q^{-1}. We propose the following TAG representation for (1).

Proposition 1: The TAG of polynomial NARX models is $G_{\text{NARX}} = \langle N, T, S, I, A\rangle$ with

- $N = \{\text{expr0}, \text{expr1}, \text{expr2}, \text{op}, \text{aff}, \text{par}\}$,

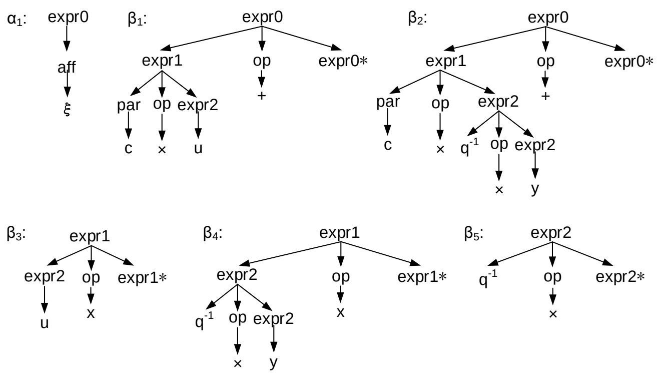


Fig. 2: Initial tree $I = \{\alpha_1\}$ and auxiliary trees $A = \{\beta_i\}_{i=1}^5$ of the TAG G_{NARX}. Symbol $*$ marks the foot node of the auxiliary tree.

- $T = \{\mathrm{u}, \mathrm{y}, \xi, \mathrm{c}, +, \times, q^{-1}\}$,
- $S = \{\mathrm{expr0}\}$,
- $I = \{\alpha_1\}$, with the initial tree α_1 depicted in Fig. 2,
- $A = \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5\}$, with the auxiliary trees $\{\beta_i\}_{i=1}^5$ depicted in Fig. 2.

The string language $L(G_{\mathrm{NARX}})$ of grammar G_{NARX} is the set of all expressions that can be expressed as the RHS of (1) with finite values of p, n_u, n_y.

For brevity, we provide a short sketch of the proof here. The simplest saturated tree that can be generated from G_{NARX} is the initial tree α_1. The yield of the tree is ξ which corresponds to the simplest model that can be generated:

$$y_k = \xi_k. \tag{2}$$

Eqn. (2) can be augmented by adjoining it with auxiliary trees from A. Observe that the auxiliary trees have the same structure as the hierarchy in (1). Adjunction with β_1, β_2 results in the addition of new input or output terms to the expression, parameterized by a place-holder parameter c. The next level of hierarchy is the multiplication with an arbitrary, but finite number of input and output factors, achieved by adjunction of β_3, β_4. Finally, adjunction of auxiliary tree β_5 can introduce an arbitrary, but finite number of delays to each factor. Hence, we can conclude that for any NARX model in the from of (1) we can generate a tree satisfying the construction rules of G_{NARX} such that the resulting tree is saturated and its yield is the RHS of (1). Similarly, we can show that any saturated tree in $L_{\mathrm{T}}(G_{\mathrm{NARX}})$ has a yield which is in the form of (1).

C. Discussion

Representation of dynamical systems using the proposed G_{NARX} has some interesting implications from both the system theory and the system identification perspectives.

1) Model representation: TAG provides a new representation for dynamical systems, based on the initial and auxiliary trees used to construct the model. An example is depicted in Fig. 3. In the example, the tree labelled (A) depicts the adjunctions used to construct the model, starting from the initial tree α_1. The labels on the edges are the Gorn addresses (see [13]) of the vertices at which the adjunction takes place.

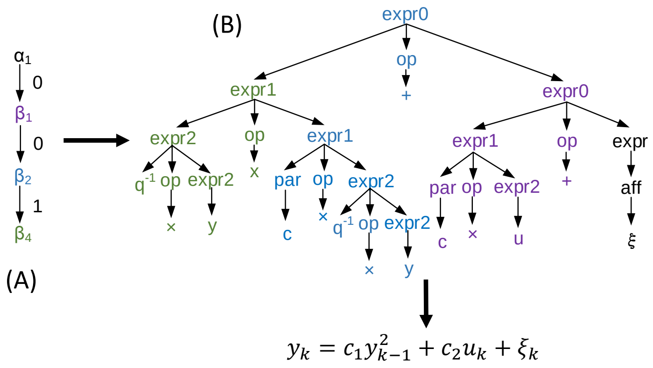


Fig. 3: Illustrative example - TAG representation of a NARX model.

In TAG terminology, this is called the *derivation tree*. The tree labelled (B) in Fig. 3 is the tree generated using G_{NARX}, and is called the *derived tree*. Notice that the derived tree depicts the model structure on the non-terminal vertices, and the model expression on the leaves.

2) Dynamical sub-classes: The set of auxiliary trees A determine the possible ways in which a model may be developed. Choosing relevant subsets of A of grammar G_{NARX} results in TAG formulations of model sub-classes such as FIR, ARX and truncated Volterra series. For example, choosing the subset $A_1 = \{\beta_1, \beta_5\}$ yields the TAG for FIR models, and the subset $A_2 = \{\beta_1, \beta_2, \beta_5\}$ yields the TAG for ARX models. Similarly, adding suitable auxiliary trees results in TAG for more flexible model classes such as NARMAX. In a data-driven modelling context, this provides a systematic way to introduce prior knowledge of model structure.

3) Ranking of models across various model classes: In system identification, Occam's Razor principle is often used as a guiding heuristic for model selection. A pre-requisite for using this principle is that one should be able to rank various models in terms of some complexity measure. TAG based representations provide an opportunity to rank models that belong to different model classes. However, such a ranking cannot possibly be 1-dimensional, as is usually the case. In order to rank a model, one must take into account not only the dimensions of the TAG representation of the model (for example, the number of vertices in the derivation tree in Fig. 3), but also the specific auxiliary trees used. Various auxiliary trees add to the complexity of a model differently. For example, adjunction of β_3 of G_{NARX} introduces a multiplicative non-linearity, while adjunction of β_1 introduces a new linear term to a model.

III. System Identification using TAG Representations

In this Section, we describe a TAG-based approach to identify dynamical systems from measured data. For any system identification problem, three crucial choices have to be made - the model class, the performance criterion and the algorithm and numerical machinery used for model estimation (as per the chosen model class and performance criterion). Each of these choices are introduced and motivated in this Section.

A. Model class and complexity

Since our aim is to estimate model structure and complexity, the notion of "model class" needs to be made more flexible to include a collection of model structures. The proposed grammar G_{NARX} can be used to generate FIR, ARX, truncated Volterra series or polynomial NARX models of varying complexities. Hence, the notion of "model class" is used to describe the generative capacity of the grammar. In this paper, the chosen model class is the set of all models that can be generated by G_{NARX}. Note that, while the choice of grammar remains a user-choice, it is not as critical as the choice of a specific model class and complexity, since multiple model classes with varying complexities can be generated by the same TAG.

In the proposed identification approach, operations related to the search of model structure will be based on the derivation tree representation, and operations related to parameter optimization will use the symbolic representation (1).

B. Performance criteria

Due to the rich representational capability of TAG, we propose the use of multiple performance criteria in order to measure the quality of proposed models along multiple dimensions. We use 1-step-ahead prediction error to measure short-term prediction capability of the model and to ensure stochastic optimality of the parameter estimates (under certain conditions, see [1]). However, prediction error is typically less sensitive to errors in model structure, which is unavoidable given the generality of the model set introduced in Sec. III-A. Hence, simulation error, which is typically more sensitive to model errors (see [14], [15]), is used to measure long-term prediction capability of the model. Finally, the number of model parameters is used as a complexity measure. The three performance criteria are used in a multi-objective optimization setting (see [16]).

Simulation of a non-linear stochastic model is a non-trivial task. The common approach to simulation is to set the noise contributions to 0. However, it has been shown in [17] that this leads to biased simulation models in the case of non-linear systems. An approach to compute simulation models from stochastic models is proposed in [17], and is used here.

C. Algorithm

This section describes the algorithm for estimation of model structure and model parameters. Since model structure estimation, even under the proposed TAG representation, is a combinatorial problem for which no systematic solution exists, it is reasonable to rely on heuristic solution approaches. Genetic Programming (GP) provides a set of biologically-inspired heuristics that have yielded competitive results in multiple domains of science and engineering (e.g., [18]). GP also provides a numerical platform to solve the multi-objective optimization problem using the notion of non-dominated solutions and pareto-optimality [18]. For these reasons, GP is a natural choice for the estimation of model structure and with TAG, the evolutionary search via GP can be efficiently formulated. See overview in Algorithm 1.

1) Genetic Programming: GP is an iterative scheme that develops and propagates a set of M solutions (the population) iteratively. In each iteration, a new set of population is proposed by using *genetic operators* such as *crossover* and *mutation*. Subsequently, a *selection* scheme is used to select M solutions from the existing and the newly proposed solutions, based on a user-defined performance measure. The selected models are propagated to the next iteration, and this process is repeated for a fixed number of maximum iterations L. See [19] for details.

Algorithm 1 Multi-objective optimization using TAG3P and LS

```
Require: population size $M > 0$, number of iterations $L > 0$, grammar $G$, crossover rate $p_c$, mutation rate $p_c, p_m$
1: Initialize population $X^{(0)}$, $l = 0$, $X^{(-1)} = \{\}$    ▷ See [19]
2: repeat
3:     Estimate parameters in $X^{(l)}$    ▷ See Sec. III-C.2
4:     Compute multi-objective fitness of models in $X^{(l)}$
5:     Perform non-dominated sorting of populations $X^{(l-1)}$ and $X^{(l)}$    ▷ See [16]
6:     $X^{(l)} \leftarrow$ first $M$ individuals of the sorted combined population.
7:     Propose new population $X^{(l+1)}$ using crossover and mutation    ▷ See [20]
8:     $l \leftarrow l + 1$
9: until $l \leq L + 1$ return $X^{(L)}$
```

In the proposed algorithm, we use a variant of GP called Tree Adjoining Grammar-Guided GP (TAG3P) [20], that was developed for TAG formulations. Hence, the genetic operations of crossover and mutation are adapted for derivation tree representation of the model. The search scheme is initialized with M randomly generated derivation trees from G_{NARX}. In each iteration, a non-dominated sorting and selection algorithm, proposed in [16], is used to select and propagate pareto-optimal solutions. Other hyper-parameters involved are the number of iterations L, the probability for crossover p_c and the probability of mutation p_m. A high probability of crossover ensures that sub-structures of models that achieve better performance are transferred to other models in the population. A high probability of mutation allows for more frequent random explorations in the space of models.

2) Parameter estimation: In each iteration of GP, the new solutions contain yet-to-be-determined model parameters. Since TAGs may generate models that belong to different model classes, the parameter estimation approach should be robust enough to deal with variability of model structures. Optimization methods such as CMA-ES (see [21]) can be used to estimate model parameters in a linear or non-linear setting. An alternative, more efficient approach would be to make use of one out of a collection of parameter

TABLE I: Algorithm hyper-parameters

Hyper-parameter	**Value**
Population Size M	100
Maximum GP iterations L	150
Maximum adjunctions	150
Probability of crossover p_c	1
Probability of mutation p_m	0.8
Grammar	G_{NARX}

estimation algorithms, depending on the model structure generated by the grammar. The use of TAG makes it possible to systematically infer the model structure, and hence the appropriate optimization algorithm.

Since the scope of this paper is limited to polynomial NARX models, the parameter estimation problem can be solved efficiently. Minimization of the sum-of-squares of the prediction error of (1) leads to a quadratic cost function that can be solved using LS [1].

IV. Identification results

To illustrate the proposed TAG-based identification procedure, we use the benchmark Silverbox data-set proposed in [22]. The silverbox system in an electronic implementation of a mass-spring-damper system with a non-linear spring. The data-set, measured at a sampling rate of 610.35 Hz, consists of 2 parts. The excitation signal used in the first 40000 samples is a low-pass filtered Gaussian noise signal with a bandwidth of 200 Hz and a linearly increasing amplitude ranging from 0 to a maximum value of about 0.3 V. The data-set is plotted in Fig. 4a. The excitation signal in the second part of the data-set consists of 10 realizations of a random odd multi-sine signal (see [22] for details). We use the first 9 realizations of the multi-sine data-set as *estimation data-set* to estimate model parameters in Step 3, and the last realization as *validation data-set* to evaluate the multi-objective fitness of the proposed models in Step 4. Furthermore, the Gaussian excitation signal associated part of the data-set is used as a *test data-set* that is independent of the remaining data used during the identification procedure, and is used to evaluate the performance of the pareto-front obtained at the end of Alg. 1.

The hyper-parameters used in the algorithm are given in Table I. As there are no guarantees of convergence, the hyper-parameters are chosen conservatively. It should be noted that no other information specific to the benchmark example was incorporated into the identification procedure. The results obtained are plotted in Fig. 4. To plot these figures, the complexity measure was chosen as the number of real parameters in the model. Fig. 4d depicts the evolution of the performance measures (average and minimum over all models in the population) with respect to GP iterations. Fig. 4b and 4c depict 2-D projections of the final pareto-front obtained from the identification procedure, the bold dots indicate the pareto-front of the best models found and the smaller dots indicate some of the sub-optimal models found during the procedure. It should be noted that the bold dots in both Fig. 4b and 4c correspond to the same models for each level of complexity.

TABLE II: Performance if estimated models computed in the test data-set and compared with literature. The blank cells correspond to values not reported in the literature.

Identified model	Test RMS simulation (mV)	Test RMS prediction (mV)	Val. RMS simulation (mV)	Val. RMS prediction (mV)
M_1	1.8046	0.4525	1.235	0.2941
M_2	0.4196	0.1017	0.2698	0.0731
Best Linear Approx. [24]	13.5	-	6.9	-
PNLSS [24]	0.26	-	-	-

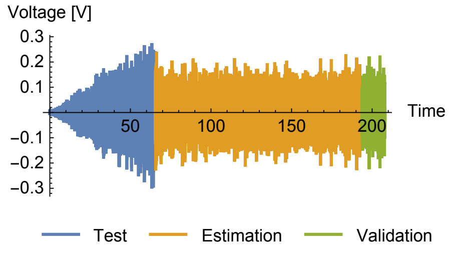
From Fig. 4b and 4c we observe that beyond the complexity level of 6 (i.e. 6 model parameters), the improvement in both performance measures is only gradual. Furthermore, the model that achieves the best performance in both measures contains 27 parameters. For the sake of analysis and comparison, we select two models from the pareto-front - M_1 corresponding to the best model containing 6 parameters, and M_2 corresponding the best model with 19 parameters. Model M_1 is described by the following equation

$$y_k = 0.3694u_{k-1} + 0.0467u_k + 0.1024y_{k-3} - 1.0939y_{k-2} + 1.5809y_{k-1} - 1.3923y_{k-1}^3 + \xi_k. \quad (3)$$

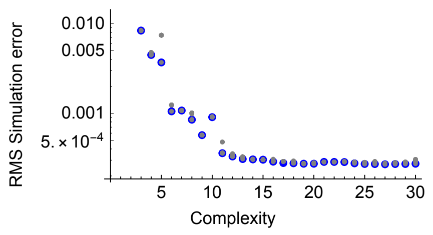
The structure of M_1 is the same as the structure of the ideal circuit as reported in [23]. However, in [23], it was also reported that the realization of the electronic circuit was non-ideal, which explains the steady improvement in the pareto-fronts in Fig. 4b and 4c beyond the model complexity of 6. Fig. 4e and 4f depicts the prediction and simulation errors (scaled up by a factor of 5) of models M_1 and M_2. We can observe in Fig. 4f that M_2 performs adequately well also in the latter half of the test data-set where the model is required to extrapolate (since the magnitude of the input is greater than that in the estimation and validation set). The performance metrics of M_1 and M_2 are given in Table II. The performance of the proposed approach is comparable to that of state-of-the-art non-linear system identification methods. The results obtained by PNLSS identification [24] are also provided in Table II for comparison. While the PNLSS method produces a single model, in this case with 37 parameters, the proposed approach provides a pareto-front of models. This enables the user to choose the performance-complexity trade-off *a-posteriori*. Furthermore, in the proposed method, the user is not required to make any critical choices, while in the case of PLNSS, the user must make critical decisions, such as the order of non-linearity considered and the initialization method.

V. Conclusions

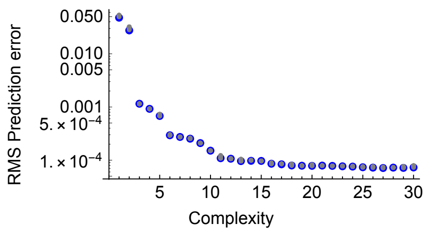
We proposed a new identification procedure that uses TAG-based representations to identify both the structure and the parameters of a model. The proposed framework is formulated on auto-regressive forms ranging from linear to (non-linear) polynomial models. It is demonstrated on a benchmark data-set that, without prior information on the system dynamics, the proposed method was able to correctly estimate an efficient choice of the model structure and also achieve comparable estimation results as those achieved by



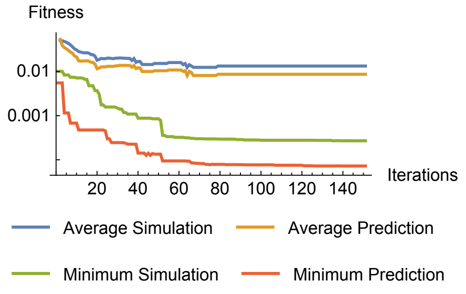
(a) Silverbox measured data.



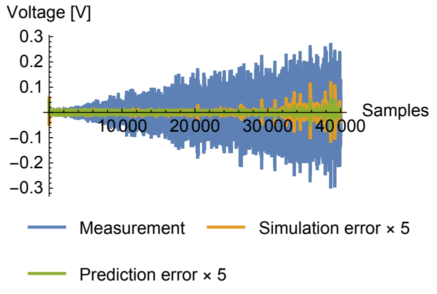
(b) Pareto front - simulation vs complexity.



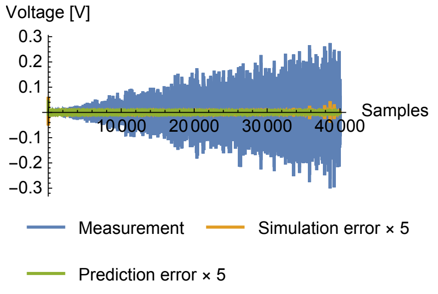
(c) Pareto front - prediction vs complexity.



(d) Evolution of performance measures over iterations.



(e) Evaluation of model M_1 on test data-set.



(f) Evaluation on model M_2 on test data-set.

Fig. 4: Illustration of numerical results.

user-assisted identification methods. TAG-based representation of models is vital to this approach as it provides a mechanism to isolate the numerical algorithm from the choice of model class, thereby allowing systematic exploration of model structures among different model classes. While the scope of the paper was restricted to illustrate the idea efficiently, it is certainly possible to extend it to general non-linear system descriptions and beyond.

REFERENCES

[1] L. Ljung, Ed., *System Identification (2nd Ed.): Theory for the User*. Prentice Hall PTR, 1999.

[2] A. K. Joshi, "An introduction to tree adjoining grammars," *Mathematics of language*, vol. 1, pp. 87–115, 1987.

[3] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[4] C. M. Fonseca and P. J. Fleming, "Non-linear system identification with multiobjective genetic algorithms," in *Proc. of 13th IFAC World Congress*, 1996, pp. 1169–1174.

[5] K. Rodríguez-Vázquez and P. J. Fleming, "Use of genetic programming in the identification of rational model structures," in *In Proc. of European Conference on Genetic Programming*. Springer, 2000, pp. 181–192.

[6] J. Madár, J. Abonyi, and F. Szeifert, "Genetic programming for the identification of nonlinear input- output models," *Industrial & engineering chemistry research*, vol. 44, no. 9, pp. 3178–3186, 2005.

[7] M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack, "Prediction of dynamical systems by symbolic regression," *Physical Review E*, vol. 94, no. 1, p. 012214, 2016.

[8] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 5, pp. 1033–1046, 1992.

[9] K. Worden, R. Barthorpe, E. Cross, N. Dervilis, G. Holmes, G. Manson, and T. Rogers, "On evolutionary system identification with applications to nonlinear benchmarks," *Mechanical Systems and Signal Processing*, vol. 112, pp. 194–232, 2018.

[10] N. X. Hoai, R. I. McKay, D. Essam, and R. Chau, "Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: The comparative results," in *Proc. of the 2002 Congress on Evolutionary Computation, 2002*, vol. 2, 2002, pp. 1326–1331.

[11] A. K. Joshi and Y. Schabes, "Tree-adjoining grammars," in *Handbook of formal languages*. Springer, 1997, pp. 69–123.

[12] L. Kallmeyer, "A declarative characterization of different types of multicomponent tree adjoining grammars," *Research on Language and Computation*, vol. 7, no. 1, pp. 55–99, 2009.

[13] S. Gorn, "Explicit definitions and linguistic dominoes," Univ. of Western Ontario, pp. 77–115, 1965.

[14] L. Piroddi and W. Spinelli, "An identification algorithm for polynomial narx models based on simulation error minimization," *International Journal of Control*, vol. 76, no. 17, pp. 1767–1781, 2003.

[15] L. A. Aguirre, B. H. Barbosa, and A. P. Braga, "Prediction and simulation errors in parameter estimation for nonlinear systems," *Mechanical Systems and Signal Processing*, vol. 24, no. 8, pp. 2855–2867, 2010.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[17] D. Khandelwal, M. Schoukens, and R. Tóth, "On the Simulation of Polynomial NARMAX Models," *ArXiv e-prints*, Oct. 2018.

[18] A. Arias-Montano, C. A. Coello, and E. Mezura-Montes, "Multiobjective evolutionary algorithms in aeronautical and aerospace engineering," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 662–694, 2012.

[19] J. R. Koza, *Genetic programming II, automatic discovery of reusable subprograms*. MIT Press, Cambridge, MA, 1992.

[20] N. X. Hoai, R. I. McKay, and H. A. Abbass, "Tree adjoining grammars, language bias, and genetic programming," in *European Conference on Genetic Programming*. Springer, 2003, pp. 335–344.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *Control Conference (ECC), 2013 European*. IEEE, 2013, pp. 2933–2938.

[23] J. Schoukens, J. G. Nemeth, P. Crama, Y. Rolain, and R. Pintelon, "Fast approximate identification of nonlinear systems," *Automatica*, vol. 39, no. 7, pp. 1267–1274, 2003.

[24] A. Marconato, J. Sjöberg, J. Suykens, and J. Schoukens, "Identification of the silverbox benchmark using nonlinear state-space models," in *IFAC Proceedings. 16th IFAC Symposium on System Identification*, vol. 16, no. 1, 2012, pp. 632–637.