DOI: 10.48011/asba.v2i1.1384

Multi-gene Genetic Programming for Structure Selection of Polynomial NARMAX models

Henrique Carvalho de Castro *
Bruno Henrique Groenner Barbosa **

* *Programa de Pós-graduação em Engenharia de Sistemas e Automação, Universidade Federal de Lavras, MG, (e-mail: henriquec.castro@outlook.com).*
** *Departamento de Automática, Universidade Federal de Lavras, MG, (e-mail: brunohb@ufla.br)*

Abstract: In the area of black-box identification, NARMAX models are of great interest. The main difficulty faced when working with such models is the selection of the correct structure to represent the underlying system in the data. Orthogonal Least Squares (OLS) methods are widely used for this task, however, there are systems with a high degree of non-linearity and long term dependencies, which makes the use of traditional OLS methods computationally impracticable. In this sense, this paper studies the use of Multi-Gene Genetic Programing (MGGP) together with the traditional OLS method to increase the search space and turn the structure selection practicable for average performance computer. It is shown that, in real-life problem data, the algorithm can find better models than previous works' models. The MGGP found a model for a hydraulic pumping system with a better one-step-ahead prediction error (0.058 mlc^2 against 0.070 mlc^2) using PEM technique and better free-run simulation error (0.997 mlc^2 against 1.120 mlc^2) using SEM technique. The MGGP found a model with such a degree of non-linearity and maximum input-output lags that totalizes 142505 candidate terms for traditional OLS analysis, which is impracticable for average performance computers.

Keywords: Nonlinear system identification, Evolutionary algorithms, Polynomial NARMAX models

1. INTRODUCTION

Building dynamical models directly from input and output data is a process known as *System Identification*. The system to be modeled can be a manufacturing process, an energy production system, stock market, or climatic phenomena, among others. A model allows the understanding of the system characteristics and the prediction of its behavior (Garg et al., 2017). In order to solve an identification problem, one must follow the steps: 1 - *dynamic tests*; 2 - *choice of mathematical representation*; 3 - *model structure determination*; 4 - *parameter estimation*; and 5 - *model validation* (Aguirre, 2015).

Generally, most of the real systems of interest are nonlinear (Pope and Rayner, 1994). In this sense, NARMAX models (*Nonlinear AutoRegressive Moving Average with eXogenous variables*)(Leontaritis and Billings, 1985) are of great interest in the area, due to its flexibility and representation capacity. Even with its advantages, working with NARMAX models has its difficulties. The main problem encountered in doing so is to select the appropriate model structure, that is, detecting the regressors that together best represent the system. The key point of structure selection is to choose a model structure as simple as possible, but sufficiently complex to capture the dynamics underlying the data (Aguirre and Letellier, 2009).

A widely used criterion for NARMAX model structure selection is the *Error Reduction Ratio* (ERR) (Billings et al., 1989), which evaluates how good each single model term is to explain the output data variance. It is considered a one-step-ahead *prediction error minimization* (PEM) technique. Some algorithms were built based on ERR criterion for structure selection, as for example the *Forward Regression Orthogonal Estimator* (FROE) (Billings et al., 1989) and others *Orthogonal Least Squares* (OLS) based methods (Chen et al., 1989). Piroddi and Spinelli (2003) discuss the limitations of ERR based algorithms, mainly for training data with the presence of certain input characteristic, and the use of *simulation error minimization* (SEM) techniques. Another issue is that such techniques suffer from the *curse of dimensionality* with the increment of the degree of non-linearity and higher long term dependencies.

Alternative methods to solve the structure selection problem can be derived from *Evolutionary Algorithms*, as *Genetic Algorithms* (GA) (Holland, 1975; Goldberg and Holland, 1988) and *Genetic Programming* (GP) (Koza, 1992). Some examples of such methods can be seen in Li and Jeon (1993), Madar et al. (2005), Chen et al. (2007) and

⋆ This work was financially supported by CAPES foundation.

Castro and Barbosa (2020). However, these methods also depend on the assembling of a full regressors matrix for given maximum delays and non-linearity degree, which may become computationally impracticable.

One very flexible algorithm to be used in System Identification is the *Multi-Gene Genetic Programming* (MGGP) (Hinchliffe et al., 1996; Hinchliffe, 2001; Hinchliffe and Willis, 2003), that can be seen as a GA in which each gene is composed by a GP. Hinchliffe et al. also introduce an interesting feature in their algorithm that is a delay function operator (or back-shift operator) capable of increment the lag of a variable. This feature allows the evolution of programmings without the need for assembling a regressors matrix with all possible terms, which avoids the *curse of dimensionality*.

Recently, several modeling and forecasting works have been developed with the use of MGGP (as in Ghareeb and El Saadany (2013), Mehr and Kahya (2017), Safari and Mehr (2018) and Riahi-Madvar et al. (2019)). The algorithm has shown itself to be very flexible and to present good performance. In it, the user is able to determine the set of functions to be used in the modeling (as multiplication, division, exponential function, etc) and change it very easily, without the need for modifications in the algorithm itself. However, none could be found applying MGGP in a NARX/NARMAX modeling context, apart from the original work of Hinchliffe et al. (hence, the *back-shift operator* has not been worked with).

Therefore, this paper proposes the use of MGGP together with OLS/ERR for the structure selection problem of *polynomial* NARMAX models. It is used the back-shift operator for variable lag determination and the OLS/ERR structure selection is applied before the evaluation step. Both algorithms (MGGP and OLS/ERR) work in a *symbiotic* way, that can be interpreted as: *i)* the MGGP algorithm selects groups of terms from the candidate regressors space. Over those groups is applied the OLS/ERR structure selection algorithm and the resultant models are assessed by a cost function. This interaction is supposed to facilitate the OLS/ERR task in a "divide and conquer" manner when the search space is very large, which avoids the curse of dimensionality; and *ii)* the OLS/ERR algorithm works as a pruning method over the MGGP individuals. It leaves only relevant terms in all models. So, this interaction is supposed to guide the population evolution towards a promising region in the search space. The results show that the hybridization MGGP/ERR explores a very wide search space in which traditional OLS algorithms would require high computational power and memory.

The remainder of this paper is organized as follows:
Section 2 presents a theoretical foundation for a basic understanding of the features included into the proposed algorithm; Section 3 presents the methodology applied on this research; Section 4 discuss the results and Section 5 the conclusions.

2. THEORETICAL FOUNDATION

2.1 Polynomial NARMAX models

The NARMAX models can be understood as a combination of delayed input and output signals. In other words, the current value of the model output is a combination of past input and output values. They can be represented by the following equation:

$$\begin{aligned} y(k) =& f^l[y(k-1), ..., y(k-n_y), u(k-1), ..., \\ & u(k-n_u), e(k-1)...e(k-n_e)] + e(k), \end{aligned} \tag{1}$$

where f is a non-linear function, $y(k)$, $u(k)$ and $e(k)$ are the output, input and noise, and n_y, n_u and n_e their corresponding maximum delays. In the case of *polynomial* models, the non-linear function is a polynomial function of degree l (f^l). NARX models are characterized by the absence of noise terms. For one-step-ahead predictions, the terms $y(k-n)$ are from system output and the residuals in this case are called one-step-ahead *prediction error* (PE). As for free-run simulations, the model is defined as:

$$\begin{aligned} y(k) =& f^l[\hat{y}(k-1), ..., \hat{y}(k-n_y), u(k-1), ..., \\ & u(k-n_u), e(k-1)...e(k-n_e)] + e(k), \end{aligned} \tag{2}$$

where $\hat{y}(k-n)$ is the model output and the residuals are called free-run *simulation error* (SE).

It is important to notice that the amount of terms the model contains increases exponentially with the raising of non-linearity degrees and the input and output maximum lags. Moreover, an over parametrized model can lead to numerical instability in parameter estimation, unnecessary computational cost and the representation of dynamics that do not exist in the real system (Aguirre and Letellier, 2009). The aforementioned issues justify the importance of the structure selection step in system identification.

2.2 Forward Regression Orthogonal Estimator

In FROE algorithms, the model structure is incremented iteratively until it is achieved a certain precision of one-step-ahead prediction. The model parameters are estimated via *Orthogonal Least Squares* (OLS). These orthogonalization techniques are made in such a way that, at each step, the relevance of each regressor candidate can be assessed separately via ERR:

$$[ERR]_j = \frac{\hat{g}_j^2 \sum_{i=1}^{N} w_j^2(i)}{\sum_{i=1}^{N} y^2(i)} \tag{3}$$

where w_j is the j-th auxiliary orthogonal regressor and $\hat{g}_j$ its corresponding estimated parameter. The regressors with the highest ERR are included in the model (see more in Chen et al. (1989)). Several similar techniques are proposed in the literature to accomplish structure selection and parameter estimation at the same time and they will be referred to as OLS/ERR in this paper.

2.3 Evolutionary Algorithms

The *Evolutionary Algorithms* (EA) are based on metaphors of natural processes to build computational models capable of solving problems. There is a wide variety of proposed algorithms that simulate the evolution of species through the natural phenomena of selection, mutation and

reproduction that occur in a given population of individuals. Generally, EA have a standard behavior: it begins with a initial population of random individuals (chromosomes) and at each generation (main loop) the best solutions are sought (*selection*), then combined (*recombination/reproduction/crossover* and *mutation*) in order to generate even better individuals.

There are two points of extreme significance in EA: the representation of the individuals and the evaluation function (or cost function). In terms of representation, several algorithms are proposed. This paper requires the understanding of three of them:

- The *Genetic Algorithms* (GA) were introduced and enhanced by Holland (1975); Goldberg and Holland (1988). In its simplest form, GA has a binary representation, that is, an individual is codified by a binary vector in which each bit or a group of bits represents one variable of the problem. For example, an individual I represented by a 5 bit vector is codified as $I = [\,1\ 0\ 0\ 1\ 0\,]$. In a NARMAX structure selection context, each bit can be interpreted as a Boolean variable which indicates whether the corresponding candidate term will be included in the model or not.

- The *Genetic Programming* (GP) (Koza, 1992; Eiben et al., 2003; Poli et al., 2008) is an evolutionary algorithm in which the programmer does not need to know or specify the structure of the solution in advance. The individuals of a population are computer programs that randomly evolve into new programs. The most common representation in GP is the *tree* one. In this representation, from a root node, the tree is divided into several branches in which internal nodes have arithmetic functions (+, -, *, /, max, ...) and terminals, also called leaves, have variables and constants. As a result, the tree representation hierarchically synthesizes a mathematical function.

- The *Multi-Gene Genetic Programming* (MGGP) was introduced by Hinchliffe et al. (1996). It is based on methods already established in Systems Identification in which the models are constructed by combining a number of functions to generate the model output. MGGP can be represented as the combination of separate basis functions:

$$g(\varphi, \Theta) = \sum_{i=1}^{m} \theta_i g_i(\varphi), \tag{4}$$

where m is the number of basis functions, g_i represents individual functions and θ_i the model parameters. It can be seen as a GA in which each *gene* contains one GP as the basis function (see more in Hinchliffe (2001) and Hinchliffe and Willis (2003)).

2.4 Extended Least Squares

As NARX models are linear-in-parameters, the parameter estimation can be made by the *Least Squares* estimator (LS) as follows:

$$\hat{\theta}_{LS} = [\Psi^T \Psi]^{-1} \Psi^T y \tag{5}$$

where Ψ is the regressors matrix, y is the output data vector and $\hat{\theta}_{LS}$ the parameters estimated via LS.

For problems with output error, LS gives a biased parameter estimation. In these cases, it is necessary to identify a noise model. The NARX model becomes a NARMAX model, which parameters can not be estimated via the traditional LS estimator. A method that can be used to estimate these parameters is the *Extended Least Squares* (ELS) (Young, 1968; Aguirre, 2015). For this case, consider that the prediction residuals ($\xi = y - \Psi\hat{\theta}_{LS}$) can be modeled as:

$$\xi(k) = c_i \upsilon(k-i) + \upsilon(k), \tag{6}$$

where $\upsilon(k)$ is white noise and c_i the parameter of the corresponding noise model term. The $\upsilon(k-1)$ term is included into the regressors matrix:

$$\Psi^* = \begin{bmatrix} & \upsilon(k-1) \\ & \upsilon(k) \\ \Psi & \upsilon(k+1) \\ & \dots \\ & \upsilon(k+N-2) \end{bmatrix}, \tag{7}$$

which is called *extended matrix*. A new vector of parameters is defined as $\theta^* = [\,\theta\ c_i\,]$ and its values are to be estimated via LS. As $\upsilon(k)$ is unknown, the process must be iterative and the residuals vector calculated at each iteration.

2.5 The NARMAX model for a White Noise Output Error Problem

According to Aguirre (2015), a *white noise* is a signal characterized by a function of auto-correlation that satisfies $r_{\xi\xi}(x) = 0$, $\forall k \neq 0$, with power spectrum that contains energy in all frequencies, which are all equally important. On the other hand, in a *colored noise* not all frequencies are equally important. This kind of noise can be modeled by an AR process excited by a white noise signal. The power spectrum of a *colored noise* does not have energy in all frequencies, its power density is concentrated in a relatively narrow range of frequencies.

Let an example system S_{ex} be:

$$S_{ex} : \begin{cases} w(k) = \theta_1 w(k-2) + \theta_2 u(k-1) + v(k) \\ y(k) = w(k) + e(k) \end{cases} \tag{8}$$

where $v(k)$ represents *equation error* (EE), and $e(k)$ represents *output error* (OE).

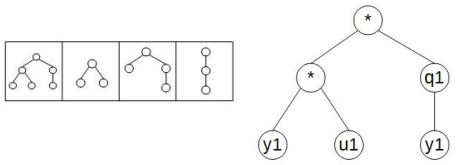
A white noise output error problem considers $v(k) = 0$ and $e(k)$ a *White Gaussian Noise* (WGN) with zero mean. The white noise OE becomes a colored EE noise, so that, the residuals from the model will have some correlation with previous values. This can be mathematically understood by isolating $w(k)$ from the second equation of the system (8) and replacing it on the first equation.

The NARMAX model to represent system S_{ex} is:

$$y(k) = \theta_1 y(k-2) + \theta_2 u(k-1) - \theta_1 e(k-2) + e(k). \tag{9}$$

3. MATERIALS AND METHODS

This paper proposes a hybrid MGGP/ERR algorithm to solve the structure selection problem of *polynomial* NARMAX models. The set of primitive functions (corresponding to the *nodes* of a GP individual) is restricted to the multiplication operator and the delay function operator (or back-shift operator), so that, each *gene* represents a single polynomial term which degree of non-linearity is



(a) MGGP (b) GP

Figure 1. The MGGP representation (a) is a *chromosome* in which each *gene* contains one GP individual (b). The GP individual contains *nodes* with multiplication operators ($*$) and the delay function operator ($q1$) and represents the mathematical function $y(k-1) \cdot u(k-1) \cdot y(k-2)$.

determined by the multiplication of variables (*leaves* or *terminals*). The back-shift operator is a technique proposed by Hinchliffe and Willis (2003) that adjusts the variables lags. As for the set of terminals, it is composed of the one-step delayed input ($u(k-1)$) and output ($y(k-1)$) and a constant value "1". The latter is responsible to determine whether a constant value will be included in the model or not. Figure 1 shows the MGGP representation (Figure 1(a)) as a 4 gene individual in which each gene is a GP individual (Figure 1(b)). The GP tree represented shows an example of the delay function operator use. In this case, $q1$ is responsible for applying 1-step lag over the variable $y(k-1)$, that becomes $y(k-2)$. The use of the back-shift operator confers flexibility to the algorithm. The variable lag can be incremented by just stacking another operator at the variable sub-tree.

The user must set 3 parameters to constrain the individual representation, which are:

- the *maximum GP height*. This parameter restrains the terms maximum non-linearity degree and maximum lags. For example, Figure 1(b) represents a GP of height 2, which is able to represent a maximum of fourth order polynomial with on-step lagged variable $\big(mul(mul(x1,x1),mul(x1,x1)) = x^4(k-1)\big)$ or one variable with a maximum lag determined by the use of two delay function operators ($q^{-1}q^{-1}x1 = x(k-3)$);

- the *maximum number of MGGP terms*. This parameter corresponds to the maximum number of *genes* in a *chromosome*. MGGP population differs from GA population as the first has no fixed chromosome size. This size must be constrained to avoid large individuals which could lead to high computational cost.

- the set of *delay functions*. It will constrain the search space regarding the variable lags. A GP of maximum height 2 and a set of delay functions restricted to q^{-1} is able to represent a maximum 3 lagged variable ($q^{-1}q^{-1}x(k-1)$), whereas with a q^{-2} function, it is able to represent a 5 lagged variable ($q^{-2}q^{-2}x(k-1)$). The user can use a set of different back-shift operator, *e.g.* $\left[q^{-1}, q^{-2}, q^{-3}\right]$.

About the *genetic operators*, MGGP works with two levels of crossover: the *low level crossover* and *high level crossover*. In the first, one gene is randomly selected from each parent and its GP individuals exchange sub-trees as genetic material. While in the second, the genetic materials are exchanged as entire basis functions, that is, the MGGP parents exchanges its GP individuals in a way similar to the GA one-point crossover. It is important to notice that, in *high level crossover*, the resultant offspring can have different sizes, even from its parents. It occurs because the selected point of crossover is chosen for each parent. In this sense, the MGGP algorithm works with fluctuating individuals size, which increases its flexibility. It is used two kinds of mutation as well, an inner mutation, that occurs as a GP sub-tree mutation, and an outer mutation, which swaps a gene for a new one, with an entirely new GP. As recommended for GP evolution (Poli et al., 2008), either the individual suffers crossover or mutation. Both operators will not be applied to the same individual. Figure 2 exhibits the algorithm flowchart. It begins with an initial population that is evaluated. Then, the generations loop starts: *i)* parent individuals are selected via tournament; *ii)* each parent couple has a chance to be recombined (*CXPB*); *iii)* each individual which has not been recombined has a chance to be mutated (*MTPB*); *iv)* individuals are evaluated; and *v)* elitism operator is applied.

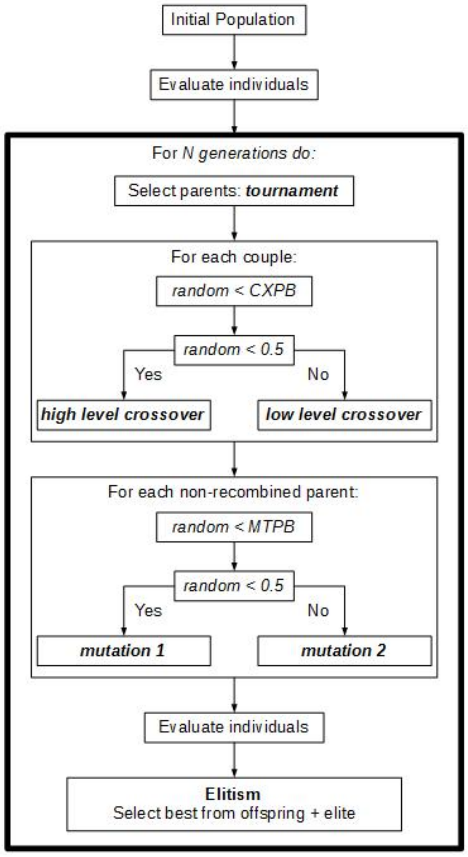
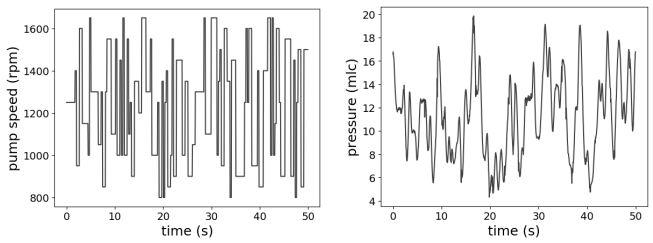


Figure 2. Algorithm flowchart

The evaluation process is composed of three steps: *i)* classical Gram-Smith OLS/ERR structure selection is applied to remove spurious terms and to estimate the parameters, which are equivalent to the ones from traditional Least Squares; *ii)* those parameters are used to simulate the one-step-ahead prediction and the residuals are used in a five iteration ELS process to remove parameter estimation



(a) Pump speed reference (b) System output pressure

Figure 3. Experimental data from a hydraulic pumping system

bias; and *iii)* fitness calculation. It follows the function definition script:

```
Evaluation(individual):
    theta,individual = OLS/ERR(individual)
    theta = ELS(individual, theta)
    fitness = score_MSE(individual, theta)
```

The OLS/ERR method returns an estimate for the parameters and a pruned model. The ELS method removes estimation biases. score_MSE applies the fitness function over the pruned individual using the unbiased parameters.

Two *fitness functions* are implemented: *i)* the one-step-ahead prediction error (PE) of the NARMAX model, in which the MA part is extended considering the problem as a white noise OE problem (see 2.5); and *ii)* the free-run simulation error (SE), in which the MA part is neglected. Both of them are *mean squared error* (MSE).

In order to assess the algorithm performance, it is run 20 Monte Carlo simulations for the following test systems (from Madar et al. (2005) and Piroddi and Spinelli (2003)):

$$\begin{aligned} S_1: \; y(k) =& 0.8u(k-1)^2 + 1.2y(k-1) \\ & - 0.9y(k-2) - 0.2, \end{aligned}$$

$$\begin{aligned} S_2: \; y(k) =& 0.75y(k-2) + 0.25u(k-1) \\ & - 0.2y(k-2)u(k-1), \end{aligned}$$

where the input signal $u(k)$ is a white Gaussian noise with zero mean and variance one ($u(k) = WGN(0,1)$). The assessment is yielded for *output error problem* - varying OE level with EE level fixed to zero ($v(k) = 0$) - and for *equation error problem* - varying EE level with OE level fixed to zero ($e(k) = 0$). The results are compared to the ones from other 20 Monte Carlo simulations in which is applied only the classical Gram-Smith OLS algorithm (Chen et al., 1989) for structure selection using the parameters ($l = 3, n_y = 3, n_u = 3$) to build the whole candidate regressors matrix. Only PE fitness function is used in this test.

The following set up was used for training:

```
population size = 300
crossover probability = 0.8
mutation probability = 0.2 (if not crossover)
OLS tolerance = 1e-3
generations = 100
maximum GP height = 3
maximum MGGP terms = 10
elitism = 10%
delay functions = q1, q2, q3
fitness function = PE
```

Then, the algorithm is tested in an experimental hydraulic pumping system data presented in Barbosa et al. (2011) and Barbosa et al. (2019). Figure 3 shows part of the input and output training data. In this case, a different set-up was used. At each evaluation step, there is a chance ($OLSPB$) of being applied the OLS/ERR method, in which tolerance is randomly selected between 0 and 10^{-7}. This randomness in the OLS/ERR tolerance includes the possibility of having different levels of pruning. The maximum value of tolerance is empirically defined. This experiment is conducted for both fitness functions (PE and SE).

```
population size = 500
crossover probability = 0.8
mutation probability = 0.2 (if not crossover)
OLSPB = 0.2
OLS tolerance = random()*1e-7
generations = 20*
maximum GP height = 5
maximum MGGP terms = 50
elitism = 10%
delay functions = q1, q2, q3, q4, q5
fitness function = PE or SE
```

*The algorithm was run several times until the fittest individual stopped evolving. At each execution, the *elite* from the previous result was included in the next initial population.

4. RESULTS

4.1 Test Systems

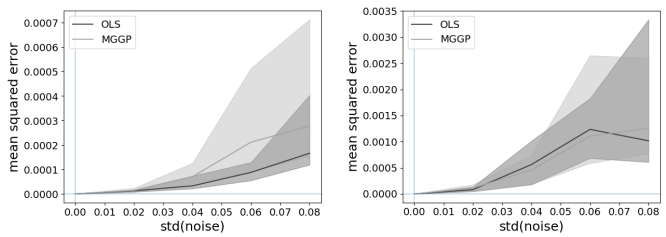
The Tables 1 and 2 show the performance of the proposed MGGP/ERR algorithm and the OLS/ERR algorithm, respectively, regarding the number of times the algorithms were able to select every term present in the model (in 20 Monte Carlo executions) and the average size of the selected models for increasing the noise level in OE (e) and EE (v) problems. Figure 4 presents the free-run simulation errors of the selected models for validation data.

Some considerations can be made from those results:

- increasing the noise level in OE and EE has no effect over the structure selection for both MGGP and OLS algorithms, since the number of times they select all correct terms does not correlate with the noise level;

- considering the Tables 1 and 2, the MGGP algorithm has better performance than the OLS for system S_1, and the opposite for system S_2. In the first, the mean of correct selected models for OE and EE problems, respectively, are 15.25 and 19.50 for MGGP and 13.00 and 12.75 for OLS (out of 20). As for the second, it is 17.00 and 16.75 for MGGP against 20.00 and 20.00 for OLS. These results could lead to the understanding that the best algorithm to be used depends on the system and data quality; however,

- from Figure 4, the OLS algorithm had better performance for both systems S_1 and S_2 in the OE problem, regarding the interquartile range for higher noisy data. Although the



(a) System S_1 - OE problem (b) System S_1 - EE problem

(c) System S_2 - OE problem (d) System S_2 - EE problem

Figure 4. The free-run mean squared errors (or simulation errors) for the selected models via OLS (dark gray) and MGGP (light gray) algorithms for the cases of increasing output error (OE) and increasing equation error (EE). The shaded plots represent the interquartile range and the inner lines represent the median value.

Table 1. MGGP selection results

	Models with correct terms			
	$std(e)$ - OE			
	0.02	0.04	0.06	0.08
S_1 (4 terms)	14/20	18/20	15/20	14/20
average size	4.7	4.85	4.75	4.75
S_2 (3 terms)	18/20	17/20	17/20	16/20
average size	3.05	3.25	3.2	4
	$std(v)$ - EE			
	0.02	0.04	0.06	0.08
S_1 (4 terms)	20/20	20/20	20/20	18/20
average size	4.6	4.95	4.55	4.55
S_2 (3 terms)	15/20	17/20	16/20	19/20
average size	3.1	3.1	3.1	3.05

MGGP algorithm selected more times the correct terms (mean of 15.25 times against 13.00 from OLS algorithm) it got more spurious terms that affect negatively the modeling of the system underlying the data. The 'MGGP' upper quartile for $std(e) = 0.08$ is around 0.0007, while the 'OLS' upper quartile is around 0.0004. Depending on the application this difference can be neglected. Anyhow, the MGGP algorithm can be run again to look for a better result. As for the increasing noise level in the EE problem, both algorithms had similar performances.

4.2 Hydraulic pumping system

Barbosa et al. (2011) work with two models which parameters were estimated by ELS method. Their performances are described in Table 3. The model *"OLS (15)"* was found via OLS algorithm for $l = 2$, $n_y = 6$ and $n_u = 6$; and the model *"OLS (17)"* for $l = 3$, $n_y = 6$ and $n_u = 6$. In this paper, two models were sought by MGGP algorithm. The first from PEM technique (using PE cost function) and

Table 2. OLS selection results

	Models with correct terms			
	$std(e)$ - OE			
	0.02	0.04	0.06	0.08
S_1 (4 terms)	13/20	13/20	13/20	13/20
average size	4.75	4.7	4.85	4.9
S_2 (3 terms)	20/20	20/20	20/20	20/20
average size	3	3	3	3
	$std(v)$ - EE			
	0.02	0.04	0.06	0.08
S_1 (4 terms)	12/20	13/20	14/20	12/20
average size	4.8	4.75	4.75	4.85
S_2 (3 terms)	20/20	20/20	20/20	20/20
average size 3	3	3	3	3

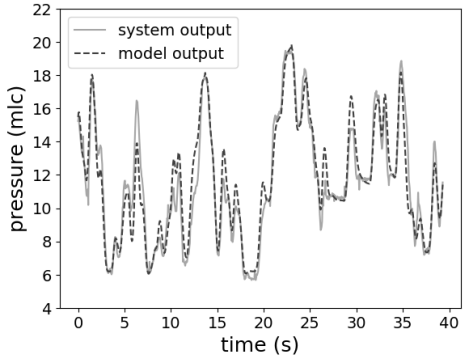
the second from SEM technique (using SE cost function). These models are, respectively, defined by the equations:

$$
\begin{aligned}
y(k) &= \theta_1 y(k-1) + \theta_2 y(k-4) + \theta_3 y(k-12) \\
&+ \theta_4 u(k-4) + \theta_5 u(k-13) + \theta_6 y(k-1)y(k-2) \\
&+ \theta_7 y(k-1)y(k-3) + \theta_8 y(k-1)u(k-4) \\
&+ \theta_9 y(k-1)u(k-6) + \theta_{10} y(k-5)u(k-2) \\
&+ \theta_{11} y(k-5)u(k-7) + \theta_{12} y(k-8)u(k-7) \\
&+ \theta_{13} u(k-1)u(k-4) + \theta_{14} u(k-1)u(k-6) \\
&+ \theta_{15} u(k-4)u(k-6) + \theta_{16} y(k-1)^2 y(k-2) \\
&+ \theta_{17} y(k-1)y(k-3)^2 + \theta_{18} y(k-1)y(k-5)^2 \\
&+ \theta_{19} y(k-3)^2 u(k-3) + \theta_{20} y(k-1)u(k-2)^2 \\
&+ \theta_{21} y(k-1)y(k-3)u(k-4) \\
&+ \theta_{22} y(k-1)u(k-3)u(k-5) \\
&+ \theta_{23} y(k-7)^2 u(k-2) + \theta_{24} y(k-1)^3 u(k-6) \\
&+ \theta_{25} y(k-1)y(k-8)^2 u(k-1) \\
&+ \theta_{26} y(k-1)y(k-3)^2 u(k-3) \\
&+ \theta_{27} y(k-2)y(k-5)^2 u(k-5) \\
&+ \theta_{28} y(k-3)^2 u(k-4)^2 \\
&+ \theta_{29} y(k-3)^2 y(k-5)u(k-5) \\
&+ \theta_{30} y(k-8)^2 u(k-1)u(k-6) \\
&+ \theta_{31} y(k-2)y(k-3)y(k-5)u(k-4) \\
&+ \theta_{32} y(k-1)^3 y(k-7)u(k-7) + \xi_{MA}(k),
\end{aligned}
\tag{10}
$$

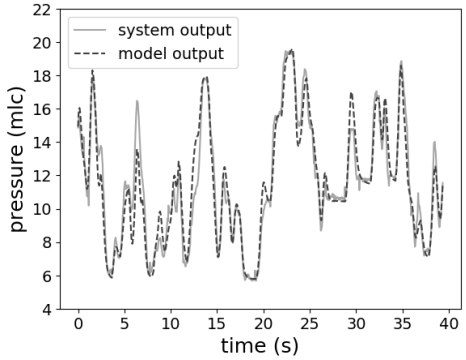
$$
\begin{aligned}
y(k) &= \theta_1 y(k-7) + \theta_2 u(k-4) + \theta_3 u(k-10) \\
&+ \theta_4 u(k-11) + \theta_5 y(k-1)y(k-3) \\
&+ \theta_6 y(k-1)y(k-2) + \theta_7 y(k-1)u(k-4) \\
&+ \theta_8 y(k-5)u(k-7) + \theta_9 y(k-6)u(k-4) \\
&+ \theta_{10} y(k-6)u(k-10) + \theta_{11} y(k-3)^3 \\
&+ \theta_{12} y(k-1)^2 y(k-2) + \theta_{13} y(k-3)^2 u(k-4) \\
&+ \theta_{14} y(k-2)y(k-4)y(k-7) \\
&+ \theta_{15} y(k-1)y(k-4)u(k-6) \\
&+ \theta_{16} y(k-6)y(k-9)u(k-12) \\
&+ \theta_{17} y(k-1)u(k-5)u(k-12) + \theta_{18} u(k-4)^3 \\
&+ \theta_{19} u(k-5)^2 u(k-10) + \theta_{20} y(k-3)^2 u(k-4)^2 \\
&+ \theta_{21} y(k-3)y(k-4)y(k-7)u(k-10) \\
&+ \theta_{22} y(k-2)y(k-3)y(k-5)u(k-4) \\
&+ \theta_{23} y(k-2)y(k-3)^2 u(k-6) \\
&+ \theta_{24} y(k-5)^2 u(k-5)u(k-6) \\
&+ \theta_{25} y(k-1)^3 y(k-5)u(k-4) \\
&+ \xi_{MA}(k),
\end{aligned}
\tag{11}
$$

Table 3. Hydraulic pump models. The first two are from Barbosa et al. (2011) obtained via OLS method. J_P is the one-step-ahead MSE (mlc^2); J_S is the free-run MSE (mlc^2); N_P the number of parameters present in the model; and *max.* N_P is the number of all possible regressors for given (l,n_y,n_u).

Model	J_P(Ident.)	J_P(Val.)	J_S(Ident.)	J_S(Val.)	N_P	(l, n_y, n_u)	max. N_P
OLS (15)	0.100	0.082	2.600	2.243	17	$(2, 6, 6)$	90
OLS (17)	0.086	0.070	1.447	1.120	23	$(3, 6, 6)$	454
MGGP (10)	0.067	0.058	1.910	1.150	32	$(5, 12, 13)$	142505
MGGP (11)	0.088	0.068	1.221	0.997	25	$(5, 9, 12)$	65779



(a)



(b)

Figure 5. Free-run simulation of models obtained by MGGP algorithm for the hydraulic pump validation data: (a) Model (10) - PEM technique and (b) Model (11) - SEM technique

where ξ_{MA} represents the noise model. The models performances are described in Table 3. Figures 5(a) and 5(b) show the models free-run simulation in comparison with the expected validation data.

Model (10) has the best one-step-ahead prediction results. Comparing with the best model from Barbosa et al. (2011), Model (10) has $J_P = 0.067$ mlc^2 against $J_P = 0.086$ mlc^2 for training data and $J_P = 0.058$ mlc^2 against $J_P = 0.070$ mlc^2 for validation data. However, its free-run simulation error got worsen (from $J_S = 1.447$ mlc^2 to $J_S = 1.910$ mlc^2 for training data and from $J_S = 1.120$ mlc^2 to $J_S = 1.150$ mlc^2 for validation data). Still, the algorithm was trained with PE cost function, which means it accomplished its objective and found a better one-step-ahead predictor model for the hydraulic pump under analysis without critically worsen its free-run prediction.

Model (11) has better one-step-ahead prediction error and free-run simulation error than the best OLS models from Barbosa et al. (2011) for validation data. The errors are $J_P = 0.068$ mlc^2 against $J_P = 0.070$ mlc^2 and $J_S = 0.997$ mlc^2 against $J_S = 1.120$ mlc^2. However, the gain in prediction error is not as good as it is for model (10). Still, the algorithm was trained with SE cost function and it found a model with the best free-run prediction. The choice of which model to use depends on the application. Whether it is necessary better one-step-ahead error or better free-run simulation error.

Table 4. Memory usage for building one full candidate regressors matrix for the hydraulic pumping training data (3200 samples)

	(l, n_y, n_u)			
	$(3, 3, 3)$	$(5, 5, 5)$	$(5, 10, 10)$	$(5, 12, 12)$
Memory (mb)	2.02	73.18	1293.04	≈ 2820.00

It is imperative to highlight that the proposed algorithm built a model of non-linearity degree 5 and maximum input and output delays 13 and 12, respectively ($l = 5$, $n_u = 13$, $n_y = 12$). That means a total of 142505 possible regressors for a traditional OLS analysis (not considering the constant term). Building a matrix this size is already impracticable with the resources applied in this research (personal laptop). Table 4 presents the memory usage for building the full candidate regressor matrix for different sets of the degree of non-linearity, maximum output and input lags (l, n_y, n_u). The memory usage increases exponentially and for the last set $(5, 12, 12)$ a memory error was raised in trying to build the matrix of shape (3188 x 118754). It is shown that the proposed MGGP algorithm can explore a considerably higher search space in comparison to the traditional OLS method without the need for high-performance computers.

5. CONCLUSION

This paper introduced the use of a hybrid MGGP/ERR algorithm for the structure selection of polynomial NARMAX models. It is used a very flexible GP feature (backshift operator) to determine variable lags that dispenses the need of building a whole candidate regressors matrix. The algorithm is assessed in two well-known test systems with short-term dependencies and in a real-life hydraulic pumping identification problem. The OLS/ERR algorithm had better performance than the MGGP/ERR algorithm in selecting the structure for systems with short-term dependencies. However, for the hydraulic pumping identification problem, the MGGP/ERR algorithm was able to find models with better performance than previous works models for prediction error (using PEM technique) and for simulation error (using SEM technique) using longer-term dependencies and higher degree of non-linearity in such a way that OLS/ERR algorithms would require much more

computational power. An extension of this work intends to analyze the integration between MGGP and OLS/ERR algorithms by varying the probability of applying the OLS/ERR algorithm in a more complex test system for PEM and SEM techniques to testify the improvement in using the hybridization of the two algorithms.

REFERENCES

Aguirre, L.A. (2015). *Introdução à Identificação de Sistemas - Técnicas Lineares e Não-Lineares: Teoria e Aplicação*. Editora UFMG, 4^a edition.

Aguirre, L.A. and Letellier, C. (2009). Modeling nonlinear dynamics and chaos: a review. *Mathematical Problems in Engineering*, 2009.

Barbosa, B.H.G., Aguirre, L.A., Martinez, C.B., and Braga, A.P. (2011). Black and gray-box identification of a hydraulic pumping system. *Control Systems Technology, IEEE Transactions on*, 19(2), 398–406.

Barbosa, B.H.G., Aguirre, L.A., Martinez, C.B., and Braga, A.P. (2019). Piecewise affine identification of a hydraulic pumping system using evolutionary computation. *IET Control Theory & Applications*, 13(9), 1394 – 1403.

Billings, S., Chen, S., and Korenberg, M. (1989). Identification of mimo non-linear systems using a forward-regression orthogonal estimator. *International journal of control*, 49(6), 2157–2189.

Castro, H.C. and Barbosa, B.H.G. (2020). Algoritmos multi-objetivos para detecção de estruturas em modelos narx utilizando técnicas pem e sem. In *Anais do 14º Simpósio Brasileiro de Automação Inteligente*. Galoá.

Chen, Q., Worden, K., Peng, P., and Leung, A. (2007). Genetic algorithm with an improved fitness function for (n) arx modelling. *Mechanical Systems and Signal Processing*, 21(2), 994–1007.

Chen, S., Billings, S.A., and Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control*, 50(5), 1873–1896.

Eiben, A.E., Smith, J.E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.

Garg, A., Tai, K., and Panda, B. (2017). System identification: Survey on modeling methods and models. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, 607–615. Springer.

Ghareeb, W.T. and El Saadany, E.F. (2013). Multi-Gene Genetic Programming for Short Term Load Forecasting. In *2013 3rd International Conference on Electric Power and Energy Conversion Systems (EPECS)*, International Conference on Electric Power and Energy Conversion Systems.

Goldberg, D.E. and Holland, J.H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2), 95–99.

Hinchliffe, M.P. (2001). *Dynamic modelling using genetic programming*. Ph.D. thesis, University of Newcastle upon Tyne, UK.

Hinchliffe, M.P. and Willis, M.J. (2003). Dynamic systems modelling using genetic programming. *Computers & Chemical Engineering*, 27(12), 1841 – 1854.

Hinchliffe, M., Willis, M., Hiden, H., Tham, M., McKay, B., and Barton, G. (1996). Modelling chemical process systems using a multi-gene genetic programming algorithm. In *Genetic Programming: Proceedings of the First Annual Conference (late breaking papers)*, 56–65.

Holland, J.H. (1975). Adaptation in natural and artificial systems. *The University of Michigan Press*.

Koza, J.R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

Leontaritis, I. and 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.

Li, C.J. and Jeon, Y. (1993). Genetic algorithm in identifying non linear auto regressive with exogenous input models for non linear systems. In *1993 American Control Conference*, 2305–2309. IEEE.

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

Mehr, A.D. and Kahya, E. (2017). A Pareto-optimal moving average multigene genetic programming model for daily streamflow prediction. *Journal of Hydrology*, 549, 603–615.

Piroddi, L. and Spinelli, W. (2003). An identification algorithm for polynomial narx models based on simulation error minimization. *International Journal of Control*, 76(17), 1767–1781.

Poli, R., Langdon, W.B., and McPhee, N.F. (2008). *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. (With contributions by J. R. Koza).

Pope, K.J. and Rayner, P.J. (1994). Non-linear system identification using bayesian inference. In *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, IV–457. IEEE.

Riahi-Madvar, H., Dehghani, M., Seifi, A., and Singh, V.P. (2019). Pareto Optimal Multigene Genetic Programming for Prediction of Longitudinal Dispersion Coefficient. *Water Resources Management*, 33(3), 905–921.

Safari, M.J.S. and Mehr, A.D. (2018). Multigene genetic programming for sediment transport modeling in sewers for conditions of non-deposition with a bed deposit. *International Journal of Sediment Research*, 33(3), 262–270.

Young, P.C. (1968). The use of linear regression and related procedures for the identification of dynamic processes. In *Seventh Symposium on Adaptive Processes*, 53–53. IEEE.