Genetic Programming for the Identification of Nonlinear Input-Output Models

János Madár, János Abonyi* and Ferenc Szeifert
Department of Process Engineering, University of Veszprém,
P.O. Box 158, Veszprém 8201, Hungary

February 8, 2005

Abstract

Linear-in-parameters models are quite widespread in process engineering, e.g. NAARX, polynomial ARMA models, etc. This paper proposes a new method for structure selection of these models. The method uses Genetic Programming (GP) to generate nonlinear input-output models of dynamical systems that are represented in a tree structure. The main idea of the paper is to apply Orthogonal Least Squares algorithm (OLS) to estimate the contribution of the branches of the tree to the accuracy of the model. This method results in more robust and interpretable models. The proposed approach has been implemented as a freely available MATLAB Toolbox `www.fmt.veim.hu/softcomp`. The simulation results show that the developed tool provides an efficient and fast method for determining the order and the structure for nonlinear input-output models.

Keywords: Structure identification, Genetic Programming, Orthogonal Least Squares, Linear-in-parameters models

*To whom correspondence should be addressed. Tel: +36 88 622793. Fax: +36 88 624171. E-mail: abonyij@fmt.vein.hu.

1 Introduction to Data-Driven System Identification

In this paper, we focus on data-driven identification of nonlinear input-output models of dynamical systems. The data-driven identification of these models involves the following tasks [1]:

a, *Structure selection.* How to select the regressor (model order) and the structure of the nonlinear static functions used to represent the model.

b, *Input sequence design.* Determination of the input sequence which is injected into the modelled object to generate the output sequence that can be used for identification.

c, *Noise modelling.* Determination of the dynamic model which generates the noise.

d, *Parameter estimation.* Estimation of the model parameters from the input-output sequence.

e, *Model validation.* Comparison of the output of the modelled object and the model based on data not used in model development.

Most data-driven identification algorithms assume that the model structure is *a priori* known or that it is selected by a higher-level 'wrapper' structure-selection algorithm. Several information-theoretic criteria have been proposed for structure selection of linear dynamic input-output models. Examples of the classical criteria are the Final Prediction-Error (FPE) and the Akaike Information Criterion (AIC) [2]. Later, the Minimum Description Length (MDL) criterion developed by Schwartz and Rissanen was proven to produce consistent estimates of the structure of linear dynamic models [3]. With these tools, determining the structure of linear systems is a rather straightforward task.

Relatively little research has been done into the structure selection for *nonlinear models.* In the paper of Aguirre and Billings [4] it is argued that if a certain type of term in a nonlinear model is spurious. In [5] this approach is used to the structure selection of polynomial models. In [6] an alternative solution to the model structure selection problem is introduced by conducting a forward search through the many possible candidate model terms initially and then performing an exhaustive all subset model selection on the resulting model. A backward search approach based on orthogonal parameter-estimation is also applied [7, 8].

As can be seen, these techniques are 'wrapped' around a particular model construction method. Hence, the result of the estimate can be biased due to the particular construction method used. To avoid this problem in the recent research a 'model free' approach is followed where no particular model needs to be constructed in order to select the model of the modeled system. The advantage of this approach is that this estimate is based on geometrical/embedding procedures and does not depend on the model representation that will be used a posteriori, i.e. the results would have a rather general character. This is an important advantage, as the construction of a NARX model consists of the selection of many structural parameters which have significant effect to the performance of the designed model: e.g. the model order, type of the nonlinearity (Hammerstein or Wiener type system) [9], scheduling variables, number of neurons in a neural network, etc. The simultaneous selection of these structural parameters is a problematic task. The primary objective of this paper is to decompose this complex problem by providing some useful guidance in selecting a tentative model with a correct model order. Deterministic suitability measures [10] and false nearest neighbor (FNN) algorithms [11] have already been proposed for data-based selection of the model order. These methods build upon similar methods developed for the analysis of chaotic time series [12]. The idea behind the FNN algorithm is geometric in nature. If there is enough information in the regression vector to predict the future output, then for any two regression vectors which are close in the regression space, the corresponding future outputs are also close in the output space. The structure is then selected by computing the percentage of false neighbors, i.e., vectors that violate the above assumption. A suitable threshold parameter must be specified by the user. For this purpose, heuristic rules have been proposed [10]. Unfortunately, for nonlinear systems the choice of this parameter will depend on the particular system under study [11]. The computational effort of this method also rapidly increases with the number of data samples and the dimension of the model. To increase the efficiency of this algorithm, in [13] two clustering-based algorithms have been proposed and the model structure is then estimated on the basis of the cluster covariance matrix eigenvalues.

It should be kept in mind that there is no escape of performing a model-driven structure selection, once a certain model representation is chosen. For instance, suppose one of the above presented "model-free" model order selection algorithm is used to determine the correct model order. If a neural network is used to model the process, the designer still need to decide on the activation function, the number of nodes etc. Therefore, the model order selection method that will be presented in the above mentioned papers definitely not spare the user of having to go through some sort of structure selection. However, this procedure can be fully automatized, since most of these models proven universal representation abilities and it is often possible to find some hybridization with advanced structure optimization techniques that are able to automatically generate models with adequate structural parameters.

This paper proposes such hybridization of Genetic Programming (GP) and the Orthogonal Least Squares (OLS) for the structure selection of nonlinear models that are linear-in-parameters. This method is based on a "tree representation" based symbolic optimization technique developed by John Koza [14]. This representation is extremely flexible, since trees can represent computer programs, mathematical equations or complete models of process systems. This scheme has been already used for circuit design in electronics, algorithm development for quantum computers, and it is suitable for generating model structures: e.g. identification of kinetic orders [15], steady-state models [16], and differential equations [17]. Although these successful applications confirm the applicability of GP in chemical and process engineering, GP cannot be directly used for the identification of nonlinear input-output models. Hence, the aim of this paper is to tailor this GP based techniques to the identification of linear-in-parameters dynamical models by extending the GP operators by an Orthogonal Least Squares based model reduction tool.

The paper is organized as follows: In Sect. 2 the structure of linear-in-parameters models and the OLS are presented, in Sect. 3 a modified GP algorithm is presented which is suitable for linear-in-parameters models and polynomial models. Finally in Sect. 4 the application examples are shown.

2 Linear-in-Parameters Models

When the information necessary to build a fundamental model of dynamical processes is lacking or renders a model that is too complex for an on-line use, empirical modeling is a viable alternative. Empirical modeling and identification is a process of transforming available input-output data into a functional relation that can be used to predict future trends. In this section, before the discussion of the GP based model structure identification algorithm, the most widely used linear-in-parameters model structures will be reviewed.

2.1 Introduction to Linear-in-parameters Models

The identification of a discrete input-output dynamical model is based on the observed inputs $\{u(k)\}_k$ and outputs $\{y(k)\}_k$ [18],

$$\{u(k)\}_k = [u(1), u(2), \ldots, u(k)] \,, \tag{1}$$

$$\{y(k)\}_k = [y(1), y(2), \ldots, y(k)] \,, \tag{2}$$

Our aim is to find a relationship between past observations and future output. Instead of using the whole previous input-output sequence, $\{u(k-1)\}_k$ and $\{y(k-1)\}_k$, a finite-dimensional regression vector, $\mathbf{x}(k)$, can be used which is a subset of the previous input and output variables of the $f(.)$ model

$$\widehat{y}(k) = f(\mathbf{x}(k), \theta) \,. \tag{3}$$

$$\begin{aligned} \mathbf{x}(k) = (&u(k-n_d-1), \cdots, u(k-n_d-n_b), \\ &y(k-n_d-1), \cdots, y(k-n_d-n_a), \end{aligned} \tag{4}$$

where the $\mathbf{x}(k)$ input vector of model consists of the lagged u input and y output, while n_d represents the dead-time, n_b, n_a are the input- and output-orders, respectively.

Many general nonlinear model structures (like neural networks) can be used to represent such models, only the θ parameters of the model have to be estimated based on the available input-output data. In some cases the excessive number of unknown coefficients leads to ill-conditioned estimation problem causing numerical difficulties and high sensitivity to measurement errors. Furthermore, nonlinear optimization algorithms used to the identification of these parameters may get stuck in local minima.

To handle these difficulties this paper proposes an approach based on the Gabor–Kolmogorov Analysis of Variance (ANOVA) decomposition a general nonlinear function:

$$\begin{aligned} \widehat{y}(k) = f(\mathbf{x}(k), \theta) \simeq f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} f_{ij}(x_i, x_j) + \cdots \\ + f_{1,2,\ldots,n}(x_1, \ldots, x_n), \end{aligned} \tag{5}$$

where the $f(\mathbf{x}(k), \theta)$ function is approximated by an additive decomposition of simpler subfunctions; in which f_0 is a bias term and $f_i(x_i), f_{ij}(x_i, x_j), \ldots$ represent univariate, bivariate, $\ldots$ components. Any function, and hence any reasonable dynamical system can be represented by this decomposition.

Therefore, this ANOVA approach can be easily used to the input-output based modelling of dynamical systems

With the use of this definition all the linear-in-parameters models that are used in process engineering can be obtained, such as Nonlinear Additive AutoRegressive models (NAARX), Volterra models or Polynomial ARMA models:

- NAARX Nonlinear Additive AutoRegressive models with eXogenous inputs models are defined as [19]

$$\hat{y}(k) = \sum_{i=1}^{n_a} f_i(y(k-i)) + \sum_{j=1}^{n_b} g_j(u(k-j)) + e(k) \tag{6}$$

  where the functions f_i and g_i are scalar nonlinearities, and $e(k)$ represents the modeling error. As can be seen, this model does not permit 'cross terms' involving products of input and output values at different times.

- Volterra models are defined as multiple convolution sums

$$\begin{aligned} \hat{y}(k) = y_0 + \textstyle\sum_{i=1}^{n_b} b_i u(k-i) \\ + \textstyle\sum_{i=1}^{n_b} \sum_{j=1}^{n_b} b_{ij} u(k-i) u(k-j) + \cdots + e(k)\,. \end{aligned} \tag{7}$$

- Polynomial ARMA models are superior to Volterra series models in the sense that the number of parameters needed to approximate a system is generally much less with polynomial models [20] because of the use of previous output values.

$$\begin{aligned} \hat{y}(k) = y_0 + \sum_{i=1}^{n_a} a_{1,i} y(k-i) + \sum_{i=1}^{n_b} b_{1,i} u(k-i) \\ + \sum_{i=1}^{n_a} \sum_{j=1}^{i} a_{1,ij} y(k-i) y(k-j) \\ + \sum_{i=1}^{n_b} \sum_{j=1}^{i} b_{2,ij} u(k-i) u(k-j) + \ldots + e(k)\,. \end{aligned} \tag{8}$$

Since certain input and output interactions can be redundant, some components of the ANOVA decomposition can be ignored that can result in a more parsimonious and adequate representation. Hence, the aim of this paper is to present an efficient method for the data-driven selection of the model order (n_d, n_a, n_b) and the model structure that is member of the above presented model classes.

2.2 Model Structure Selection for Linear-in-parameters Models

Linear-in-parameters models can be formulated as:

$$y(k) = \sum_{i=1}^{M} p_i F_i\left(\mathbf{x}(k)\right), \tag{9}$$

where $F_1, \ldots, F_M$ are nonlinear functions (they do not contain parameters), and $p_1, \ldots, p_M$ are model parameters. The problem of model structure selection for linear-in-parameters models is to find the proper set of nonlinear functions. To attack this problem two approaches can be distinguished:

- The first approach generates all of the possible model structures and selects the best.
- The second approach transforms the problem into an optimization problem and solves it based on a (heuristic) search algorithm.

The bottleneck of the first approach is that there is a vast number of possible structures, hence, in practice, it is impossible to evaluate all of them. Even, if the set of the possible structures is restricted only to polynomial models

$$\begin{aligned} y(k) = p_0 + \sum_{i_1=1}^{m} p_{i_1} x_{i_1}(k) + \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} p_{i_1 i_2} x_{i_1}(k) x_{i_2}(k) \\ + \cdots + \sum_{i_1=1}^{m} \cdots \sum_{i_d=i_{d-1}}^{m} p_{i_1 \cdots i_d} \prod_{j=1}^{m} x_{i_j}(k), \end{aligned} \tag{10}$$

the number of possible terms could be very large. If the number of regressors is m and the maximum polynomial degree is d, the number of parameters (number of polynomial terms) is

$$n_p = \frac{(d+m)!}{d!m!}\,. \tag{11}$$

E.g. if $m = 8$ and $d = 4$, $n_p = 495$.

In case of reasonable number of regressors (submodels) the first approach can be followed: the polynomial terms are sorted based on their error reduction ratio, and the best terms are selected.

In case of larger model orders and higher polynomial degree the first approach cannot be followed due to the complexity of the initial model.

Hence, the second approach to the structure selection problem should be used that transforms the structure selection problem into an optimization problem, in which the search space consists of possible structures. This method uses a search algorithm, which looks for the optimal structure. This paper suggests the application of Genetic Programming to this task.

3 Genetic Programming for Linear-in-parameters Models

Genetic Programming is a symbolic optimization technique, developed by John Koza [14]. It is an evolutionary computation technique (like e.g. Genetic Algorithm, Evolutionary Strategy) based on so called "tree representation". This representation is extremely flexible, since trees can represent computer programs, mathematical equations or complete models of process systems. Because the algorithm of Genetic Programming is well-known, we will not present the details of the algorithm but focus here on the specific details. It should be noted that there are several variants of Genetic Programming, e.g. Gene Expression Programming [21], the [22] provides a good general review of the GP algorithm we used in this paper.

3.1 Model Representation

Unlike common optimization methods, in which potential solutions are represented as numbers (usually vector of real numbers), the symbolic optimization algorithms represent the potential solutions by structured ordering of several symbols. One of the most popular method for representing structures is the binary tree. A population member in GP is a hierarchically structured tree consisting of functions and terminals. The functions and terminals are selected from a set of functions (operators) and a set of terminals. For example, the set of operators F can contain the basic arithmetic operations: $F = \{+, -, *, /\}$; however, it may also include other mathematical functions, Boolean operators, conditional operators or Automatically Defined Functions (ADFs). ADFs [23] are sub-trees which are used as functions in the main tree, and they are varied in the same manner as the main trees. It is especially worth using of ADF if the problem is regularity-rich, because the GP with ADF may solve these problems in a hierarchical way (e.g. chip design). In this work we only used arithmetic operations and mathematical functions (see Sect. 4). The set of terminals T contains the arguments for the functions. For example $T = \{y, x, p_i\}$ with x and y being two independent variables, and p_i represents the parameters. Now, a potential solution may be depicted as a rooted, labeled tree with ordered branches, using operations (internal nodes of the tree) from the function set and arguments (terminal nodes of the tree) from the terminal set.

Generally, GP creates nonlinear models and not only linear-in-parameters models. To avoid nonlinear in parameter models the parameters must be removed from the set of terminals, i.e. it contains only variables: $T = \{x_1(k), \cdots, x_m(k)\}$, where $x_i(k)$ denotes the i-th regressor variable. Hence a population member represents only the F_i nonlinear functions (9). The parameters are assigned to the model after 'extracting' the F_i function terms from the tree, and they are determined using LS algorithm (14).

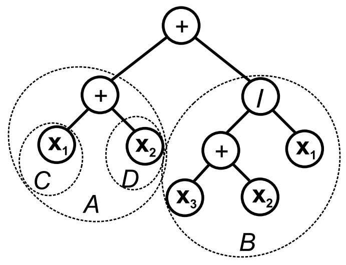

Figure 1: Decomposition of a tree to function terms

We used a simple method for the decomposition of the tree into function terms. The subtrees, which represent the F_i function terms, were determined by decomposing the tree starting from the root as far as reaching non-linear nodes (nodes which are not '+' or '-'). E.g. let us see Fig. 1. The root node is a '+' operator, so it is possible to decompose the tree into two subtrees: 'A' and 'B' trees. The root node of the 'A' tree is again a linear operator, so it can be decomposed into 'C' and 'D' trees. The root node of the 'B' tree is a nonlinear node ('/') so it cannot be decomposed. The root nodes of 'C' and 'D' trees are nonlinear too, so finally the decomposition procedure results in three subtrees: 'B', 'C' and 'D'. Based on the result of the decomposition it is possible to assign parameters to the functional terms represented by the obtained subtrees. In the case of this example the resulted linear-in-parameters model is: $y = p_0 + p_1(x_3 + x_2)/x_1 + p_2x_1 + p_3x_3$. Certainly, one may use other decomposition methods (which may lead different results, e.g. $y = p_0+p_1x_3/x_1+p_2x_2/x_1+p_3x_1+p_4x_3$), however this type of decomposition would not use the benefits the GP and OLS reduction algorithms.

GP can be used for the selection from special model classes, such as polynomial models. To achieve this goal, one has to restrict the set of operators and introduce some simple syntactic rules. For example, if the set of operators is defined as $F = \{+, *\}$, and there is a syntactic rule that exchanges the internal nodes that are below a '$*$'-type internal nodes to '$*$'-type nodes; the GP generates models that are in the polynomial NARX model class.

3.2 Fitness Function

Genetic Programming is an Evolutionary Algorithm. It works with a set of individuals (potential solutions), and these individuals form a generation. In every iteration, the algorithm evaluates the individuals, selects individuals for reproduction, generates new individuals by mutation, crossover and direct reproduction, and finally creates the new generation.

In the *selection* step the algorithm selects the parents of the next generation and determines which individuals survive from the current generation. The fitness function reflects the goodness of a potential solution which is proportional to the probability of the selection of the individual. Usually, the fitness function is based on the mean square error (MSE) between the calculated and the measured output values,

$$\chi^2 = \frac{1}{N} \sum_{k=1}^{N} \left(y(k) - \sum_{i=1}^{M} p_i F_i \left(\mathbf{x}(k) \right) \right), \tag{12}$$

where N is the number of the data-points used for the identification of the model. Instead of MSE, in symbolic optimization often the correlation coefficient, r, of the measured and the calculated output values are used [24].

A good model is not only accurate but simple, transparent and interpretable. In addition, a complex over-parameterized model decreases the general estimation performance of the model. Because GP can result in too complex models, there is a need for a fitness function that ensures a tradeoff between complexity and model accuracy. Hence, [16] suggests the incorporation of a penalty term into the fitness function:

$$f_i = \frac{r_i}{1 + \exp\left(a_1(L_i - a_2)\right)}, \tag{13}$$

where f_i is the calculated fitness value, r_i is the correlation coefficient, L_i is the size of the tree (number of nodes), a_1 and a_2 are parameters of penalty function.

In practice, a model which gives good prediction performance on the training data may be over-parameterized and may contain unnecessary, complex terms. The penalty function (13) handles this difficulty, because it decreases fitness values of trees that have complex terms. However, parameters of this penalty term are not easy to determine and the penalty function does not provide efficient solution for this difficulty. An efficient solution may be the elimination of complex and unnecessary terms from the model. For linear-in-parameters models it can be done by the Orthogonal Least Squares (OLS) algorithm.

3.3 Orthogonal Least Squares (OLS) Algorithm

The great advantage of using linear-in-parameters models is that the Least Squares Method (LS) can be used for the identification of the model parameters, which is much less computationally demanding than other nonlinear optimization algorithms, since the optimal $\mathbf{p} = [p_1, \ldots, p_M]^T$ parameter vector can be analytically calculated:

$$\mathbf{p} = \left(\mathbf{F}^{-1}\mathbf{F}\right)^{\mathrm{T}} \mathbf{F}\mathbf{y}, \tag{14}$$

where $\mathbf{y} = [y(1), \ldots, y(N)]^T$ is the measured output vector, and the $\mathbf{F}$ regression matrix is:

$$\mathbf{F} = \begin{pmatrix} F_1(\mathbf{x}(1)) & \ldots & F_M(\mathbf{x}(1)) \\ \vdots & \ddots & \vdots \\ F_1(\mathbf{x}(N)) & \ldots & F_M(\mathbf{x}(N)) \end{pmatrix}. \tag{15}$$

In case most of process systems certain input and output interactions will be redundant and hence components in the ANOVA decomposition could be ignored, which can result in more parsimonious representations. The OLS algorithm [25, 26] is an effective algorithm to determine which terms are significant in a linear-in-parameters model. The OLS introduces the error reduction ratio (*err*) which is a measure of the decrease in the variance of output by a given term.

The compact matrix form corresponding to the linear-in-parameters model (9) is

$$\mathbf{y} = \mathbf{F}\mathbf{p} + \mathbf{e}, \tag{16}$$

where the $\mathbf{F}$ is the regression matrix (15), $\mathbf{p}$ is the parameter vector, $\mathbf{e}$ is the error vector. The OLS technique transforms the columns of the $\mathbf{F}$ matrix

(15) into a set of orthogonal basis vectors in order to inspect the individual contributions of each terms.

The OLS algorithm assumes that the regression matrix $\mathbf{F}$ can be orthogonally decomposed as $\mathbf{F} = \mathbf{WA}$, where $\mathbf{A}$ is an $M \times M$ upper triangular matrix (it means $A_{i,j} = 0$ if $i > j$) and $\mathbf{W}$ is an $N \times M$ matrix with orthogonal columns in the sense that $\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{D}$ is a diagonal matrix. (N is the length of $\mathbf{y}$ vector and M is the number of regressors.) After this decomposition one can calculate the OLS auxiliary parameter vector $\mathbf{g}$ as

$$\mathbf{g} = \mathbf{D}^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{y}, \tag{17}$$

where g_i is the corresponding element of the OLS solution vector. The output variance $(\mathbf{y}^{\mathrm{T}}\mathbf{y})/N$ can be explained as

$$\mathbf{y}^{\mathrm{T}}\mathbf{y} = \sum_{i=1}^{M} g_i^2 w_i^{\mathrm{T}} w_i + \mathbf{e}^{\mathrm{T}}\mathbf{e}. \tag{18}$$

Thus the error reduction ratio, $[err]_i$ of F_i term can be expressed as

$$[err]_i = \frac{g_i^2 w_i^{\mathrm{T}} w_i}{\mathbf{y}^{\mathrm{T}}\mathbf{y}}. \tag{19}$$

This ratio offers a simple mean for order and select the model terms of a linear-in-parameters model according to their contribution to the performance of the model.

3.4 GP and OLS

To improve the GP algorithm, this paper suggests the application of OLS in the GP algorithm. During the operation of GP the algorithm generates a lot of potential solutions in the form of a tree-structure. These trees may have terms (subtrees) that contribute more or less to the accuracy of the model.

The concept is the following: firstly the trees (the individual members of the population) are decomposed to subtrees (function terms of the linear-in-parameters models) in such a way it was presented in Sect. 3.1; then the error reduction ratios of these function terms are calculated; finally the less significant term(s) is/are eliminated. This "tree pruning" method is realized in every fitness evaluation before the calculation of the fitness values of the trees. The main goal of the application of this approach is to transform the trees to simpler trees which are more transparent, but their accuracy are close to the original trees. Because the further goal is to preserve the original structure of the trees as far as it possible (because the genetic programming works with the tree structure) the decomposition of trees was based on the algorithm presented in Sect. 3.1. This method always guarantees that the elimination of one or more function terms of the model can be done by ”pruning” the corresponding subtrees, so there is no need for structural rearrangement of the tree after this operation.

Let us see a simple example that illustrates how the proposed method works. This example is taken from the Example I (see Sect. 4.2), where the function, which must be identified, is $y(k) = 0.8u(k-1)^2 + 1.2y(k-1) - 0.9y(k-2) - 0.2$. After a few generation the GP algorithm found a solution with four terms: $u(k-1)^2$, $y(k-1)$, $y(k-2)$, $u(k-1) * u(k-2)$ (see Fig. 2). Table 1 shows the calculated error reduction ratio values for these function terms and the mean square error of the linear-in-parameters model represented by this tree. Based on the OLS, the subtree that had the least error reduction ratio ($F_4 = u(k-1) * u(k-2)$) was eliminated from the tree. After that the error reduction ratios and the MSE (and the parameters) were calculated again. The results shows that the new model has a little higher mean square error but it has more adequate structure.

Table 1: OLS example

	Before OLS	After OLS
$[err]_1$	0.9170	0.7902
$[err]_2$	0.0305	0.1288
$[err]_3$	0.0448	0.0733
$[err]_4$	0.0002	–
MSE	0.558	0.574

There are several possibilities to apply this pruning approach. The pruning of the tree can be done in every fitness evaluation. In the application examples an $[err]_{limit}$ parameter has been used which determines the minimal allowed $[err]$ values for valid function terms. According to this strategy the algorithm eliminates the subtrees which has smaller error reduction ratios than this parameter.

4 Application Examples

In this section the application of the proposed GP-OLS technique is illustrated. Firstly the developed MATLAB GP-OLS Toolbox is presented that

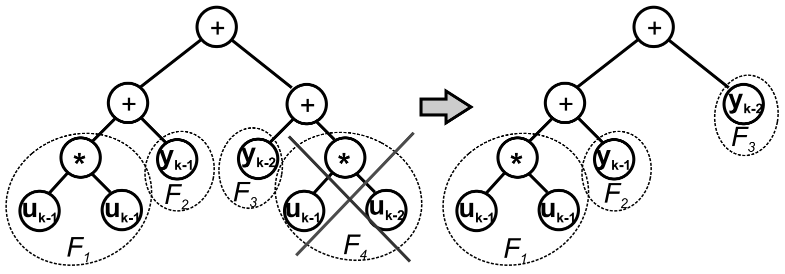

Figure 2: OLS example

was used in the case studies of the paper. In the first example, the structure of a known input-output model is identified. This example illustrates that the proposed OLS method improves the performance of GP and it is able to correctly identify the structure of nonlinear systems that are member of the class of linear-in-parameters models. In the second example the model order of a continuous polymerization reactor is estimated. This example is used for the illustration that the proposed approach is a useful tool for the selection of the model order of nonlinear systems. Finally, a more detailed example is given where both the order and the structure of the model of a nonlinear unstable chemical reactor are estimated.

4.1 The MATLAB GP-OLS Toolbox

The proposed approach has been implemented in MATLAB that is the most widely applied rapid prototyping system [27].

The aim of the toolbox is the data-based identification of static and dynamic models, since the approach proposed in this paper is can also be applied for static nonlinear equation discovery.

At the development of the toolbox special attention has been given to the identification of dynamical input-output models. Hence, the generated model equations can be simulated to get one- and/or n-step ahead predictions.

The toolbox is freeware, and it is downloadable from the website of the authors: `www.fmt.veim.hu/softcomp`. The toolbox has a very simple and user-friendly interface. The user should only define the input-output data, the set of the terminal nodes (the set of the variables of the model, which in case of a dynamical system means the maximum estimate of the input-output model orders), select the set of the internal nodes (the set of mathematical operators) and set some parameters of the GP.

Based on our experiments we found that with the parameters given in Table 2 the GP is able to find good solutions for various problems. Hence these parameters are the default parameters of the toolbox that have not been modified during the simulation experiments presented in this paper.

Table 2: Parameters of GP in the application examples

Population size	50
Maximum number of evaluated individuals	2500
Type of selection	roulette-wheel
Type of mutation	point-mutation
Type of crossover	one-point (2 parents)
Type of replacement	elitist
Generation gap	0.9
Probability of crossover	0.5
Probability of mutation	0.5
Probability of changing terminal – non-terminal nodes (vica versa) during mutation	0.25

Since polynomial models play an important role in process engineering, in this toolbox there is an option of generating polynomial models. If this option is selected the set of operators is defined as $F = \{+, *\}$, and after every mutation and cross-over the GP algorithm validates the model structure whether is in the class of polynomial models. If it is necessary, the algorithm exchanges the internal nodes that are below a '$*$'-type internal node to '$*$'-type nodes. This simple trick transforms every tree into a well-ordered polynomial model.

The OLS evaluation is inserted into the *fitness evaluation* step. The OLS calculates the error reduction ratio of the branches of the tree. The terms that have error reduction ratio bellow a threshold value are eliminated from the tree. With the help of the selected branches the OLS estimates the parameters of the model represented by the reduced tree. After this reduction and parameter estimation step the new individual proceeds on its way in the classical GP algorithm (fitness evaluation, selection, etc.).

4.2 Example I: Nonlinear Input-Output Model

In the first example a simple nonlinear input-output model which is linear in its parameters is considered:

$$y(k) = 0.8u(k-1)^2 + 1.2y(k-1) - 0.9y(k-2) - 0.2, \tag{20}$$

where $u(k)$ and $y(k)$ are the input and the output variables of the model at the k-th sample time. The aim of the experiment is the identification of the model structure from measurements. The measurements was generated by simulation of the system and 4% relative normal distributed noise was added to the output (Fig. 3 shows input and output data).

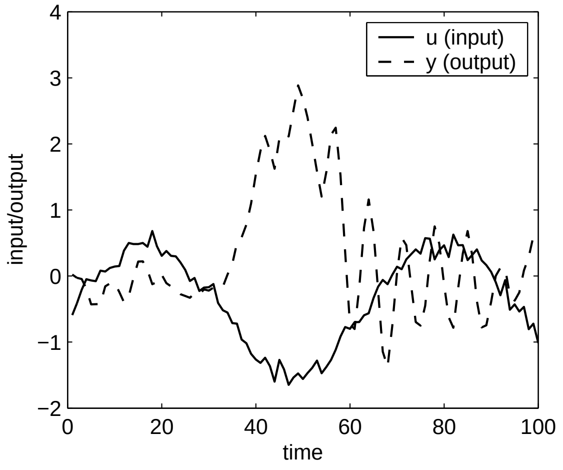

Figure 3: Input-output data for model structure identification (Example I.)

During the identification process the function set F contained the basic arithmetic operations $F = \{+, -, *, /\}$, and the terminal set T contained the following arguments $T = \{u(k-1), u(k-2), y(k-1), y(k-2)\}$. Based on the OLS method, the terms of every model terms were sorted by error reduction ratio values. In this application example maximum four terms were allowed, which means that the OLS procedure eliminated the worst terms and only the best four terms remained. (Certainly, this also means that if the original model does not contain more than four terms, this procedure will not change the model.) Three different approaches were compared:

- Method 1.: Classical GP (without penalty function and OLS).
- Method 2.: GP with penalty function and without OLS.

- Method 3.: GP with penalty function and OLS.

Because GP is a stochastic optimization algorithm ten independent runs were executed for each methods, while the maximum number of function evaluation in every run was constrained to 1000.

Table 3: Results of Example I. (average of ten independent runs)

	Method 1	Method 2	Method 3
Found perfect solution	0	6	7
Found non-perfect solution	10	4	3
Average number of function evaluation to found a proper solution	1000	880	565
Average runtime (sec)	33.3	33.3	39.5

Remark: average runtime: 1000 fun.evaluations, P4 2.4 GHz PC

As Table 3 shows, Method 3. proved the best, it was able to find the perfect model structure seven times, and found the perfect structure in the shortest time (averagely this method needed the smallest number of function evaluations to find the perfect solution). The average runtime values illustrates that the OLS technique slows down the algorithm (because it needs extra computations), but the improvement in the efficiency compensates this small disadvantage.

4.3 Example II: Continuous Polymerization Reactor

In the previous example the structure of the identified model was perfectly known. In this experiment the perfect model structure does not exist, but the correct model order is known. This experiment demonstrates that the proposed GP-OLS technique is a useful tool for model order selection.

This model order selection example is taken from [11]. The input-output dataset is generated by a simulation model of a continuous polymerization reactor. This model describes the free-radical polymerization of methyl-methacrylate with azo-bis(isobutyro-nitrile) as an initiator and toluene as a solvent. The reaction takes place in a jacketed CSTR. The first-principle model of this process is given in [28]:

$$\begin{aligned}
\dot{x}_1 &= 10(6 - x_1) - 2.4568 x_1 \sqrt{x_2} \\
\dot{x}_2 &= 80u - 10.1022 x_2 \\
\dot{x}_3 &= 0.024121 x_1 \sqrt{x_2} + 0.112191 x_2 - 10 x_3 \\
\dot{x}_4 &= 245.978 x_1 \sqrt{x_2} - 10 x_4
\end{aligned}$$

$$y = \frac{x_4}{x_3}. \tag{21}$$

The dimensionless state variable x_1 is the monomer concentration, and x_4/x_3 is the number-average molecular weight (the output y). The process input u is the dimensionless volumetric flowrate of the initiator. According to [11], a uniformly distributed random input over the range 0.007 to 0.015 is applied and the sampling time was 0.2 s.

With four states, a sufficient condition for representing the dynamics is a regression vector that includes four delayed inputs and outputs. In this case, however, the system has two states that are weakly observable. This week observability leads to a system that can be approximated by a smaller input–output description [29]. This is in agreement with the analysis of Rhodes [11] who showed that a nonlinear model with $m = 1$ and $n = 2$ orders is appropriate; in other words the model can be written in the following form:

$$y(k) = G\left(y(k-1), u(k-1), u(k-2)\right), \tag{22}$$

if the discrete sample time $T_0 = 0.2$.

To estimate the model structure, 960 data points were generated by computer simulation. In this example, we examined four methods:

- Method 1. generates all of the possible polynomials with degree $d = 2$. The model consists of all of these terms.
- Method 2. generates all of the possible polynomials with degree $d = 2$, but the model only consists of the terms which have greater error reductions ratios than 0.01.
- Method 3. is the polynomial GP-OLS technique. The operator set is $F = \{*, +\}$. The OLS threshold value is 0.02.
- Method 4.: Polynomial GP-OLS technique. The operator set is $F = \{*, +, /, \sqrt{}\}$. The OLS threshold value is 0.02.

Table 4 shows the mean square errors (MSE) of resulted models for one-step ahead and for free-run predictions. Since GP is a stochastic algorithm, 10 identical experiments were performed for the third and fourth method, and the table contains the minimum, the maximum and the mean of the results of these experiments. The input and output order of the models were limited to four: $u(k-1), \cdots, u(k-4), y(k-1), \cdots, y(k-4)$.

In the first method, the model consisted of 45 polynomial terms ($m = 8, d = 2$). This model was very accurate for one-step ahead prediction, but

Table 4: Results of Example II.

	Free-run MSE			One-step-ahead MSE		
	min	mean	max	min	mean	max
Method 1		Inf			7.86	
Method 2		26.8			30.3	
Method 3	1.65	10.2	23.7	1.66	9.23	22.1
Method 4	0.95	7.15	20.6	0.84	6.63	17.8

Remark: MSE in 10^{-3}

it was unstable in free-run prediction. Hence, this model can not be used in free-run simulation.

In the second method, the error reduction ratios were calculated for the 45 polynomial terms, and the terms which have very small error reduction values (below 0.01) were eliminated. After this reduction only three terms were remained:

$$u(k-1), y(k-1), y(k-2)\,;$$

all of the bilinear terms were eliminated by OLS. This model is a simple linear model, that is stable in free-run simulation, but its performance is quite week.

The third method resulted different models in the 10 experiments, due to its stochastic nature. All of resulted models were stable in free-run. The most accurate model contained the next terms:

$$u(k-1) * u(k-1), y(k-1), u(k-2), u(k-1) * y(k-1)\,;$$

which has correct model order (see (22)). This method found the correct model order in six cases from the ten.

The fourth method (GP-OLS) resulted correct model orders in three cases from the ten. This method found the most accurate model and all of resulted models were stable in free-run. Statistically, this method generated the most accurate models, but the third method was better at finding the correct model order.

4.4 Example III: Van der Vusse Reactor

The process considered in this section is a third order exothermic van der Vusse reaction

$$\begin{aligned} A \rightarrow B \rightarrow C \\ 2A \rightarrow D \end{aligned} \tag{23}$$

placed in a cooled Continuously Stirred Tank Reactor (CSTR). It is a strongly nonlinear process with a non-minimum-phase behavior and input multiplicity. The model of the system is given by

$$\begin{aligned}
\dot{x}_1 &= -x_1 k_1 e^{-\frac{E_1}{x_3}} - x_1^2 k_3 e^{-\frac{E_3}{x_3}} + (x_{10} - x_1)\, u_1 \qquad (24)\\
\dot{x}_2 &= x_1 k_1 e^{-\frac{E_1}{x_3}} - x_2 k_2 e^{-\frac{E_2}{x_3}} - x_2 u_1\\
\dot{x}_3 &= -\frac{1}{\varrho c_p}\left[\Delta H_1 x_1 k_1 e^{-\frac{E_1}{x_3}} + \Delta H_2 x_2 k_2 e^{-\frac{E_2}{x_3}}\right.\\
&+ \left.\Delta H_3 x_1^2 k_3 e^{-\frac{E_3}{x_3}}\right] + (x_{30} - x_3)\, u_1 + \frac{u_2}{\varrho c_p V}\\
y &= x_2,
\end{aligned}$$

where $x_1[mol/l]$ is the concentration of the A component, $x_2[mol/l]$ is the concentration of the B component, $x_3[K]$ is the temperature in the reactor, $u_1[1/s]$ is the dilution rate of the reactor, $u_2[J/s]$ is the heat exchanged between the CSTR and the environment, x_{10} is the concentration of A in the inlet stream and x_{30} is the temperature of the inlet stream. From the application point of view the u_1 input flow rate is chosen as the input of the system while u_2 is kept constant through the experiments [30].

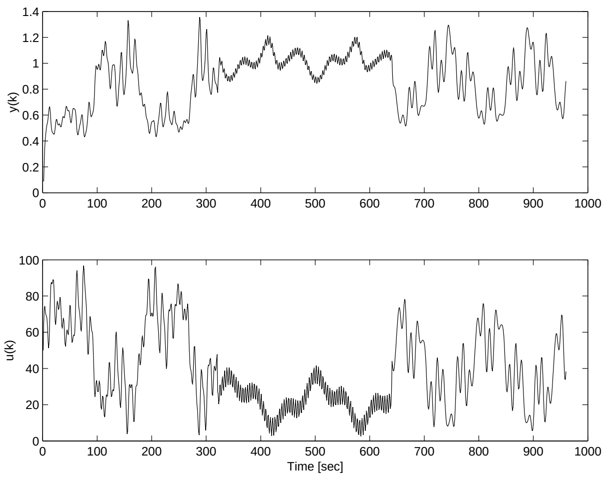

Figure 4: Collected input–output data of the van der Vusse reactor (Example III.)

To estimate the model structure, 960 data points were used (see Fig. 4). In this example, the same four methods were used as in the previous example.

Table 5: Results of Example III.

	Free-run MSE			One-step-ahead MSE		
	min	mean	max	min	mean	max
Method 1		Inf			0.002	
Method 2		20.3			0.31	
Method 3	5.30	9.65	Inf	0.088	0.24	0.62
Method 4	2.44	11.5	Inf	0.15	0.53	0.99

Remark: MSE in 10^{-3}

In the first method the model consisted of 45 polynomial terms ($m = 8, d = 2$). Similar to previous experience, this model was very accurate for one-step ahead prediction, but it was unstable in free-run prediction.

In the second method the error reduction ratios were calculated for the 45 polynomial terms, and the terms which have very small values (below 0.01) were eliminated. After that, seven terms remained:

$$y(k-2), y(k-3), u(k-1), u(k-2), y(k-4), u(k-3), y(k-1);$$

all of the bilinear terms were eliminated by OLS. This model is a linear model, it was stable in free-run, but it was not accurate.

Contrast to previous example, the third method results in free-run unstable models in five cases from the ten experiments. It is due to that the van der Vusse process has complex nonlinear behavior, so this example is more difficult than the previous. The model which had the smallest MSE value in one-step ahead prediction was unstable in free-run prediction, too. The best model (free-run MSE) consisted of the next terms:

$$u(k-2), u(k-1), u(k-1)u(k-1)u(k-4), y(k-4),$$

$$u(k-4)u(k-3), u(k-4) + u(k-3).$$

The simplest model consisted of the

$$u(k-1), u(k-2), u(k-1)u(k-1), u(k-3)y(k-1), y(k-4)$$

terms.

In the fourth method, three models were unstable from the ten. The most accurate model contained the following terms:

$$y(k-3), y(k-1)u(k-1), y(k-2)\sqrt{u(k-3)},$$

$$(y(k-3) + u(k-2))(y(k-4) + y(k-1)), y(k-2).$$

The simplest model contained the

$$y(k-1), y(k-2), u(k-1)/u(k-2), u(k-2), u(k-1)$$

terms.

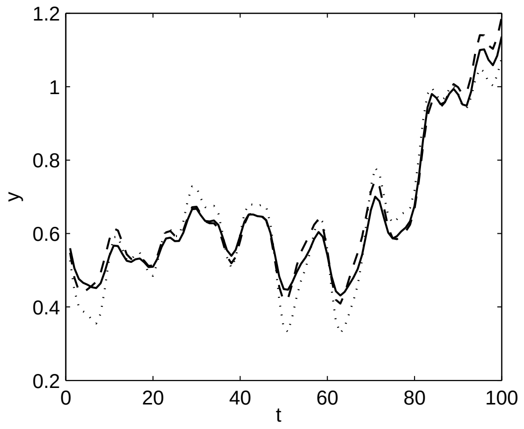

Figure 5: Free-run simulation of resulted models. Solid line: original output, Dotted line: estimated output by Method 2., Dashed line: estimated output by Method 4 (best model).

As these results show the model that gives the best free run modelling performance is given by the GP-OLS algorithm (see Fig. 5), and the structure of this model is quite reasonable comparing to the original state-space model of the system.

5 Conclusions

This paper proposed a new method for the structure identification of nonlinear input-output dynamical models. The method uses Genetic Programming (GP) to generate linear-in-parameters models represented by tree structures. The main idea of the paper is the application of Orthogonal Least Squares algorithm (OLS) for the estimation of the contributions of the branches of the tree to the accuracy of the model. Since the developed algorithm is able to incorporate information about the contributions of the model terms (subtrees) into the quality of the model into GP, the application of the proposed GP-OLS tool results in accurate, parsimonious and interpretable models. The proposed approach has been implemented as a freely available

MATLAB Toolbox. The simulation results show that this tool provides an efficient way for model order selection and structure identification of nonlinear input-output models.

List of captions (graphics)

References

Ljung87 1. Ljung, L. *System Identification, Theory for the User*; Prentice–Hall: New Jersey, 1987.

Akai74 2. Akaike, H. A new look at the statistical model identification. *IEEE Trans. Automatic Control* **1974**, *19*, 716–723.

Liang93 3. Liang, G.; Wilkes, D.; Cadzow, J. ARMA model order estimation based on the eigenvalues of the covariance matrix. *IEEE Trans. on Signal Processing* **1993**, *41(10)*, 3003–3009.

Aguirre95 4. Aguirre, L.A.; Billings, S.A. Improved Structure Selection for Nonlinear Models Based on Term Clustering. *International Journal of Control* **1995,** *62*, 569–587.

Aguirre96 5. Aguirre, L.A.; Mendes, E.M.A.M. Global Nonlinear Polynomial Models: Structure, Term Clusters and Fixed Points. *International Journal of Bifurcation and Chaos* **1996**, *6*, 279–294.

Mendes01 6. Mendes, E.M.A.M.; Billings, S.A. An alternative solution to the model structure selection problem. *IEEE Transactions on Systems Man and Cybernetics part A - Systems and Humans* **2001**, *31(6)*, 597–608.

Korenberg88 7. Korenberg, M.; Billings, S.A.; Liu, Y.; McIlroy, P. Orthogonal Parameter-Estimation Algorithm for Nonlinear Stochastic-Systems. *International Journal of Control* **1988**, *48*, 193–210.

Abonyi03 8. Abonyi, J. *Fuzzy Model Identification for Control*; Birkhauser: Boston, 2003.

Pearson03 9. Pearson, R. Selecting nonlinear model structures for computer control. *Journal of Process Control* **2003**, *13(1)*, 1–26.

Bomberger98 10. Bomberger, J.; Seborg, D. Determination of model order for NARX models directly from input–output data. *Journal of Process Control* **1998**, *8*, 459–468.

Rhodes98 11. Rhodes, C.; Morari, M. Determining the model order of nonlinear input/output systems. *AIChE Journal* **1998**, *44*, 151–163.

Kennen92 12. Kennen, M.; Brown, R.; Abarbanel, H. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review* **1992**, *A*, 3403–3411.

Abonyi04 13. Feil, B.; Abonyi, J.; Szeifert, F. Model order selection of nonlinear inputoutput models a clustering based approach. *Journal of Process Control* **2004**, *14(6)*, 593–602.

Koza92 14. Koza, J. *Genetic Programming: On the programming of Computers by Means of Natural Evolution*; MIT Press: Cambridge, 1992.

Cao99 15. Cao, H.; Yu, J.; Kang, L.; Chen, Y. The kinetic evolutionary modeling of complex systems of chemical reactions. *Computers and Chem. Eng.* **1999**, *23*, 143–151.

McKay97 16. McKay, B.; Willis, M.; Barton, G. Steady-state modelling of chemical process systems using genetic programming. *Computers and Chem. Eng.* **1997**, *21*, 981–996.

Sakamoto01 17. Sakamoto, E.; Iba, H. Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*; IEEE Press: COEX, World Trade Center, **2001**, 720–726.

Sjoberg95 18. Sjoberg, J.; Zhang, Q.; Ljung, L.; Benvebiste, A.; Delyon, B.; Glornnec, P.; Hjalmarsson, H.; Judutsky, A. On the use of regularization in system identification. *Automatica* **1995**, *31*, 1691–1724.

Pearson97 19. Pearson, R.; Ogunnaike, B. Nonlinear Process Identification. Chapter 2 in *Nonlinear Process Control*; Henson, M.A.; Seborg, D.E., Eds.; Prentice–Hall: Englewood Cliffs, NJ, 1997.

Hernandez93 20. Hernandez, E.; Arkun, Y. Control of Nonlinear Systems Using Polynomial ARMA Models. *AICHE Journal* **1993**, *39(3)*, 446–460.

Ferreira01 21. Ferreira, C. Gene expression programming: a new adaptive algorithm for solving problems. *Complex Syst* **2001**, *13*, 87–129.

Sette01 22. Sette, S.; Boullart, L. Genetic Programming: principles and applicaionts. *Engineering Applications of Artificial Intelligence* **2001**, *14*, 727–736.

Koza94II 23. Koza, J. *Genetic programming II: automatic discovery of reusable programs*; MIT Press, 1994.

South94 24. South, M. *The application of genetic algorithms to rule finding in data* *analysis, PhD. thesis*; Dept. of Chemical and Process Eng.: The Univeristy of Newcastle upon Tyne, UK, 1994.

Billings88 25. Billings, S.; Korenberg, M.; Chen, S. Identification of nonlinear output-affine systems using an orthogonal least-squares algorithm *International Journal of Systems Science* **1988**, *19*, 1559–1568.

Chen89 26. Chen, S.; Billings, S.; Luo, W. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control* **1989**, *50*, 1873–1896.

Matlab02 27. *MATLAB Optimization Toolbox;* MathWorks Inc.: Natick, MA, 2002.

Doyle95 28. Doyle, F.; Ogunnaike, B.; Pearson, R. K. Nonlinear model-based control using second-order volterra models. *Automatica* **1995**, *31(5)*, 697–714.

Letellier02 29. Letellier, C.; Aguirre, L. Investigating Nonlinear Dynamics from Time Series: The Influence of Symmetries and the Choice of Observables. *Chaos* **2002**, *12*, 549–558.

Braake99 30. Braake, H.A.B.; Roubos; J.A. Babuska, R. Semi-mechanistic modeling and its application to biochemical processes. In *Fuzzy Control: Advances in Applications*; Verbruggen, H.B.; Babuska, R. eds.; World Scientific: Singapore 1999, pp. 205–226