# A Tree Adjoining Grammar representation for models of stochastic dynamical systems

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](Link to publication)

Brief paper

# A Tree Adjoining Grammar representation for models of stochastic dynamical systems☆

Dhruv Khandelwal *, Maarten Schoukens, Roland Tóth

*Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands*

## ARTICLE INFO

## ABSTRACT

Model structure and complexity selection remains a challenging problem in system identification, especially for parametric non-linear models. Many Evolutionary Algorithm (EA) based methods have been proposed in the literature for estimating model structure and complexity. In most cases, the proposed methods are devised for estimating structure and complexity within a specified model class and hence these methods do not extend to other model structures without significant changes. In this paper, we propose a Tree Adjoining Grammar (TAG) for stochastic parametric models. TAGs can be used to generate models in an EA framework while imposing desirable structural constraints and incorporating prior knowledge. In this paper, we propose a TAG that can systematically generate models ranging from FIRs to polynomial NARMAX models. Furthermore, we demonstrate that TAGs can be easily extended to more general model classes, such as the non-linear Box–Jenkins model class, enabling the realization of flexible and automatic model structure and complexity selection via EA.

## 1. Introduction

In recent years, there has been a resurgence in the use of Evolutionary Algorithms (EAs) for data-driven modelling of dynamical systems. Undoubtedly, one of the main driving forces for this is the steady growth of computation power. EAs are being increasingly used in a multitude of engineering domains and life science (Eiben et al., 2003; Arias-Montano et al., 2012). Across several domains, EAs have generated results that are competitive and, sometimes, even surprising (Eiben et al., 2003). Another factor contributing to the growing popularity of EAs is that these algorithms can be used to generate solutions for complex problems for which no systematic solution approach exists in general. In parametric system identification, the estimation of model structure and model complexity is one such problem.

Model structure selection is a classical problem in system identification. Over the years, a variety of methods for system identification have been developed. Each of these methods adopt different approaches to solve the problem of model structure selection. While methods like Prediction Error Minimization (PEM) treat model structure selection as a user's choice (Ljung, 1999), other methods (for example, Pillonetto et al. (2011), Laurain et al. (2020)) rely on a flexible model structure, and attempt to estimate or control the complexity of the model-to-be-estimated via regularization. Furthermore, the appropriate model complexity is often chosen by ranking models based on an information metric, such as AIC, BIC, or based on a user-defined complexity measure (Rojas et al., 2014). In cases where the number of candidate models grows combinatorially with respect to the length (or the complexity) of the model, a ranking-based complexity selection strategy becomes intractable, restricting model structure selection to regularization or shrinkage based methods.

As a consequence of the aforementioned challenges, heuristics-based methods such as EAs have been used to estimate model structure and complexity, with a fair amount of success. However, the application of EAs have been, to some extent, superficial. The premise of the biologically-inspired heuristics used in EAs is that the solutions of a given problem can be constructed from fundamental building blocks, and these fundamental components can be interchanged between different solutions. In the system identification literature, the proposed EA-based approaches to model structure and complexity selection can be categorized as follows:

(i) approaches that choose a fixed model structure and use EAs to determine the appropriate model complexity (or model terms), and

E-mail addresses: D.Khandelwal@tue.nl (D. Khandelwal), M.Schoukens@tue.nl (M. Schoukens), R.Toth@tue.nl (R. Tóth).

(ii) approaches that use EAs to explore model structure and model complexity.

In the first category of EA-based approaches, the basic building blocks of an EA are chosen such that only models with a specific model structure can be generated. Hence, these approaches cannot be typically extended to other model structures without significant modifications. This approach can be found in Fonseca and Fleming (1996), Rodriguez-Vazquez et al. (2004) and Rodríguez-Vázquez and Fleming (2000), where the authors use EAs to perform term selection within a chosen model structure. This approach is also used in Kristinsson and Dumont (1992), where the authors use GAs to estimate pole-zero locations for ARMAX models.

In the second category of EA-based approaches, more generic set of building blocks are used in the EA, allowing the generation of models with arbitrary model structures. In this case, EAs are used to determine not just the appropriate complexity of the model, but also the appropriate model structure (e.g., in terms of the non-linear functions to be included in the model). However, unrestrained generation of arbitrary model structures using EA may result in models that are not well-posed, e.g., models with discontinuities, non-causality, or finite escape-time. Typically, these problems are avoided by using arbitrary ad-hoc solutions, e.g, setting all discontinuities to 0. Another common drawback of EA-based approaches that fall in the second category is that prior knowledge of the dynamical system cannot be incorporated systematically in the identification procedure. In Madár et al. (2005), the authors use GP to identify NARMAX models that may contain arbitrary non-linearities. While the authors are interested in models that are linear-in-the-parameters, GP may return models that do not belong to that class. Consequently, the authors use an ad-hoc solution to ensure that the candidate model structures generated by GP are linearly parameterized. A similar approach was used in Quade et al. (2016) with a larger set of mathematical operations. Again, the proposed approach does not allow for systematic inclusion of model structure constraints or prior knowledge of the system. A slightly different approach is used in Gray et al. (1998), where the authors use GP to construct linear or non-linear models from basic elements like SIMULINK blocks and static non-linearities. Again, the combination of various SIMULINK blocks cannot be systematically structured to avoid ill-posed models.

In this paper, we propose a generative grammar based representation of stochastic parametric dynamical systems. The proposed representation allows for the generation of complex, yet well-posed dynamical models by combining a set of fundamental building blocks in well-specified ways. The resulting generative declaration of models defines a notion of *model set* that is more generalized than that conventionally used, for example, in Ljung (1999). The generative grammar used in this work is called Tree Adjoining Grammar (TAG) (Joshi & Schabes, 1997). The use of TAG in an EA-based approach makes it possible to develop a system identification framework where EAs are used to automatically determine the structure and complexity of a model from a generic, well-posed class of dynamical models, while systematically incorporating model structure constraints and prior knowledge. A preliminary concept of the proposed framework (without proofs) was presented in Khandelwal et al. (2019b). The proposed approach for grammar-based identification was found to produce results that were comparable to state-of-the-art non-linear system identification approaches, while using no specialized knowledge of the benchmark system being identified.

The main contributions of this paper are the following. We present a detailed discussion on the discrete-time input–output representation of dynamical systems using TAG, and introduce a new notion of a model set defined by the generative capacity of a TAG. Subsequently, we develop a TAG for the polynomial NARMAX model class. We prove that any model structure generated by the proposed TAG belongs to the class of polynomial NARMAX models, and conversely, any polynomial NARMAX model can be represented using the proposed TAG (for which an algorithm is also proposed). We demonstrate that the model set corresponding to the proposed TAG includes, as special cases, other commonly used model structures such as FIR, ARX and Truncated Volterra series models. We also demonstrate that the proposed representation can be easily extended to other model structures (namely polynomial Non-linear Box–Jenkins, or NBJ). Note that, while the TAG-based model set notion developed in this contribution is motivated by its applicability in an EA-based identification methodology, the identification approach itself is *not* in the scope of the present contribution. A preliminary version of such an identification methodology can be found in Khandelwal et al. (2019a, 2019b).

The contributions in this paper differ from Khandelwal et al. (2019b) in the following respects:

- we formulate a TAG for a larger class of dynamical systems (the polynomial NARMAX class), and prove their equivalence,
- we provide an algorithm to compute an equivalent TAG representation of a given polynomial NARMAX model,
- we illustrate, via examples, the restriction (and generalization) of the proposed TAG in order to generate models with more specific (or generic) structures.

The remainder of the paper is structured as follows. The concept of TAG is introduced, both informally and formally, in Section 2. In Section 3 we introduce the notion of model set as defined by a given TAG, and propose a TAG that generates the class of polynomial NARMAX models. Several examples are used to illustrate the concept in Section 4, followed by concluding statements in Section 5.

## 2. Tree Adjoining Grammar

To set the stage for the development of TAG for stochastic non-linear systems, first we introduce the basic concepts of TAG. Since TAG was initially developed from linguistic considerations, a linguistic example will be used to illustrate the methodology. This will be followed by formal definitions. To make the example illustrative, we first specify an example string, and then infer a TAG that would generate the given string. Conversely, for the formal definitions, we will begin with the basic components of a TAG and lead up to the definition of TAG and operations that can be performed on TAGs.

### 2.1. An informal description

Informally, a *formal grammar* can be described as a set of rules for generating strings. The resulting set of strings is called the *language* generated by the grammar. In contrast, TAG describes a set of rules for generating trees. The resulting set of trees is called the *tree language* of the TAG. The *yield* of all the trees in the tree set subsequently determines the corresponding language.

The following example has been derived from Joshi and Schabes (1997). Consider the sentence "A man saw Mary". Simple grammatical constructs can be used to decompose the given sentence into its basic components. For example, the sentence consists of articles ("A"), nouns ("man", "Mary") and verbs ("saw"). Other underlining structures, such as subjects and predicates, can also be observed in the sentence. The sentence, together with the underlying grammatical structure can be represented in a single tree structure as shown in Fig. 1. The tree depicted in Fig. 1 is
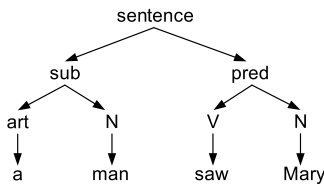
**Fig. 1.** A derived tree with the yield "A man saw Mary". The tree depicts the grammatical constructs that are evident in the structure of the sentence — a subject (sub) and a predicate (pred), an article (art), a verb (V) and nouns (N).
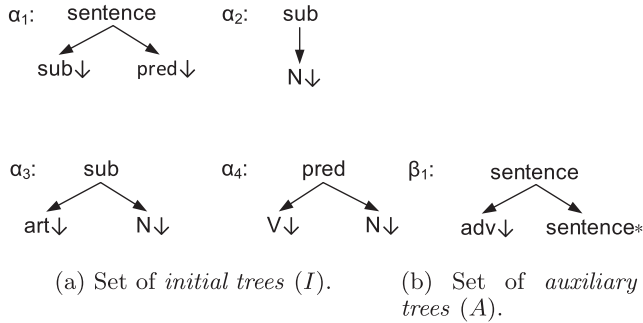


(a) Set of *initial trees* ($I$).  (b) Set of *auxiliary trees* ($A$).

**Fig. 2.** The sets $I$ and $A$ serve as building blocks of the tree set of a TAG.



(a) Derivation tree for "a man saw Mary".

(b) Derivation tree for "yesterday a man saw Mary".

**Fig. 3.** Derivation tree representation — dashed lines represent substitutions, solid lines represent adjunction, and labels on the edges represent the Gorn addresses (a method to assign a label to a node in a tree structure, see Gorn (1965)) of the nodes participating in substitution or adjunction.

called a *derived tree*. The *yield* of a derived tree are the labels associated with the leaves of the tree. Hence, the yield of the derived tree in Fig. 1 is "A man saw Mary".

The given derived tree can be obtained by combining basic building blocks that are constituents of the TAG. Fig. 2 depicts the set of initial trees $I$ and auxiliary trees $A$, collectively known as *elementary trees*, that can be combined in specific ways to produce the derived tree in Fig. 1. The set of initial trees $I$ can be informally described as a set of non-recursive replacement rules that can be used to generate a set of trees. The set of auxiliary trees can be described as a set of recursive replacement rules. Consequently, each auxiliary tree has a terminal node with the same label as that of its root node.

The downward arrow symbol $\downarrow$ and the star symbol $\star$ in Fig. 2 represent nodes in a tree that are available for a *substitution* and *adjunction* operation respectively. A substitution operation can be used to substitute an initial tree into, for instance, another initial tree, if and only if the latter has a terminal node (leaf) with a label that matches the label of the root node of the prior. On the other hand, adjunction can be loosely described as the operation of inserting an auxiliary tree into a syntactic tree. Adjunction of an auxiliary tree can take place on a non-terminal node of a syntactic tree if and only if the node has a label that matches the label of the root node of the auxiliary tree to be adjoined.

Consider the following sequence of operations. The initial tree $\alpha_3$ can be substituted in $\alpha_1$ at the location of the "sub" node. Let us denote the resulting tree as $\gamma_1$. The tree $\gamma_1$ is an example of a *syntactic tree*, a tree obtained by applying an arbitrary number of substitution and adjunction operations to a given initial tree. Again, the initial tree $\alpha_4$ can be substituted to the syntactic tree $\gamma_1$ at the location of the "pred" node. Let the result be denoted as $\gamma_2$. Note that $\gamma_2$ has the same structure as the example in Fig. 1, up to the last level of the derived tree, where specific articles, nouns and verbs are substituted in the tree to obtain the yield "a man saw Mary". Substitution can be performed on a initial tree or syntactic tree as long as there exist nodes available for substitution, marked by $\downarrow$. A *derived tree* is a syntactic tree in which none of the terminal nodes (leaves) are available for substitution. The initial and auxiliary trees provide an alternative

representation, the *derivation tree*, as shown in Fig. 3a. Based on the TAG in Fig. 2, more complex sentences can also be generated. For example, the auxiliary tree $\beta_1$ can be adjoined to the root node of $\gamma_2$ since both root nodes have the label "sentence". This operation effectively adds an adverb before the sentence, yielding the sentence "yesterday a man saw Mary". The resulting derivation tree is depicted in Fig. 3b.

The set of all derived trees that can be obtained, by starting from a given start symbol, say "sentence", and applying an arbitrary number of adjunctions and/or substitutions using elementary trees is called the *tree language* of the corresponding TAG. The string yield of all trees in the tree set is called the *string language* of the corresponding TAG.

We can now introduce the formal definitions of the concepts that were informally described in this example.

### 2.2. The formal definitions

The formal definitions of TAG and related concepts can be found in Joshi and Schabes (1997) and Kallmeyer (2009). These definitions are reproduced here for completeness.

**Definition 1.** A *finite tree* is a directed graph, denoted by $\gamma = \langle V, E, r \rangle$, where $V$ is the set of vertices, $E$ is the set of edges, and $r \in V$ is the root node, such that

- $\gamma$ contains no cycles,
- $r \in V$ has in-degree (number of incoming edges) 0,
- All $v \in V \setminus \{r\}$ have in-degree 1,
- Every $v \in V$ is accessible from $r$,
- A vertex with out-degree (i.e., number of outgoing edges) 0 is a leaf.

**Definition 2.** A *labelling* of a graph $\gamma = \langle V, E \rangle$ over a *signature* $\langle A_1, A_2 \rangle$ is a pair of functions $l : V \rightarrow A_1$ and $g : E \rightarrow A_2$, with $A_1, A_2$ being a set of disjoint alphabets.

For the next definitions, assume $N$ and $T$ to be disjoint sets of non-terminals and terminals, respectively.

**Definition 3.** A *syntactic tree* is an ordered, labelled tree $\langle V, E, r \rangle$ such that the label $l(v) \in N$ for each vertex $v$ with out-degree at least 1 and $l(v) \in (N \cup T \cup \epsilon)$ for each leaf $v$.

**Definition 4.** An *auxiliary tree* is a syntactic tree $\langle V, E, r \rangle$ such that there is a unique leaf $f$, marked as *foot node*, with $l(f) = l(r)$. An auxiliary tree is denoted as $\langle V, E, r, f \rangle$.

**Definition 5.** An *initial tree* is a non-auxiliary syntactic tree.

With the basic concepts defined, we can now define TAG, and the related operations.

**Definition 6.** A *Tree Adjoining Grammar* is a tuple $G = \langle N, T, S, I, A \rangle$, where
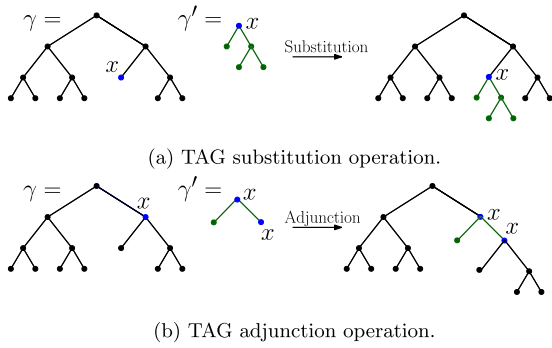
(a) TAG substitution operation.



(b) TAG adjunction operation.

**Fig. 4.** Illustration of the TAG operations (Khandelwal et al., 2019b).

– $N, T$ are disjoint alphabets of non-terminals and terminals,
– $S \in N$ is a start symbol,
– $I$ is a finite set of initial trees and $A$ is a finite set of auxiliary trees.

The set of trees $I \cup A$ is called *elementary trees*.

**Definition 7** (*Substitution*)**.** Let $\gamma = \langle V, E, r \rangle$ be a syntactic tree and $\gamma' = \langle V', E', r' \rangle$ be an initial tree and $v \in V$. The result of substituting $\gamma'$ into $\gamma$ at node $v$, denoted as $\gamma[v, \gamma']$, is defined as follows

– If $v$ is not a leaf or $v$ is a foot node or $l(v) \neq l(r')$, then $\gamma[v, \gamma']$ is not defined,
– otherwise, $\gamma[v, \gamma'] = \langle V'', E'', r \rangle$ with

$$V'' = V \cup V' \setminus \{v\}, \tag{1}$$

and

$$E'' = (E \setminus \{\langle v_1, v_2 \rangle \mid v_2 = v \text{ and } v_1 \in V\}) \ \cup$$
$$E' \ \cup \ \{\langle v_1, r' \rangle \mid v_1, v \in E\}. \tag{2}$$

The substitution operation is illustrated in Fig. 4a.

**Definition 8** (*Adjunction*)**.** Let $\gamma = \langle V, E, r \rangle$ be a syntactic tree and $\gamma' = \langle V', E', r', f \rangle$ be an auxiliary tree and $v \in V$ with out-degree at least 1. The result of adjoining $\gamma'$ into $\gamma$ at node $v$, denoted as $\gamma[v, \gamma']$, is defined as follows

– if $l(v) \neq l(r')$ then $\gamma[v, \gamma']$ is undefined,
– else $\gamma[v, \gamma'] = \langle V'', E'', r'' \rangle$ with

$$V'' = V \cup V' \setminus v, \tag{3}$$

and

$$E'' = (E \setminus \{\langle v_1, v_2 \rangle \mid v_1 = v \text{ or } v_2 = v\}) \ \cup$$
$$E' \ \cup \ \big\{\langle v, r' \rangle \mid \langle v_1, v \rangle \in E\big\} \ \cup$$
$$\{\langle f, v_2 \rangle \mid \langle v, v_2 \rangle \in E\}. \tag{4}$$

The adjunction operation is illustrated in Fig. 4b.

Recall that a tree obtained by performing an arbitrary number of valid substitution and adjunction operations to an initial tree $\gamma = \langle V, E, r \rangle$ with $l(r) = S$ is called a *derived tree* (for example, as in Fig. 1). Also recall that the substitution and adjunction operations performed can be represented in a tree representation called *derivation tree* (for example, as in Fig. 3b). A derived tree is said to be *saturated* if all leaves of the derived tree belong to the set $T$ and cannot be further substituted. The corresponding derivation tree is also said to be *saturated*.

**Definition 9** (*Tree Language and String Language*)**.** Let $G = \langle N, T, S, I, A \rangle$ be a TAG. The *tree language* $L_T(G)$ of grammar $G$ is defined as the set of all saturated derived trees in $G$ with root $S$.

The *string language* $L(G)$ of $G$ is the set of yields of the trees in $L_T(G)$.

## 3. TAG description of dynamical systems

In this Section, we define a notion of *model set* based on TAG and propose a TAG for a generic class of dynamical models — the polynomial NARMAX class.

### 3.1. Model set

Consider the following discrete-time input–output representation of a non-linear dynamical model

$$y_k = f(u_k, \dots, u_{k-n_u}, y_{k-1}, \dots, y_{k-n_y}, \xi_{k-1}, \dots,$$
$$\xi_{k-n_\xi}) + \xi_k \tag{5}$$

where $u_k, y_k \in \mathbb{R}$ are the input and output signals at time-instant $k$, $\xi_k \sim \mathcal{N}(0, \sigma_\xi^2)$ is a noise signal independent of input $u$, constants $n_u, n_y$ and $n_\xi$ are the corresponding maximum time-lags and the non-linear function $f(\cdot)$ belongs to an arbitrary set of functions $\mathcal{M}$. In PEM, the set of functions $\mathcal{M}$, also known as the *model set*, along with a specified choice for $n_u, n_y$ and $n_\xi$, is determined by a user based on expert knowledge, prior information and informative experiments. It will be demonstrated in Section 3.2 that TAG can be used to generate trees that yield non-linear functions $f(\cdot)$ with desirable structural properties and varying choices of arguments (time lags of the involved $u, y$ and $\xi$ signals). This capability of TAG leads to a more generalized notion of model set $\mathcal{M}$. In order to formalize this concept, we introduce a function $\Pi_f(u, y, \xi, k)$ that maps from function $f$ to the right-hand-side expression in (5) (in string form). We can now define a new notion of model set, based on TAG, defined as follows.

**Definition 10.** For a given TAG $G$, the corresponding *model set* $\mathcal{M}(G)$ is defined as the set of models in the form of (5) such that $\Pi_f(u, y, \xi, k) \in L_T(G)$.

Note that this is a more generalized notion of model set as compared to that used in PEM. In PEM, a model set is typically determined by choosing a fixed model structure along with a suitable parameterization (i.e. model complexity). On the other hand, in this work, the choice of initial and auxiliary trees of a TAG automatically determines the model set. The advantage of such a declaration of a model set is that, when no prior information is available, the model set can be chosen to span a number of commonly used model classes without a prior specification of the model complexity. On the other hand, when prior information on the structure or complexity of the model is available, the grammar can be suitably refined to restrict the model set. In the subsequent sections, we propose a TAG for a generic model class, and demonstrate that the resulting model set spans a number of model structures commonly used in PEM.

### 3.2. The polynomial NARMAX model class

The NARMAX model class is a flexible class on non-linear input–output dynamical models, see Leontaritis and Billings (1985). The polynomial NARMAX model class is the set of all NARMAX models where the non-linear relationships are of the polynomial kind. Polynomial NARMAX is a convenient model representation since any continuous function on a closed space can be approximated arbitrary well using polynomial functions (based on Weierstrass' theorem, see Stone (1948)). Furthermore,

Fig. 5. Initial trees $I$ of TAG $G_N$.



Fig. 6. Auxiliary trees $A$ of TAG $G_N$.

the family of polynomial NARMAX models includes, as special cases, other commonly used model classes such as FIR and AR-MAX. It will be shown that these models can be generated by suitably restricting the TAG presented here.

A discrete-time SISO polynomial NARMAX model can be represented as (see Billings (2013))

$$y_k = \theta_0 + \sum_{i_1=1}^{n} \theta_{i_1} x_{i_1,k} + \sum_{i_1=1}^{n} \sum_{i_2=i_1}^{n} \theta_{i_1 i_2} x_{i_1,k} x_{i_2,k} + \cdots$$

$$\sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}}^{n} \theta_{i_1 i_2 \ldots i_l} x_{i_1,k} x_{i_2,k} \ldots x_{i_l,k} + \xi_k, \qquad (6)$$

where $l$ is the order of the polynomial non-linearity, $\theta_{i_1 i_2 \ldots i_m}$ are the model parameters, and $x_k = (x_{1,k} \cdots x_{n_y+n_u+n_\xi,k})^\top$ is a vector consisting of the past input, output and noise values building up the regressors

$$x_{m,k} = \begin{cases} y_{k-m} & 1 \le m \le n_y \\ u_{k-(m-n_y-1)} & n_y + 1 \le m \le n_y + n_u + 1 \\ \xi_{k-(m-n_y-n_u-1)} & n_y + n_u + 2 \le m \le n_y + n_u + \\ & \qquad\qquad\qquad n_\xi + 1. \end{cases} \qquad (7)$$

We will also use the following alternative and equivalent representation for polynomial NARMAX models:

$$y_k = \sum_{i=1}^{p} c_i \prod_{j=0}^{n_u} u_{k-j}^{b_{i,j}} \prod_{l=1}^{n_\xi} \xi_{k-l}^{d_{i,l}} \prod_{m=1}^{n_y} y_{k-m}^{a_{i,m}} + \xi_k, \qquad (8)$$

where $p$ is the number of model terms, $c_i$ are the model parameters, $a_{i,m}, b_{i,j}, d_{i,l} \in \mathbb{N}$ are the exponents for output, input and noise terms.

### 3.3. Proposed TAG representation

In this section we propose a TAG for the polynomial NAR-MAX model class. The proposed TAG captures the structural relationships in (8). In the sequel, the time index will be dropped in the context of the proposed TAG, as $q^{-1}$ will be used to denote a backward time shift. For convenience, introduce the following notation. For a given model in the form of (8), define $J_i := \{j \in \mathbb{N}_{\ge 0} \mid b_{i,j} \ne 0\}$, $L_i := \{l \in \mathbb{N}_{>0} \mid d_{i,l} \ne 0\}$ and $M_i := \{m \in \mathbb{N}_{>0} \mid a_{i,m} \ne 0\}$. For the $i$th model term, the sequence of delays in the input, noise and output factors are denoted by $\left(\bar{j}_n^{(i)}\right)_{n \in J_i}$, $\left(\bar{l}_n^{(i)}\right)_{n \in L_i}$, $\left(\bar{m}_n^{(i)}\right)_{n \in M_i}$ respectively.

**Theorem 1.** *Consider the TAG $G_N = \langle N, T, S, I, A \rangle$ with*

- $N = \{\text{expr0, expr1, expr2, op, par}\}$,
- $T = \{\text{u}, \text{y}, \xi, +, \text{c}, \times, \ q^{-1}\}$,
- $S = \text{expr0}$,
- $I = \{\alpha_1\}$, *where initial tree $\alpha_1$ is depicted in Fig. 5*,
- $A = \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7\}$, *where the auxiliary trees $\beta_i$'s are depicted in Fig. 6*.

*The model set $\mathcal{M}(G_N)$ is equivalent to the set of all models that can be expressed as (8) with finite values of $p$, $n_u$, $n_y$ and $n_\xi$.*
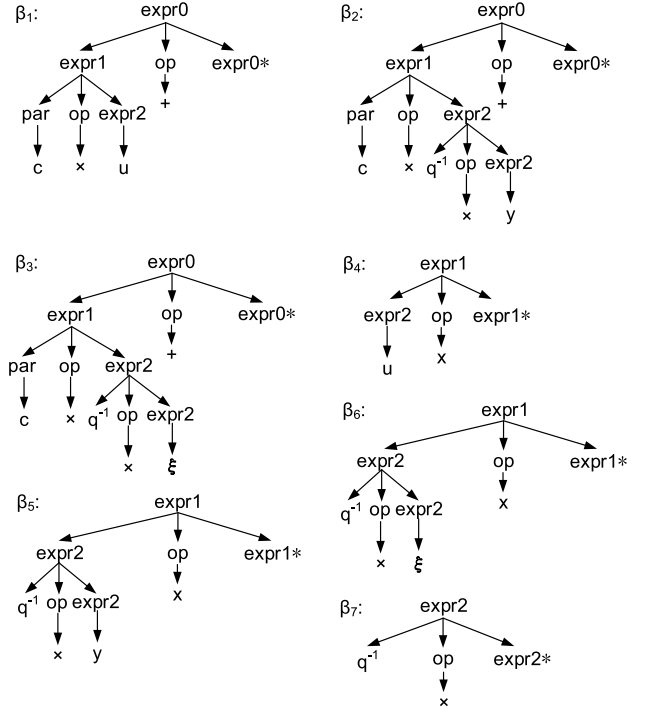
**Proof.** For the first part of the proof, we show that for any polynomial NARMAX model in the form of (8), there exists a derivation tree such that the resulting derived tree has a yield that is equal to the RHS of (8). Algorithm 1 constructs such a derivation tree for a given polynomial NARMAX model. The procedure DELAYS($\gamma$, $v$, $n$) adjoins $n$ auxiliary tree $\beta_7$ to the derivation tree $\gamma$ at vertex $v$. The algorithm constructs the derivation tree by introducing the first factor ($u, y$ or $\xi$) of each of the $p$ model terms, and subsequently building each of the branches by introducing the remaining factors with the corresponding delays and exponents.

For the second part of the proof, it needs to be shown that all expressions in $L(G_N)$, i.e., yields of all possible trees generated by $G_N$, are RHS expressions of polynomial NARMAX models. This is proven by structural induction. We first observe that the simplest tree in $L(G_N)$ is the initial tree $\alpha_1$ with the yield $\xi$. This corresponds to the model

$$y_k = \xi_k, \qquad (9)$$

which belongs to the polynomial NARMAX class. Now, consider an arbitrary saturated derived tree $\gamma \in L_T(G_N)$ whose yield is the RHS of a polynomial NARMAX model. This implies that the yield is a polynomial expression in terms of the factors $u, y$ and $\xi$. To complete the principle of induction, it must be shown that any possible adjunction to $\gamma$ results in a new tree in $L_T(G_N)$ whose yield is also a polynomial expression in terms of the aforementioned factors.

For convenience, the auxiliary trees are grouped based on the operators involved - $\beta_1, \beta_2, \beta_3$ are called *additive-type* auxiliary trees, $\beta_4, \beta_5, \beta_6$ are called *multiplicative-type*, and $\beta_7$ is called *delay-type* auxiliary tree. The following adjunctions be made on $\gamma$:

- adjunction of an additive-type tree. Such an adjunction introduces an input, output or noise term additively in the expression while respecting the causality of the expression. Hence the resulting expression is also a polynomial;

**Algorithm 1** Parse NARMAX model (8) to derivation tree.

**Require:** $p, J_i, L_i, M_i, \left(\bar{j}_n^{(i)}\right)_{n \in J_i}, \left(\bar{l}_n^{(i)}\right)_{n \in L_i}, \left(\bar{m}_n^{(i)}\right)_{n \in M_i}$

1: $V \leftarrow \{v_0\}; \quad l(v_0) \leftarrow \alpha_1$       ▷ initialize with start tree
2: $r \leftarrow v_0$
3: $V \leftarrow \bigcup_{i=1}^{p} \{v_{i,1}\} \cup V$ ▷ Insert $p$ vertices to begin the $p$ summation branches
4: $E \leftarrow \bigcup_{i=2}^{p} \{\langle v_{i-1,1}, v_{i,1}\rangle\} \cup \{\langle v_0, v_{1,1}\rangle\}$
5: **for** $i \leftarrow 1, p$ **do**
6:     **if** $J_i \neq \phi$ **then**     ▷ If there is an input factor in the $i^{\text{th}}$ term
7:       $l(v_{i,1}) \leftarrow \beta_1$     ▷ For each summation branch, assign the appropriate label to the first vertex
8:       $\langle V, E, r\rangle \leftarrow \text{DELAYS}(\langle V, E, r\rangle, v_{i,1}, \bar{j}_1^{(i)})$   ▷ Adjoin delay trees
9:       $b_{i,\bar{j}_1^{(i)}} \leftarrow b_{i,\bar{j}_1^{(i)}} - 1$ ▷ Reduce the corresponding exponent by 1
10:     **else if** $L_i \neq \phi$ **then**
11:       $l(v_{i,1}) \leftarrow \beta_3$
12:       $\langle V, E, r\rangle \leftarrow \text{DELAYS}(\langle V, E, r\rangle, v_{i,1}, \bar{l}_1^{(i)})$
13:       $d_{i,\bar{l}_1^{(i)}} \leftarrow d_{i,\bar{l}_1^{(i)}} - 1$
14:     **else if** $M_i \neq \phi$ **then**
15:       $l(v_{i,1}) \leftarrow \beta_2$
16:       $\langle V, E, r\rangle \leftarrow \text{DELAYS}(\langle V, E, r\rangle, v_{i,1}, \bar{m}_1^{(i)})$
17:       $a_{i,\bar{m}_1^{(i)}} \leftarrow a_{i,\bar{m}_1^{(i)}} - 1$
18:     $s_i \leftarrow 1$     ▷ Counter for multiplying remaining factors
19:     **for all** $j \in J_i$ **do**
20:       $V \leftarrow \bigcup_{n=1}^{b_{i,j}} \{v_{i,s_i+n,1} \cup V\}; \quad l(v_{i,s_i+n,1}) \leftarrow \beta_4$
21:       $E \leftarrow \bigcup_{n=1}^{b_{i,j}} \{\langle v_{i,s_i+n-1,1}, v_{i,s_i+n,1}\rangle\} \cup E$
22:       **for** $n \leftarrow 1, \bar{b}_j^{(i)}$ **do**     ▷ Adjoin delays for multiple factors
23:         $\langle V, E, r\rangle \leftarrow \text{DELAYS}(\langle V, E, r\rangle, v_{i,s_i+n,1}, j)$
24:       $s_i \leftarrow s_i + \bar{b}_j^{(i)}$
25:     **for all** $l \in L_i$ **do**
26:       $V \leftarrow \bigcup_{n=1}^{d_{i,l}} \{v_{i,s_i+n,1} \cup V\}; \quad l(v_{i,s_i+n,1}) \leftarrow \beta_6$
27:       $E \leftarrow \bigcup_{n=1}^{d_{i,l}} \{\langle v_{i,s_i+n-1,1}, v_{i,s_i+n,1}\rangle\} \cup E$
28:       **for** $n \leftarrow 1, \bar{d}_l^{(i)}$ **do**
29:         $\langle V, E, r\rangle \leftarrow \text{DELAYS}(\langle V, E, r\rangle, v_{i,s_i+n,1}, l)$
30:       $s_i \leftarrow s_i + \bar{d}_l^{(i)}$
31:     **for all** $m \in M_i$ **do**
32:       $V \leftarrow \bigcup_{n=1}^{a_{i,m}} \{v_{i,s_i+n,1} \cup V\}; \quad l(v_{i,s_i+n,1}) \leftarrow \beta_5$
33:       $E \leftarrow \bigcup_{n=1}^{a_{i,m}} \{\langle v_{i,s_i+n-1,1}, v_{i,s_i+n,1}\rangle\} \cup E$
34:       **for** $n \leftarrow 1, \bar{a}_m^{(i)}$ **do**
35:         $\langle V, E, r\rangle \leftarrow \text{DELAYS}(\langle V, E, r\rangle, v_{i,s_i+n,1}, m)$
36:       $s_i \leftarrow s_i + \bar{a}_m^{(i)}$
      **return** $\langle V, E, r\rangle$

- adjunction of a multiplicative-type tree. This simply introduces multiplicative factors to an existing model term, and hence, the resulting expression is also a polynomial;
- adjunction of a delay-type tree. This operation simply adds delays to an existing monomial, and hence preserves the polynomial structure of the expression.

Since all possible operations yield a causal polynomial expression, it can be concluded that $L(G_N)$ consists of only dynamical polynomial expressions in terms of the factors $u, y$ and $\xi$ which corresponds to a polynomial NARMAX model. This concludes the proof. □

Theorem 1 demonstrates that structural properties of a rich class of dynamical models can be captured within a compact set of trees of a TAG. The expansive representational capability of TAG can be exploited using EAs such as GP to identify models without prior specification of structure and complexity, as demonstrated in Khandelwal et al. (2019b). Furthermore, Algorithm 1 provides a method to compute the derivation tree representation of a given polynomial NARMAX model in terms of

grammar $G_N$. Consequently, available prior information about the model of the system can be translated to TAG representation (or incorporated in tree sets $I, A$), thereby making the evolutionary search more efficient. Hence, the use of TAG enables identification within a larger class of dynamical models without requiring user-interaction, while simultaneously allowing the user to restrict the evolutionary search effectively.

## 4. Illustrations

In this section we discuss aspects of TAG useful for EA-based SI. We demonstrate the use of TAG $G_N$ to generate polynomial NARMAX models. It is also shown that models belonging to simpler model classes can be generated by scaling down the set of elementary trees of $G_N$ appropriately. Furthermore, more flexible model classes can be represented by scaling up the set of elementary trees. This is demonstrated by extending the proposed TAG to generate Non-linear Box–Jenkins (NBJ) models.

### 4.1. Model generation using $G_N$

Three illustrative examples are used to demonstrate the generation of models using $G_N$. The models generated belong to the ARX, polynomial NARX and polynomial NARMAX model classes. It will be demonstrated that by restricting the elementary trees $I$ and $A$ to subsets of the elementary trees in the proposed TAG $G_N$, we can generate models that only belong to model sub-classes that are properly included in the set of polynomial NARMAX models, such as FIR and truncated Volterra series.

#### 4.1.1. ARX example
ARX models can be described by the equation

$$y_k = \sum_{i=0}^{n_u} b_i u_{k-i} + \sum_{j=1}^{n_y} a_j y_{k-j} + \xi_k, \tag{10}$$

where $a_j, b_i \in \mathbb{R}$ are coefficients. The grammar $G_N$ can be used to generate ARX models by restricting the auxiliary tree set $A$ as

$$A' = \{\beta_1, \beta_2, \beta_7\} \subset A. \tag{11}$$

Consider the example depicted in Fig. 7a. Tree (A) is a derivation tree with initial tree $\alpha_1$ at the root node, and auxiliary trees $\beta_1$ and $\beta_2$ in subsequent vertices. The edges are labelled with Gorn addresses of vertices in the auxiliary trees at which adjunctions take place. Performing the adjunctions results in derived tree (B) in Fig. 7a. The RHS of the resulting model appears at the leaves of the derived tree, and the corresponding model is

$$y_k = c_1 y_{k-1} + c_2 u_k + \xi_k. \tag{12}$$

#### 4.1.2. NARX example
Polynomial NARX models can be described by the equation

$$y_k = \sum_{i=1}^{p} c_i \prod_{j=0}^{n_u} u_{k-j}^{b_{i,j}} \prod_{m=1}^{n_y} y_{k-m}^{a_{i,m}} + \xi_k. \tag{13}$$

By restricting auxiliary trees to the set

$$A'' = \{\beta_1, \beta_2, \beta_4, \beta_5, \beta_7\} \subset A \tag{14}$$

we can restrict the proposed grammar to generate polynomial NARX models only. Consider the example derivation tree (A) in Fig. 7b, which is an extension of the previous example. The derivation tree consists of the initial tree $\alpha_1$, and auxiliary trees $\beta_2$, $\beta_3$ and $\beta_4$. Performing the adjunctions described by the derivation tree results in the derived tree (B) in Fig. 7b. The corresponding symbolic model is

$$y_k = c_1 y_{k-1}^2 + c_2 u_k + \xi_k. \tag{15}$$
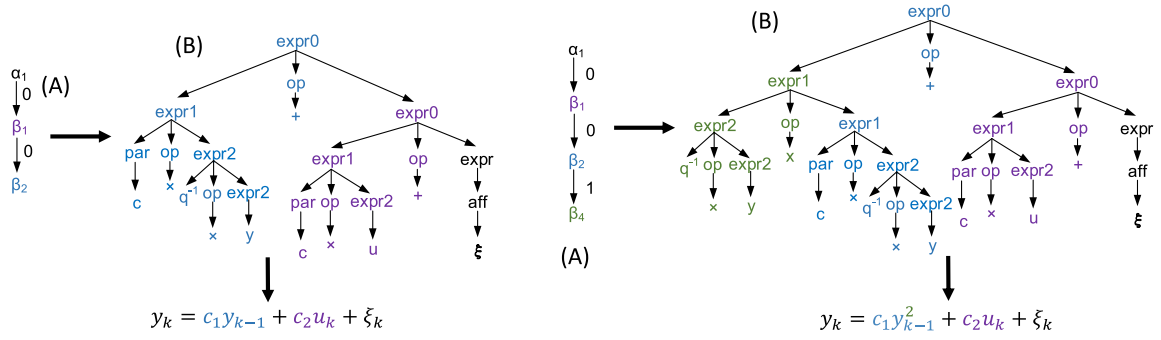
(a) Example 1 - derivation tree (A), derived tree (B) and symbolic model.



(b) Example 2 - derivation tree (A), derived tree (B) and symbolic model.
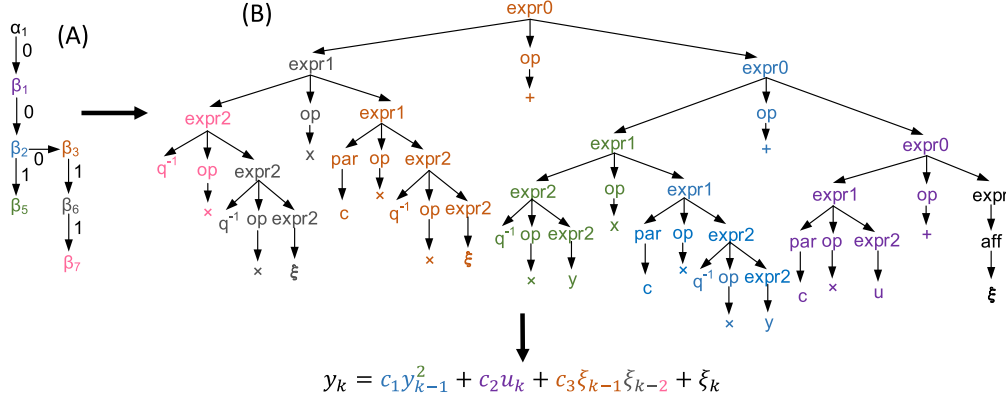


(c) Example 3 - derivation tree (A), derived tree (B) and symbolic model.

**Fig. 7.** Illustrative examples.

#### 4.1.3. NARMAX example

This example builds on the previous example by using the complete auxiliary tree set $A$ and adjoining trees $\beta_3$, $\beta_6$ and $\beta_7$ to the tree $\beta_2$. The new derivation tree and derived tree are depicted in Fig. 7c. The corresponding model,

$$y_k = c_1 y_{k-1}^2 + c_2 u_k + c_3 \xi_{k-1} \xi_{k-2} \xi_k + \xi_k, \tag{16}$$

is a polynomial NARMAX model.

#### 4.2. Non-linear Box–Jenkins extension

Just like the proposed grammar can be scaled down to generate specific dynamic sub-classes, it can also be extended to generate models that belong to a more generalized class of models. We illustrate this by extending the proposed grammar to a more generalized models structure − Non-linear Box–Jenkins (NBJ).

In the case of linear systems, a Box–Jenkins model structure is an extension of the Output Error (OE) model structure, where the error is modelled as an ARMA process (Ljung, 1999). The BJ class also includes, as special cases, other linear model structures such as ARMAX and OE. In the same spirit, NBJ model structure can be expressed as a Non-linear Output Error (NOE) model where the error is subsequently modelled as a NARMA process. The NBJ model structure is given by the following equations

$$\hat{y}_k = f(\hat{y}_{k-1}, \dots, \hat{y}_{k-n_y}, u_k, \dots, u_{k-n_u}),$$

$$v_k = g(v_{k-1}, \dots, v_{k-n_v}, u_k, \dots, u_{k-n_u}, \xi_{k-1}, \dots, \xi_{k-n_\xi}) + \xi_k,$$

$$y_k = \hat{y}_k + v_k, \tag{17}$$

where $f(\cdot)$ and $g(\cdot)$ are polynomial functions in terms of their arguments. Notice that the RHS expressions of the equations describing the process and noise dynamics have the same structure that was studied in Section 3.2 for NARMAX models (see (8)). Hence, the proposed TAG can be extended to generate NBJ models. Fig. 8 depicts the initial and auxiliary trees of the grammar for NBJ model structures $G_{NBJ}$. The structure of the initial tree $\alpha_1$ ensures that all elements in $L(G_{NBJ})$ contain two expressions, separated by a comma, that represent the functions $f(\cdot)$ and $g(\cdot)$ respectively. Each of these expressions can be expanded by adjoining auxiliary trees that ensure that the polynomial structure is maintained.

### 5. Conclusions

We presented a TAG based concept of a model set, that is more general than that commonly used in the system identification literature. A TAG $G_N$ was proposed that captures the dynamical structure of polynomial NARMAX models. It was demonstrated that sub-classes of the polynomial NARMAX class can be represented by choosing an appropriate subset of the elementary trees of $G_N$. Similarly, more flexible model classes like Non-linear Box–Jenkins can be represented by extending the set of elementary trees. This illustrates that a compact set of elementary trees can be used to express the dynamical relationships across a variety of model classes, thereby enabling the design of TAG-based EA approaches for SI that require minimal user-interaction. The practical soundness of this concept has been demonstrated in Khandelwal et al. (2019b), where a TAG-based EA approach was used to identify a non-linear benchmark dataset with minimal user-interaction, and also in Khandelwal et al. (2019a), where the same TAG-based EA approach is used to identify multiple real physical systems and benchmark data set with minimal changes in the methodology itself.
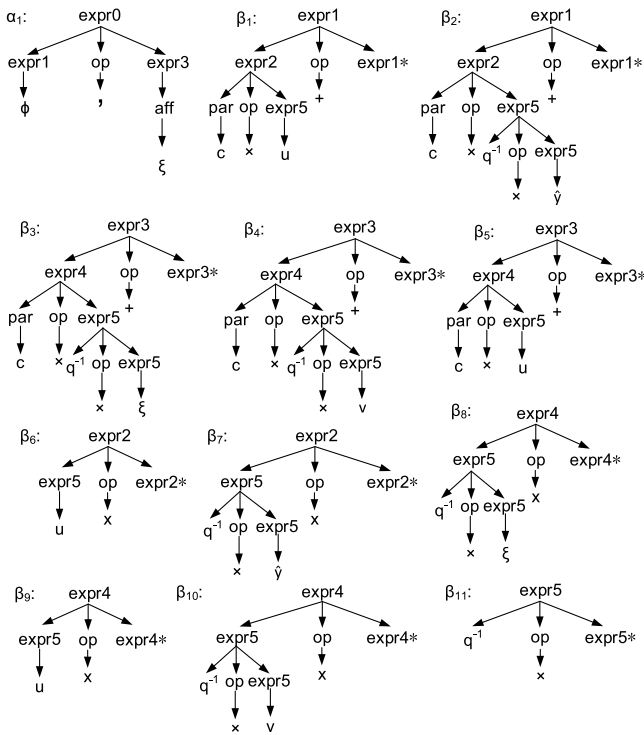
**Fig. 8.** Initial and auxiliary trees (*I* and *A*) of TAG $G_{\text{NBJ}}$.

## References

Arias-Montano, A., Coello, C. A. C., & Mezura-Montes, E. (2012). Multiobjective evolutionary algorithms in aeronautical and aerospace engineering. *IEEE Transactions on Evolutionary Computation*, *16*(5), 662–694.

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

Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing, Vol. 53*. Springer.

Fonseca, C. M., & Fleming, P. J. (1996). Non-linear system identification with multiobjective genetic algorithms. In *Proc. of 13th IFAC World Congress* (pp. 1169–1174). San Francisco, USA.

Gorn, S. (1965). Explicit definitions and linguistic dominoes. In *Conference on systems and computer science* (pp. 77–115). Univ. of Western Ontario.

Gray, G. J., Murray-Smith, D. J., Li, Y., Sharman, K. C., & Weinbrenner, T. (1998). Nonlinear model structure identification using genetic programming. *Control Engineering Practice*, *6*(11), 1341–1352.

Joshi, A. K., & Schabes, Y. (1997). Tree-adjoining grammars. In *Handbook of formal languages* (pp. 69–123). Springer.

Kallmeyer, L. (2009). A declarative characterization of different types of multi-component tree adjoining grammars. *Research on Language and Computation*, *7*(1), 55–99.

Khandelwal, D., Schoukens, M., & Tóth, R. (2019a). Data-driven modelling of dynamical systems using tree adjoining grammar and genetic programming. In *Proc. of the IEEE congress on evolutionary computation* (pp. 2673–2680). Wellington, New Zealand.

Khandelwal, D., Schoukens, M., & Tóth, R. (2019b). Grammar-based representation and identification of dynamical systems. In *Proc. of the 18th European Control Conference (ECC)* (pp. 1318–1323). Naples, Italy.

Kristinsson, K., & Dumont, G. A. (1992). System identification and control using genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(5), 1033–1046.

Laurain, V., Tóth, R., Piga, D., & Darwish, M. A. H. (2020). Sparse rkhs estimation via globally convex optimization and its application in lpv-io identification. *Automatica*, *115*, 108914.

Leontaritis, I., & Billings, S. A. (1985). Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International Journal of Control*, *41*(2), 303–328.

Ljung, L. (Ed.), (1999). *System identification (2nd Ed.): Theory for the user*. Prentice Hall PTR.

Madár, J., Abonyi, J., & Szeifert, F. (2005). Genetic programming for the identification of nonlinear input- output models. *Industrial & Engineering Chemistry Research*, *44*(9), 3178–3186.

Pillonetto, G., Chiuso, A., & De Nicolao, G. (2011). Prediction error identification of linear systems: a nonparametric gaussian regression approach. *Automatica*, *47*(2), 291–305.

Quade, M., Abel, M., Shafi, K., Niven, R. K., & Noack, B. R. (2016). Prediction of dynamical systems by symbolic regression. *Physical Review E*, *94*(1), 012214.

Rodríguez-Vázquez, K., & Fleming, P. J. (2000). Use of genetic programming in the identification of rational model structures. In *Proc. of the European conference on genetic programming* (pp. 181–192). Berlin, Heidelberg.

Rodriguez-Vazquez, K., Fonseca, C. M., & Fleming, P. J. (2004). Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *34*(4), 531–545.

Rojas, C. R., Tóth, R., & Hjalmarsson, H. (2014). Sparse estimation of polynomial and rational dynamical models.. *IEEE Transactions on Automatic Control*, *59*(11), 2962–2977.

Stone, M. H. (1948). The generalized weierstrass approximation theorem. *Mathematics Magazine*, *21*(5), 237–254.

**Dhruv Khandelwal** was born in 1991 in Kolkata, India. He received his Bachelors degree in Electrical and Electronic Engineering from VIT University in Vellore, India in 2013. He received his MSc. degree and PhD. degree from the Department of Electrical Engineering at Eindhoven University of Technology in 2015 and 2020, respectively. His research interests are in system identification and in the use of machine learning techniques in modelling of control.

**Maarten Schoukens** is an Assistant Professor in the Control Systems group of the Department of Electrical Engineering at the Eindhoven University of Technology. He received the master's degree in electrical engineering and the Ph.D. degree in engineering from the Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 2010 and 2015 respectively. From 2015 to 2017, he has been a Post-Doctoral Researcher with the ELEC Department, VUB. In October 2017 he joined the Control Systems research group, TU/e, Eindhoven, The Netherlands as a Post-Doctoral Researcher, in 2018 he became an Assistant Professor in the same group. Maarten was awarded an FWO Ph.D. Fellowship in 2011, and a Marie Skodowska-Curie Individual Fellowship in 2018. His main research interests include the measurement and data-driven modelling and control of linear parameter-varying and nonlinear dynamical systems using system identification and machine learning techniques.

**Roland Tóth** was born in 1979 in Miskolc, Hungary. He received the B.Sc. degree in Electrical Engineering and the M.Sc. degree in Information Technology in parallel with distinction at the University of Pannonia, Veszprém, Hungary, in 2004, and the Ph.D. degree (cum laude) from the Delft Center for Systems and Control (DCSC), Delft University of Technology (TUDelft), Delft, The Netherlands, in 2008. He was a Post-Doctoral Research Fellow at DCSC, TUDelft, in 2009 and at the Berkeley Center for Control and Identification, University of California, Berkeley, in 2010. He held a position at DCSC, TUDelft, in 2011–2012. Currently, he is an associate Professor at the Control Systems Group, Eindhoven University of Technology (TU/e). He is an associate Editor of the IEEE Transactions on Control Systems Technology.

His research interests are in linear parameter-varying (LPV) and nonlinear system identification and control, machine learning for modelling and control, model predictive control and behavioural system theory.

Dr. Tóth received the TUDelft Young Researcher Fellowship Award in 2010, the VENI award of The Netherlands Organisation for Scientific Research in 2011 and the Starting Grant of the European Research Council in 2016.