Two-Stage Orthogonal Least Squares Methods for Neural Network Construction

Long Zhang, *Member, IEEE*, Kang Li, *Senior Member, IEEE*, Er-Wei Bai, *Fellow, IEEE*, and George W. Irwin, *Fellow, IEEE*

***Abstract*— A number of neural networks can be formulated as the linear-in-the-parameters models. Training such networks can be transformed to a model selection problem where a compact model is selected from all the candidates using subset selection algorithms. Forward selection methods are popular fast subset selection approaches. However, they may only produce suboptimal models and can be trapped into a local minimum. More recently, a two-stage fast recursive algorithm (TSFRA) combining forward selection and backward model refinement has been proposed to improve the compactness and generalization performance of the model. This paper proposes unified two-stage orthogonal least squares methods instead of the fast recursive-based methods. In contrast to the TSFRA, this paper derives a new simplified relationship between the forward and the backward stages to avoid repetitive computations using the inherent orthogonal properties of the least squares methods. Furthermore, a new term exchanging scheme for backward model refinement is introduced to reduce computational demand. Finally, given the error reduction ratio criterion, effective and efficient forward and backward subset selection procedures are proposed. Extensive examples are presented to demonstrate the improved model compactness constructed by the proposed technique in comparison with some popular methods.**

***Index Terms*— Backward model refinement, computational complexity, forward selection, linear-in-the-parameters model, orthogonal least square (OLS).**

I. Introduction

THIS paper considers the construction of a wide range of neural networks such as polynomial nonlinear autoregressive models with exogenous inputs (NARX) networks [1], [2], radial basis function (RBF) neural networks [3], [4], neurofuzzy models [5], and wavelet networks [6], [7], which have been extensively investigated in the literature and used in system identification, discriminant, and classification [8]–[12]. These neural models have a standard structure consisting of one hidden layer and one output layer with linear output weights. In other words, such a model structure is a linear combination of nonlinear functions, such as polynomials, Gaussian functions, and wavelets. The construction of these nonlinear models involves the determination of tunable nonlinear parameters in the hidden nodes, linear output weights, as well as the model size. This is often achieved by minimizing a cost function, usually formulated by the mean sum squared error (SSE) between the measured and the estimated outputs or statistical information based criteria like Akaike information criterion (AIC) [13], Bayesian information criterion (BIC) [14], and final prediction error (FPE) [15]. It has been intensively researched that the above single hidden layer neural network can be transformed into the linear-in-the-parameters models if the tunable nonlinear parameters in the hidden nodes have fixed values [16]. Building a linear-in-the-parameters model is to select some representative nonlinear function regressors from a large candidate pool, which is referred to as a model selection problem.

A variety of algorithms for building the linear-in-the-parameters models have been proposed. The hybrid method combining unsupervised and supervised learning introduced in [17] has been widely studied, where the clustering method is used to determine the nonlinear parameters in the hidden layer nodes, and then the least squares algorithm is used to update their linear output weights. The drawback is that the unsupervised learning does not fully use the information contained in the training data, failing to exploit the full potential of a single hidden layer neural network [18]. Another simple yet very efficient approach is the extreme learning machine (ELM) that randomly assigns nonlinear parameters for each hidden node and then carries out a least squares estimation of their output weights [19]. However, a sparse model with a satisfactory generalization performance may not be obtainable since some unimportant model terms make little contributions to the model performance due to its stochastic procedure in assigning values for the nonlinear parameters in the hidden nodes.

Subset selection algorithms have been widely used for evaluating the suitability of a subset from the candidate models and they can determine the model size, select nonlinear parameters, and estimate the output weights simultaneously [20]. The most effective and efficient subset algorithms mainly include forward and backward selections. The forward selection begins with an empty model with no terms in it and then gradually builds a model by adding one term from the candidate pool

Manuscript received November 25, 2013; revised April 22, 2014; accepted July 27, 2014. Date of publication September 11, 2014; date of current version July 15, 2015. This work was supported in part by the U.K. Research Councils under Grant EP/G042594/1 and EP/L001063/1, in part by the Chinese Scholarship Council, in part by the National Natural Science Foundation of China under Grant 51077022, Grant 61271347, and Grant 61273040, and in part by the Science and Technology Commission of Shanghai Municipality under Grant 11ZR1413100.

L. Zhang, K. Li, and G. W. Irwin are with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT7 1NN, U.K. (e-mail: lzhang14@qub.ac.uk; k.li@qub.ac.uk; g.irwin@qub.ac.uk).

E.-W. Bai is with the Department of Electrical and Computer Engineering, University of Iowa, Iowa, IA 52242 USA, and also with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT9 5AH, U.K. (e-mail: er-wei-bai@uiowa.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2346399

that gives the largest decrease in the cost function at a time until some model stopping criterion is satisfied [21]. Typical forward selection methods include forward orthogonal least squares (OLS) method [22], [23], fast recursive algorithm (FRA) [24], matching pursuits (MP) [25], orthogonal matching pursuits (OMP) [26], and optimized OMP (OOMP) [27]. These approaches share a similar forward selection procedure from the numerical operation point of view. However, they use different criteria in determining which model term will be included in the resultant model and in finding the optimal model coefficients of the selected terms. Both the forward OLS and FRA select one model term that maximally reduces the SSE at each step, and the error reduction ratio (ERR) is often used as the stopping criterion. They determine the model coefficients using the least squares estimate. For MP and OMP, they choose the model term that has the largest correlation with the current modeling residue. MP uses this correlation as the coefficient of the selected term, while OMP updates model coefficients at each step using least squares estimate. As a variant of OMP, OOMP employs the ERR as the model selection criterion. Hence, OOMP shares the same procedure with the forward OLS. The only difference is that the OOMP does not explicitly use the least square method, thus saves more computation time than the forward OLS.

The backward selection starts with all candidate terms being included and then deletes one term at a time, which is the least significant in terms of the cost function reduction [20], [28]. The forward selection tends to be more popular than the backward selection since it is more robust and efficient with respect to numerical operations. However, both the forward and backward selections may not be optimal [29]. To improve the model compactness and generalization performance, various improved methods have been proposed.

To simultaneously improve the model approximation capability and generalization performance, composite cost functions have been proposed for the forward OLS selection. For example, A-optimality [30] and D-optimality [31] experimental designs are used to suppress high variances in the parameter estimates caused by random sampling errors [32]. The drawback is that these experimental design criteria lack a sound theoretical coverage for the identification of nonlinear systems [33].

Alternatively, regularization methods penalize large model coefficients and shrink them toward zero to improve the model generalization performance. They are preferable when data are sampled under severely noisy conditions or for illconditioned regression problems [34], [35]. Typical regularization methods include the ridge regression and least absolute shrinkage and selection operator (LASSO), which use l_2 and l_1 norm penalties, respectively. It has been shown that the LASSO enjoys some favorable properties of both the ridge regression and the subset selection, and it can be interpreted as a Bayesian estimator [36]. However, it is difficult to mathematically formulate any restrictions on the l_1 penalty parameters and there is no analytical solutions due to the nondifferentiability of the l_1 penalty function. As LASSO is a convex optimization problem, a number of algorithms have been proposed, such as block coordinate relaxation method [37], gradient descent [38], iterated ridge regression [39], and piecewise linear regularized solution [40]. More recently, a promising scheme of regularization methods—the least angle regression (LAR)—has become a hot research topic [41]. It has been shown that LAR has some distinctive merits. First, it is computationally as fast as the forward selection and more efficient than LASSO since it has a full piecewise linear solution path. Furthermore, it is easily modified to produce solutions for LASSO estimator. However, the LAR could not guarantee to build a sparser model than the forward selection and LASSO methods.

Unlike the regularization methods where the penalty terms are incorporated in the cost function, another two popular strategies are available for reevaluating the initial results produced by forward selection. The first combined the forward OLS selection with evolutionary methods [42]–[44]. Though evolutionary methods are global optimization algorithms, they often suffer from slow and premature convergence due to their stochastic sampling nature [16]. The alternative is the two-stage FRA (TSFRA), which first selects an initial model using forward selection and then reviews the significance of each term in it, replacing the insignificant ones, thereby improving model compactness and accuracy significantly. In other words, a term that is entered into the model using forward selection can become unimportant when other terms are later included. Contrary to the forward selection, TSFRA adds a backward refinement stage to discard unimportant terms and select more significant ones [24]. However, the TSFRA cannot guarantee global optimality.

Among all the subset selection methods, the forward OLS selection is perhaps one of the most popular approaches in the literature. However, there is no unified OLS-based framework to combine both forward and backward subset selections. Based on the concept of the TSFRA reported in [16], this paper introduces two-stage OLS (TSOLS) methods for constructing compact linear-in-the-parameters models. The main difference of the proposed methods with the TSFRA is that the proposed TSOLS further reduces the computational cost. Compared with the forward OLS, LAR, OOMP, and four LASSO methods, the main advantage of the TSOLS method is that it could build a more compact model with smaller model size. More specifically, the main contributions of this paper can be summarized as follows.

1) Unlike the previous proposed TSFRA, the TSOLS methods are computationally more efficient. This is achieved by establishing the simplified relations between the regression contexts of the first stage model selection and the second stage model refinement. Furthermore, unlike the scheme used in the TSFRA where two adjacent terms are interchanged sequentially in the second stage to improve the computational efficiency, the new method directly shifts the previously selected term of interest to the last position in the selected model term pool to further save computational efforts.
2) For the implementation of TSOLS methods, two types of orthogonal decomposition methods, namely the classical Gram–Schmidt (CGS) and the modified Gram–Schmidt (MGS) methods are introduced and their relations are clarified.

3) A computational complexity analysis confirms that the TSOLS is computationally more efficient than the TSFRA.
4) The effectiveness of the new method is demonstrated by constructing three widely used single hidden layer neural network, namely polynomial NARX model, RBF networks, and wavelet networks.
5) A thorough comparison with eight popular algorithms, including forward OLS selection, LAR, OOMP, TSFRA, and four LASSO methods, are given to show that the proposed two-stage method is able to construct a more compact model with a fewer model terms while they all have the similar generalization performance.

II. Preliminaries

A. Linear-in-the-Parameters Models

A linear-in-the-parameters structure for polynomial NARX models, RBF neural networks, and wavelet networks can be formulated as a linear combination of nonlinear functions, which is given by [3], [6], [16]

$$y(t) = \sum_{i=1}^{M} p_i(\mathbf{x}(t), \mathbf{v}_i)\theta_i + \xi(t) \tag{1}$$

where $t = 1, 2, \ldots, N$, N being the size of the estimation data set. $\{\mathbf{x}(t), y(t)\}$ are the model input and output variables at time instant t. The set $\{p_i, i = 1, \ldots, M\}$ represents all the candidate nonlinear function terms with fixed values for nonlinear parameters $\{\mathbf{v}_i, i = 1, \ldots, M\}$ and $\xi(t)$ is a zero-mean model residual sequence. The set $\{\theta_i, i = 1, \ldots, M\}$ is the model output weights, and M is the number of regressor terms.

It should be noted that, generally speaking, in nonlinear system identification, the model input is often not the same as the real system input, instead the model inputs often include the past system inputs, system outputs, and also possible modeling errors. This type of model is also referred to as nonlinear autoregressive (NAR) moving average with exogenous input model. For a classification problem, the input vector of a classifier generally equals the pattern vector, and its training process has no major differences from nonlinear system modeling. However, the classification results are different as the outputs of the classifier have to take discrete values representing the estimated classes.

Now, given a set of N training samples, (1) can be expressed in the matrix form as

$$\mathbf{y} = \mathbf{P}\Theta + \Xi \tag{2}$$

where $\mathbf{y} = [y(1), \ldots, y(N)]^T$ is the output vector, $\Theta = [\theta_1, \ldots, \theta_M]^T$ is the unknown parameter vector, $\Xi = [\xi(1), \ldots, \xi(N)]^T$ is the residual vector, and $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_M]$ is a N-by-M matrix with $\mathbf{p}_i = [p_i(\mathbf{x}(1), \mathbf{v}_i), \ldots, p_i(\mathbf{x}(N), \mathbf{v}_i)]^T$.

For these linear-in-the-parameters models, the nonlinear parameters $\mathbf{v}_i$ have to be predetermined. For polynomial NARX models, no nonlinear parameters exist, and only different lags for the variables have to be determined [22]. The nonlinear parameters in the RBF networks are centers and widths. The centers $\mathbf{s}_i$ can be initiated using input data samples while the widths are generally set with a fixed value [3]. For the wavelet networks, the nonlinear parameters, namely dilations and translations, are often initialized using discrete integers with a narrow range as a wavelet is compactly supported in the narrow range and shrinks to zero [45]. Alternatively, the ELM method can be applied to assign random values for these nonlinear parameters [19].

B. Orthogonal Least Square

The forward OLS selection aims to select a representative subset from the whole candidate model $\mathbf{P}$ shown in (2), which involves a series of orthogonal decompositions. Each decomposition, the Mth say, is given by [22]

$$\mathbf{P} = \mathbf{WA} \tag{3}$$

where $\mathbf{W}$ is an N-by-M matrix with orthogonal columns $\mathbf{w}_i$, which satisfies

$$\mathbf{W}^T\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T\mathbf{w}_1 & 0 & \ldots & 0 \\ 0 & \mathbf{w}_2^T\mathbf{w}_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \mathbf{w}_M^T\mathbf{w}_M \end{bmatrix} \tag{4}$$

and $\mathbf{A}$ is an $M \times M$ triangular matrix

$$\mathbf{A} = \begin{bmatrix} 1 & \alpha_{12} & \ldots & \alpha_{1M} \\ 0 & 1 & \ldots & \alpha_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1. \end{bmatrix}. \tag{5}$$

Equation (2) can thus be expressed as

$$\mathbf{y} = \mathbf{WA}\Theta + \Xi = \mathbf{Wg} + \Xi \tag{6}$$

where $\mathbf{g} = [g_1, g_2, \ldots, g_M]^T = \mathbf{A}\Theta$ is the orthogonal weight vector, which can then be calculated by [22]

$$\mathbf{g} = (\mathbf{W^T W})^{-1}\mathbf{W}^T\mathbf{y}. \tag{7}$$

The modeling error is

$$\Xi = \mathbf{y} - \mathbf{Wg} = \mathbf{y} - \mathbf{W}(\mathbf{W^T W})^{-1}\mathbf{W}^T\mathbf{y}. \tag{8}$$

The model parameters Θ can be computed using backward substitution as follows:

$$\left.\begin{aligned} \theta_M &= g_M \\ \theta_i &= g_i - \sum_{k=i+1}^{M} \alpha_{ik}\theta_k, i = M-1, \ldots, 1 \end{aligned}\right\}. \tag{9}$$

The orthogonal decomposition involved in OLS can be carried out by the CGS method. This computes one column of $\mathbf{A}$ at a time and factorizes $\mathbf{P}$ as follows:

$$\left.\begin{aligned} \mathbf{w}_1 &= \mathbf{p}_1 \\ \alpha_{ik} &= \frac{\langle \mathbf{w}_i, \mathbf{p}_k \rangle}{\langle \mathbf{w}_i, \mathbf{w}_i \rangle}, \ i = 1, \ldots, k-1 \\ \mathbf{w}_k &= \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik}\mathbf{w}_i \end{aligned}\right\} k = 2, \ldots, M \tag{10}$$

where the $\alpha_{ik}\mathbf{w}_i$ represents the projection of term $\mathbf{p}_k$ into the previously obtained orthogonal basis vector $\mathbf{w}_i$. Then, the weight vector $\mathbf{g}$ is calculated by

$$g_i = \frac{\langle \mathbf{w}_i, \mathbf{y} \rangle}{\langle \mathbf{w}_i, \mathbf{w}_i \rangle}, \quad i = 1, \ldots, M. \tag{11}$$

III. New Two-Stage CGS-Based OLS Method

The conventional TSFRA was introduced in [16], which selects an initial model using forward selection in the first stage and subsequently replaces insignificant terms in the second stage. More specifically, at the first stage, an initial model $\mathbf{P}_{M_s} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{M_s}\}$ is selected from the full set $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_M\}$ by choosing a single term each time, which maximally reduces the model error. Here, M_s indicates the number of initial model terms. In other words, the procedure selects the most significant term as $\mathbf{p}_1$ and then finds the second most significant term $\mathbf{p}_2$, repeating this until M_s terms have been found. At the second stage, each previously selected but the last term in the initial model is reviewed and compared with the left terms. The second stage can be divided into two main parts. First, a previously selected term $\mathbf{p}_n$, $n = M_s - 1, \ldots, 1$, is shifted to the M_sth position as if it were the last selected one by the first stage. The initial model

$$\mathbf{P}_{M_s} = [\mathbf{p}_1 \ \ldots \ \mathbf{p}_{n-1} \ \mathbf{p}_n \ \mathbf{p}_{n+1} \ \ldots \ \mathbf{p}_{M_s}] \tag{12}$$

thus becomes

$$\mathbf{P}_{M_s}{}' = [\mathbf{p}_1 \ \ldots \ \mathbf{p}_{n-1} \ \mathbf{p}_{n+1} \ \ldots \ \mathbf{p}_{M_s} \ \mathbf{p}_n]. \tag{13}$$

Then, apart from the last term $\mathbf{p}_n$, the former $(M_s - 1)$ terms in (13) combined with each unused term from the remaining candidate pool

$$\mathbf{P}_{\text{left}} = [\mathbf{p}_{M_s+1} \ \ldots \ \mathbf{p}_M] \tag{14}$$

construct $(M - M_s)$ new candidate models. If the training error of the initial model is larger than some of the new candidate models, it is replaced by one which reduces this error most, leading to an improved model without increasing the model size. In other words, the contribution of the $\mathbf{p}_n$ is less than some unused term and is replaced by the best one to produce an improved model with reduced error. The second stage stops when no insignificant term can be replaced.

This paper first proposes a new TSOLS algorithm based on the Gram–Schmidt orthogonal decomposition. The main difference of the proposed method with the TSFRA is that the OLS instead of FRA is employed to implement the forward and backward procedures. The advantage of the new method is that it is computationally more efficient than the TSFRA. This is achieved by simplifying the relationship between the first and second stages using the orthogonal properties within the regression context. Furthermore, it is also found that, instead of using a series of position interchanges between the two adjacent terms to shift a previously selected model term to the last position, shifting the model term of interest directly to the last position can further save computational effort under the OLS scheme. For example, if five model terms have been selected in the first stage, the initial model is given by

$$\mathbf{P}_5 = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5]. \tag{15}$$

Suppose the significance of the $\mathbf{p}_2$ is reviewed in the second stage, it needs to be moved to the last position. The TSFRA carries out the following adjacent exchanges:

$$\begin{aligned}
\mathbf{P}_5 &= [\mathbf{p}_1 \quad \underline{\mathbf{p}_2} \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5] \\
\mathbf{P}_5' &= [\mathbf{p}_1 \quad \mathbf{p}_3 \quad \underline{\mathbf{p}_2} \quad \mathbf{p}_4 \quad \mathbf{p}_5] \\
\mathbf{P}_5'' &= [\mathbf{p}_1 \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \underline{\mathbf{p}_2} \quad \mathbf{p}_5] \\
\mathbf{P}_5''' &= [\mathbf{p}_1 \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5 \quad \underline{\mathbf{p}_2}]
\end{aligned} \tag{16}$$

while the TSOLS will directly move the $\mathbf{p}_2$ to the last position

$$\begin{aligned}
\mathbf{P}_5 &= [\mathbf{p}_1 \quad \underline{\mathbf{p}_2} \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5] \\
\mathbf{P}_5' &= [\mathbf{p}_1 \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5 \quad \underline{\mathbf{p}_2}].
\end{aligned} \tag{17}$$

The TSFRA has developed a regression context, which can effectively exchange two adjacent regressors without introducing large extra computation effort. Thus, an efficient algorithm has been developed in TSFRA to shift a previously selected regressor to the final position using a series of adjacent term exchanges, as indicated in (16). However, the regression context specific to the TSFRA cannot explicitly yield an efficient algorithm to employ (17). The proposed method has thus overcome this difficulty, with the aid of the OLS method, thus is computationally more efficient.

The main challenge for the two-stage approach is that additional computations are introduced in the second stage. This computation effort can be reduced by reusing some intermediate variables produced in the first stage and avoiding repetitive computations. To achieve these purposes, this section investigates the relation between the first and the second stages. Given the initial model $\mathbf{P}_{M_s}$ shown in (12) is built by the first stage, the significance of each term will then be reviewed at the second stage as the significance of each previously selected model terms will be reduced or affected by the later introduced model terms due to their correlations. Suppose $\mathbf{p}_n, n = M_s - 1, \ldots, 1$, is to be reviewed and therefore it needs to be moved to the last position, the orthogonal decomposition of the initial model $\mathbf{P}_{M_s} = \mathbf{W}_{M_s}\mathbf{A}_{M_s}$ then becomes $\mathbf{P}'_{M_s} = \mathbf{W}'_{M_s}\mathbf{A}'_{M_s}$. The relation between $\mathbf{W}_{M_s}, \mathbf{A}_{M_s}$ and $\mathbf{W}'_{M_s}, \mathbf{A}'_{M_s}$ is now established as follows. At the first stage, the Gram–Schmidt decomposition procedure mainly computes the upper triangular matrix

$$\mathbf{A}_{M_s} = \begin{bmatrix}
1 & \ldots & \alpha_{1(n-1)} & \alpha_{1n} & \ldots & \alpha_{1M_s} \\
 & \ddots & \vdots & \vdots & \ldots & \vdots \\
 & & 1 & \alpha_{(n-1)n} & \ldots & \alpha_{(n-1)M_s} \\
 & & & 1 & \ldots & \alpha_{nM_s} \\
 & & & & \ddots & \vdots \\
0 & & & & & 1
\end{bmatrix} \tag{18}$$

and the orthogonal basis matrix

$$\mathbf{W}_{M_s} = \left[\mathbf{w}_1 \ \ldots \ \mathbf{w}_{n-1} \ \mathbf{w}_n \ \ldots \ \mathbf{w}_{M_s}\right]. \tag{19}$$

During the second stage, if $\mathbf{p}_n$ is to be moved to the last position, this leads to the columns $\mathbf{p}_{n+1}, \mathbf{p}_{n+2}, \ldots, \mathbf{p}_{M_s}$ being moved to the left by one place. The remaining columns in $\mathbf{P}$ are kept unchanged while only $\mathbf{p}_{n+1}, \ldots, \mathbf{p}_{M_s}$ and their

associated orthogonal decompositions will alter. Thus, $\mathbf{A}_{M_s}$ becomes

$$\mathbf{A}'_{M_s}=\begin{bmatrix}1 & \dots & \alpha_{1(n-1)} & \alpha'_{1n} & \dots & \alpha'_{1M_s}\\ & \ddots & \vdots & \vdots & \dots & \vdots\\ & & 1 & \alpha'_{(n-1)n} & \dots & \alpha'_{(n-1)M_s}\\ & & & 1 & \dots & \alpha'_{nM_s}\\ & & & & \ddots & \vdots\\ 0 & & & & & 1\end{bmatrix} \tag{20}$$

and $\mathbf{W}_{M_s}$ changes to

$$\mathbf{W}'_{M_s}=\begin{bmatrix}\mathbf{w}_1 & \dots & \mathbf{w}_{n-1} & \mathbf{w}_n{}' & \dots \mathbf{w}_{M_s}{}'\end{bmatrix}. \tag{21}$$

Here, the elements in these matrices with $'$ are modified, and the others remain unchanged. This paper will also show that the elements in (20) and (21) can be updated using the previously generated values without explicitly solving orthogonal decomposition. This involves the following steps.

1) The $\mathbf{p}'_n$ term and its orthogonal decomposition are updated by the CGS method using (10), which is given by

$$\left.\begin{aligned}&\mathbf{p}'_n=\mathbf{p}_{n+1}\\ &\alpha'_{in}=\frac{\langle\mathbf{w}_i,\mathbf{p}'_n\rangle}{\langle\mathbf{w}_i,\mathbf{w}_i\rangle},\ i=1,\dots,n-1\\ &\mathbf{w}'_n=\mathbf{p}'_n-\sum_{i=1}^{n-1}\alpha'_{in}\mathbf{w}_i.\end{aligned}\right\}. \tag{22}$$

Since $\mathbf{p}'_n=\mathbf{p}_{n+1}$, the projection coefficients α'_{in} can be also expressed as

$$\alpha'_{in}=\frac{\langle\mathbf{w}_i,\mathbf{p}_{n+1}\rangle}{\langle\mathbf{w}_i,\mathbf{w}_i\rangle}=\alpha_{i(n+1)},\quad i=1,\dots,n-1. \tag{23}$$

The inner products $\langle\mathbf{w}_i,\mathbf{p}_{n+1}\rangle$ and $\langle\mathbf{w}_i,\mathbf{w}_i\rangle$ obtained in the first stage and are used to compute $\alpha_{i(n+1)}$, therefore it is unnecessary to compute them again. By reusing these inner products, (22) can be transformed to

$$\left.\begin{aligned}&\mathbf{p}'_n=\mathbf{p}_{n+1}\\ &\alpha'_{in}=\alpha_{i(n+1)},\quad i=1,\dots,n-1\\ &\mathbf{w}'_n=\mathbf{p}'_n-\sum_{i=1}^{n-1}\alpha'_{in}\mathbf{w}_i\\ &\quad=\mathbf{p}_{n+1}-\sum_{i=1}^{n-1}\alpha_{i(n+1)}\mathbf{w}_i.\end{aligned}\right\}. \tag{24}$$

To verify that (24) is correct, the orthogonal property that the new orthogonal basis $\mathbf{w}'_n$ is orthogonal to all the previous basis

$$\langle\mathbf{w}'_n,\mathbf{w}_j\rangle=0,\quad j=1,\dots,n-1 \tag{25}$$

is proved in the following.

For all the previously determined orthogonal basis, they satisfy

$$\langle\mathbf{w}_j,\mathbf{w}_i\rangle\begin{cases}=0, & i\neq j\\ \neq 0, & i=j\end{cases} \tag{26}$$

where $j=1,\dots,n-1, i=1,\dots,n-1$. Using (26), the proof of (25) can be given as follows:

$$\begin{aligned}\langle\mathbf{w}'_n,\mathbf{w}_j\rangle&=\langle\mathbf{p}_{n+1}-\sum_{i=1}^{n-1}\alpha_{i(n+1)}\mathbf{w}_i,\mathbf{w}_j\rangle\\ &=\langle\mathbf{p}_{n+1},\mathbf{w}_j\rangle-\sum_{i=1}^{n-1}\alpha_{i(n+1)}\langle\mathbf{w}_i,\mathbf{w}_j\rangle\\ &=\langle\mathbf{p}_{n+1},\mathbf{w}_j\rangle-\alpha_{j(n+1)}\langle\mathbf{w}_j,\mathbf{w}_j\rangle\\ &=\langle\mathbf{p}_{n+1},\mathbf{w}_j\rangle-\frac{\langle\mathbf{w}_j,\mathbf{p}_{n+1}\rangle}{\langle\mathbf{w}_j,\mathbf{w}_j\rangle}\langle\mathbf{w}_j,\mathbf{w}_j\rangle\\ &=\langle\mathbf{p}_{n+1},\mathbf{w}_j\rangle-\langle\mathbf{w}_j,\mathbf{p}_{n+1}\rangle\\ &=0.\end{aligned} \tag{27}$$

2) The scheme to reuse the inner products for $\mathbf{p}'_n$ is also applicable for updating terms $\mathbf{p}'_j, j=n+1,\dots,M_s-1$ and their orthogonal decompositions, which is given by

$$\left.\begin{aligned}&\mathbf{p}'_j=\mathbf{p}_{j+1}\\ &\alpha'_{ij}=\alpha_{i(j+1)},\quad i=1,\dots,n-1\\ &\alpha'_{ij}=\frac{\langle\mathbf{w}'_i,\mathbf{p}_{j+1}\rangle}{\langle\mathbf{w}'_i,\mathbf{w}'_i\rangle},\quad i=n,\dots,j-1\\ &\mathbf{w}'_j=\mathbf{p}'_j-\sum_{i=1}^{j-1}\alpha'_{ij}\mathbf{w}_i\\ &\quad=\mathbf{p}_{j+1}-\sum_{i=1}^{n-1}\alpha_{i(j+1)}\mathbf{w}_i-\sum_{i=n}^{j-1}\alpha'_{ij}\mathbf{w}'_i\end{aligned}\right\}. \tag{28}$$

Here, the orthogonal basis and the upper triangular matrices are updated one column at a time. As in updating $\mathbf{p}'_n$, the projections $\alpha'_{ij}, i=1,\dots,n-1$, for $\mathbf{w}_1,\dots,\mathbf{w}_{n-1}$ are identical to $\alpha_{i(j+1)}$ in the first stage, while the projections $\alpha'_{ij}, i=n,\dots,j-1$, for $\mathbf{w}'_n,\dots,\mathbf{w}'_j$ should be updated to correspond to the new orthogonal basis.

3) The last column $\mathbf{p}'_{M_s}$ and its orthogonal decomposition are updated using

$$\left.\begin{aligned}&\mathbf{p}'_{M_s}=\mathbf{p}_n\\ &\alpha'_{iM_s}=\alpha_{in},\quad i=1,\dots,n-1\\ &\alpha'_{iM_s}=\frac{\langle\mathbf{w}'_i,\mathbf{p}_n\rangle}{\langle\mathbf{w}'_i,\mathbf{w}'_i\rangle},\quad i=n,\dots,M_s-1\\ &\mathbf{w}'_{M_s}=\mathbf{p}'_{M_s}-\sum_{i=1}^{Ms-1}\alpha'_{iM_s}\mathbf{w}_i\\ &\quad=\mathbf{p}_n-\sum_{i=1}^{n-1}\alpha_{in}\mathbf{w}_i-\sum_{i=n}^{M_s-1}\alpha'_{iM_s}\mathbf{w}'_i\end{aligned}\right\}. \tag{29}$$

After the column $\mathbf{p}_n$ has been moved to the position of the last selected column $\mathbf{p}_{M_s}$, it is compared with the remaining terms in the pool. If its net contribution to the reduction of the cost function is less than any of the other remaining terms in the candidate pool, it is replaced by the one which produces the maximal reduction to the cost function. If it is not replaced, it can stay at the last position and does not need change back to its

original position according to the least squares theory, any change in the order of selecting the regressor terms, $\mathbf{p}_1, \ldots, \mathbf{p}_k$ does not alter the overall residual [16]. Based on the relation between the first and the second stages, the procedure of the two-stage subset selection will be given in the following section.

IV. Two-Stage Subset Selection

The subset selection involves the determination of model terms, model coefficients, and model size. In this paper, the AIC is used as a cost function for controlling the model size, which is defined as

$$\mathrm{AIC} = N \log\left(\frac{1}{N}\langle \Xi, \Xi\rangle\right) + 2M_s \tag{30}$$

where M_s is the model size.

When AIC is minimized, the subset selection procedure stops. Another important issue is to determine which regressor or model term is to be included into a resultant model in each step. The ERR is a popular criterion as it effectively measures the reduction in the SSE when a new regressor is included into the model. According to the ERR, a term that maximally reduces the SSE is selected.

The sum of squares of the output variables $\mathbf{y}$ is

$$\begin{aligned}\langle \mathbf{y}, \mathbf{y}\rangle &= \langle \mathbf{Wg}, \mathbf{Wg}\rangle + \langle \Xi, \Xi\rangle \\ &= \mathbf{g}^T(\mathbf{W}^T\mathbf{W})\mathbf{g} + \Xi^T\Xi. \end{aligned}\tag{31}$$

Since $\mathbf{W}^T\mathbf{W}$ is a diagonal matrix, (31) can be further simplified as

$$\langle \mathbf{y}, \mathbf{y}\rangle = \sum_{i=1}^{M_s} g_i^2 \mathbf{w}_i^T\mathbf{w}_i + \Xi^T\Xi \tag{32}$$

where $g_i = \langle \mathbf{w}_i, \mathbf{y}\rangle / \langle \mathbf{w}_i, \mathbf{w}_i\rangle$.

To minimize the SSE $\Xi^T\Xi$, the $\sum_{i=1}^{M_s} g_i^2 \mathbf{w}_i^T\mathbf{w}_i$ has to be maximized in each step when a new term is included. It can be observed that $g_i^2\mathbf{w}_i^T\mathbf{w}_i$ is the part of the sum of squares of the output contributed from the orthogonal basis $\mathbf{w}_i$ related to the model term $\mathbf{p}_i$.

The ERR value due to $\mathbf{w}_i$ is defined as [6], [22]

$$\begin{aligned}[\mathrm{err}]_i &= g_i^2\mathbf{w}_i^T\mathbf{w}_i/(\mathbf{y}^T\mathbf{y}) \\ &= g_i\mathbf{w}_i^T\mathbf{y}/(\mathbf{y}^T\mathbf{y}). \end{aligned}\tag{33}$$

Note that $\mathbf{w}_i$ in the second stage will be changed due to the refinement process and therefore $[\mathrm{err}]_i$ needs to be recalculated accordingly.

A. TSOLS Subset Selection Procedure

Given the ERR criterion, an initial model is built by selecting the most significant term whose ERR contribution is the largest at a time during the first stage, each term then being shifted to the last column position and reselected if turns out to be less significant than the remaining terms at the second stage. The details of the subset selection procedure for TSOLS using the ERR criterion are now summarized below.

1) First Stage—Forward Subset Selection [22]: At the first step, for $1 \le i \le M$, the following are calculated:

$$\left.\begin{aligned}\mathbf{w}_1^{(i)} &= \mathbf{p}_i \\ g_1^{(i)} &= \frac{\langle \mathbf{w}_1^{(i)}, \mathbf{y}\rangle}{\langle \mathbf{w}_1^{(i)}, \mathbf{w}_1^{(i)}\rangle} \\ [\mathrm{err}]_1^{(i)} &= g_1^{(i)}\frac{\langle \mathbf{w}_1^{(i)}, \mathbf{y}\rangle}{\langle \mathbf{y}, \mathbf{y}\rangle}\end{aligned}\right\}. \tag{34}$$

The largest ERR is then found as

$$[\mathrm{err}]_1^{(i_1)} = \max\left\{[\mathrm{err}]_1^{(i)},\ 1 \le i \le M\right\} \tag{35}$$

and the regressor associated with the number i_1 is selected as

$$\mathbf{w}_1 = \mathbf{w}_1^{(i_1)} = \mathbf{p}_{i_1}. \tag{36}$$

At the kth step, for $1 \le i \le M,\ i \ne i_1, \ldots, i \ne i_{k-1}$, the following are calculated:

$$\left.\begin{aligned}\alpha_{jk}^{(i)} &= \frac{\langle \mathbf{w}_j, \mathbf{p}_i\rangle}{\langle \mathbf{w}_j, \mathbf{w}_j\rangle}, 1 \le j < k \\ \mathbf{w}_k^{(i)} &= \mathbf{p}_i - \sum_{j=1}^{k-1}\alpha_{jk}^{(i)}\mathbf{w}_j \\ g_k^{(i)} &= \frac{\langle \mathbf{w}_k^{(i)}, \mathbf{y}\rangle}{\langle \mathbf{w}_k^{(i)}, \mathbf{w}_k^{(i)}\rangle} \\ [\mathrm{err}]_k^{(i)} &= g_k^{(i)}\frac{\langle \mathbf{w}_k^{(i)}, \mathbf{y}\rangle}{\langle \mathbf{y}, \mathbf{y}\rangle}\end{aligned}\right\}. \tag{37}$$

The largest ERR value is again calculated using

$$[\mathrm{err}]_k^{(i_k)} = \max\left\{[\mathrm{err}]_k^{(i)},\ i \ne i_1, \ldots, i_{k-1}\right\} \tag{38}$$

and the regressor associated with the number i_k is chosen as

$$\mathbf{w}_k = \mathbf{w}_k^{(i_k)} = \mathbf{p}_{i_k} - \sum_{j=1}^{k-1}\alpha_{jk}^{(i_k)}\mathbf{w}_j. \tag{39}$$

This procedure terminates at the M_sth step when

$$\mathrm{AIC} = N \log\left(\frac{1}{N}\langle \Xi, \Xi\rangle\right) + 2M_s \tag{40}$$

is minimized, where

$$\langle \Xi, \Xi\rangle = \langle \mathbf{y}, \mathbf{y}\rangle\left(1 - \sum_{j=1}^{M_s}[\mathrm{err}]_j^{(i_j)}\right). \tag{41}$$

Alternatively, the prefixed term number M_s or other statistical information based criteria, like BIC [14] and FPE [15], can be used to measure the model performance and stop this procedure.

At the end of the first subset selection stage, an initial model with regressors $\{\mathbf{p}_{i_1}, \ldots, \mathbf{p}_{i_j}, \ldots, \mathbf{p}_{i_{M_s}}\}$ has been decided, i_j being the index for the selected regressor in the original candidate pool. For the convenience of notation in the second stage, the indexes of the selected regressors are reordered as $\{\mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_{M_s}\}$ and those remaining in the candidate pool are renamed sequentially to generate the remaining candidate pool $\{\mathbf{p}_{M_s+1}, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_M\}$.

2) Second Stage—Backward Model Refinement: Due to the correlations of late selected regressors with early selected ones and the fact that later regressor terms selected are of less significance than earlier ones in terms of ERR, model refinement starts in a backward direction. The orthogonal basis vector of interest corresponding to an early selected regressor $\mathbf{w}_n, n = M_s - 1, \ldots, 1$ is swapped to the last column position using (24), (28), and (29). More specifically

$$\left.\begin{aligned} \mathbf{w}'_n &= \mathbf{p}_{n+1} - \sum_{i=1}^{n-1} \alpha_{i(n+1)} \mathbf{w}_i \\ \mathbf{w}'_j &= \mathbf{p}_{j+1} - \sum_{i=1}^{n-1} \alpha_{i(j+1)} \mathbf{w}_i - \sum_{i=n}^{j-1} \frac{\langle \mathbf{w}'_i, \mathbf{p}_{j+1} \rangle}{\langle \mathbf{w}'_i, \mathbf{w}'_i \rangle} \\ &\quad \cdot \mathbf{w}'_i, \quad j = n+1, \ldots, M_s - 1 \\ \mathbf{w}'_{M_s} &= \mathbf{p}_n - \sum_{i=1}^{n-1} \alpha_{in} \mathbf{w}_i - \sum_{i=n}^{M_s-1} \frac{\langle \mathbf{w}'_i, \mathbf{p}_n \rangle}{\langle \mathbf{w}'_i, \mathbf{w}'_i \rangle} \mathbf{w}'_i \end{aligned}\right\}. \quad (42)$$

To refine the last column, for $M_s \le i \le M$, the following terms are then computed:

$$\left.\begin{aligned} \mathbf{w}_{M_s}^{(i)\prime} &= \mathbf{p}_i - \sum_{j=1}^{n-1} \alpha_{ji} \mathbf{w}_j - \sum_{j=n}^{M_s-1} \frac{\langle \mathbf{w}'_j, \mathbf{p}_i \rangle}{\langle \mathbf{w}'_j, \mathbf{w}'_j \rangle} \mathbf{w}'_j \\ g_{M_s}^{(i)\prime} &= \frac{\langle \mathbf{w}_{M_s}^{(i)\prime}, \mathbf{y} \rangle}{\langle \mathbf{w}_{M_s}^{(i)\prime}, \mathbf{w}_{M_s}^{(i)\prime} \rangle} \\ [\mathrm{err}]_{M_s}^{(i)\prime} &= g_{M_s}^{(i)\prime} \frac{\langle \mathbf{w}_{M_s}^{(i)\prime}, \mathbf{y} \rangle}{\langle \mathbf{y}, \mathbf{y} \rangle}. \end{aligned}\right\}. \quad (43)$$

Finally

$$[\mathrm{err}]_{M_s}^{(i_{M_s})\prime} = \max \{ [\mathrm{err}]_{M_s}^{(i)\prime}, \ M_s \le i \le M \} \quad (44)$$

and

$$\mathbf{w}'_{M_s} = \mathbf{w}_{M_s}^{(i_{M_s})\prime} \quad (45)$$

are calculated.

The above procedure reviews all the terms selected at the first stage. If any has been replaced, the resultant new model is also refined. The whole process is repeated until no selected term can be replaced.

Since the subset selection can still be computationally demanding for a large candidate pool, it is important to further improve computational efficiency. The major part of the computation in the above OLS algorithm is consumed in calculating the inner products repetitively. Two ways are therefore employed to reduce this and to avoid calculation of unnecessary intermediate variables. More specifically, the whole procedure involves calculation of inner products of the following three intermediate matrices and two vectors:

$$[\langle \mathbf{w}_i, \mathbf{p}_j \rangle, j = i, \ldots, M, i = 1, \ldots, M_s] \quad (46)$$

$$\left[\langle \mathbf{w}_i^{(j)}, \mathbf{y} \rangle, j = i, \ldots, M, i = 1, \ldots, M_s\right] \quad (47)$$

$$\left[\langle \mathbf{w}_i^{(j)}, \mathbf{w}_i^{(j)} \rangle, j = i, \ldots, M, i = 1, \ldots, M_s\right] \quad (48)$$

$$[\langle \mathbf{w}_i, \mathbf{w}_i \rangle, i = 1, \ldots, M_s] \quad (49)$$

$$[\langle \mathbf{w}_i, \mathbf{y} \rangle, i = 1, \ldots, M_s]. \quad (50)$$

The variables in (46), (49), and (50) can be reused at each selection step, so they are stored to avoid repetitive computation. The variables in (47) and (48) are only used once, so it is unnecessary to store their values. Furthermore, calculating ERR only requires the inner products of $\langle \mathbf{w}, \mathbf{p} \rangle$, $\langle \mathbf{w}, \mathbf{w} \rangle$, and $\langle \mathbf{w}, \mathbf{y} \rangle$ instead of the actual orthogonal basis $\mathbf{w}$. Computation of the orthogonal terms $\mathbf{w}$ is therefore unnecessary. The related computation can be significantly reduced without calculating the orthogonal basis $\mathbf{w}$ separately. Using properties of orthogonal matrix $\langle \mathbf{w}_i, \mathbf{w}_j \rangle = 0, i \ne j$, the necessary inner products can be derived as

$$\begin{aligned} \langle \mathbf{w}_k, \mathbf{p}_j \rangle &= \langle \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik} \mathbf{w}_i, \mathbf{p}_j \rangle \\ &= \langle \mathbf{p}_k, \mathbf{p}_j \rangle - \sum_{i=1}^{k-1} \alpha_{ik} \langle \mathbf{w}_i, \mathbf{p}_j \rangle \quad (51) \\ \langle \mathbf{w}_k, \mathbf{w}_k \rangle &= \langle \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik} \mathbf{w}_i, \mathbf{w}_k \rangle \\ &= \langle \mathbf{p}_k, \mathbf{w}_k \rangle \\ &= \langle \mathbf{p}_k, \mathbf{p}_k \rangle - \sum_{i=1}^{k-1} \alpha_{ik} \langle \mathbf{w}_i, \mathbf{p}_k \rangle \quad (52) \\ \langle \mathbf{w}_k, \mathbf{y} \rangle &= \langle \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik} \mathbf{w}_i, \mathbf{y} \rangle \\ &= \langle \mathbf{p}_k, \mathbf{y} \rangle - \sum_{i=1}^{k-1} \alpha_{ik} \langle \mathbf{w}_i, \mathbf{y} \rangle \quad (53) \end{aligned}$$

while a similar conclusion has been reported in [46], the calculation of $\langle \mathbf{w}, \mathbf{w} \rangle$ has been further simplified here in this paper as it avoids calculating the squares of α. Having discussed ways to reduce the computation in the TSOLS subset selection algorithm, the computation complexity is further analyzed next.

B. Computational Complexity

The computational requirements for each stage of the TSOLS approach are computed separately. Suppose N is the size of the training data set and M_s terms are selected from the pool of M candidate terms. In the algorithm, the main computation arises from calculating the inner products of $\langle \mathbf{w}, \mathbf{p} \rangle$, $\langle \mathbf{w}, \mathbf{y} \rangle$, and $\langle \mathbf{w}, \mathbf{w} \rangle$. At the first stage, the total number of inner product operations is given by

$$\begin{aligned} C_1 = NMM_s - \frac{1}{2}N(M_s+1)M_s + 2NM - M_s^2 + MM_s \\ -\frac{1}{2}(Ms-1)M_s(2M_s-1) + \frac{3}{2}MM_s(M_s-1). \quad (54) \end{aligned}$$

At the second stage, the computation includes the shifting of each selected term to the last position and then comparing its significance with those left in the pool of candidate terms. In the worst case, all the previously selected regressor terms are insignificant compared with the candidate terms, then the total computation involved in shifting and comparing during one check loop at the second stage of OLS is then given by

$$C_2 = C_{2\mathrm{shift}} + C_{2\mathrm{compare}} \quad (55)$$

TABLE I
COMPUTATIONAL COMPLEXITY FOR TSFRA AND TSOLS

	First stage	Second stage
Two-stage FRA	$o(NMM_s)$	$o(NMM_s + MM_s^4/6)$
Two-stage OLS	$o(NMM_s)$	$o(NMM_s + MM_s^3/6)$

where

$$C_{2\text{shift}} = \frac{1}{6}(M - M_s + 2)(M_s - 2)(M_s - 1)(2M_s - 3) \quad (56)$$

and

$$C_{2\text{compare}} = N(M - M_s)(M_s - 1)) + (M - M_s)(M_s - 1)(3M_s - 2). \quad (57)$$

Comparison with TSFRA is now discussed. At the first stage, our proposed method and FRA require the same computation, given in (54). In the second stage, the shifting and comparing operations are treated separately. For the shifting operations, the number of term exchanges required by our method is $1/2M_s(M_s - 1)$ per check loop, compared with $1/2M_s^2(M_s - 1)$ for TSFRA. Therefore, the total computation consumed by this new method is $1/M_s$ of the TSFRA. For the comparison operations, the two methods are identical (57). In conclusion, the proposed method is more efficient than TSFRA, and the computational saving improves as the number of selected terms M_s increases.

In practice, the number of the selected model terms M_s usually satisfies $M_s \ll N, M$, therefore the computational complexities for the TSOLS and the TSFRA can be simplified, and they are summarized in Table I, where o represents the floating operations. It is clear that the new TSOLS method needs M_s times less calculations than the original TSFRA in the second stage when the final model size M_s is reasonably large and it saves more computation effort as the model size M_s increases.

V. EXTENSION TO TWO-STAGE MGS OLS

If the decomposed matrix is illconditional or singular, the MGS method is more preferable than the CGS method as it is numerically less sensitive to round-off errors than the CGS algorithm [47], [48]. The two-stage CGS-based OLS can be easily extended to MGS-based OLS version.

The CGS and MGS methods have differences in computational procedure. The CGS computes orthogonal basis $\mathbf{w}_k$ as

$$\mathbf{w}_k = \mathbf{p}_k - \sum_{i=1}^{k-1} \mathtt{proj}_{\mathbf{w}_i}(\mathbf{p}_k) \quad (58)$$

where $\mathtt{proj}_{\mathbf{w}_i}(\mathbf{p}_k) = \langle \mathbf{w}_i, \mathbf{p}_k \rangle / \langle \mathbf{w}_i, \mathbf{w}_i \rangle \mathbf{w}_i = \alpha_{ik}\mathbf{w}_i$ represents the projection of $\mathbf{p}_k$ into the orthogonal basis vector $\mathbf{w}_i$. In the kth step of CGS, the column $\mathbf{p}_k$ is orthogonalized against $\mathbf{w}_1, \ldots, \mathbf{w}_{k-1}$. At step k, the kth column of $\mathbf{A}$ is computed and it is operated on the first k columns, and the CGS mainly computes

$$\mathbf{W}_k = [\mathbf{w}_1, \ldots, \mathbf{w}_k, \mathbf{p}_{k+1}, \ldots, \mathbf{p}_M] \quad (59)$$

where the latter columns $\mathbf{p}_{k+1}, \ldots, \mathbf{p}_M$ remain unchanged. However, at step k, the MGS computes

$$\mathbf{Q}_k = [\mathbf{w}_1, \ldots, \mathbf{w}_k, \mathbf{p}_{k+1}^{(k)}, \ldots, \mathbf{p}_M^{(k)}] \quad (60)$$

where $\mathbf{p}_{k+1}^{(k)}, \ldots, \mathbf{p}_M^{(k)}$ have been made orthogonal to $\mathbf{w}_1, \ldots, \mathbf{w}_k$. The crucial difference is that in MGS, the projections are updated by the subtraction of $\mathbf{p}_{k+1}$ to $\mathbf{p}_M$ once the kth orthogonal basis vector is obtained. In other words, when each new orthogonal basis is determined, all the remaining columns minus their projections to the new basis. The MGS procedure calculates one row of $\mathbf{A}$ at a time and factorizes $\mathbf{P}$ as follows:

$$\left.\begin{aligned} \mathbf{w}_k &= \mathbf{p}_k^{(k-1)} \\ \alpha_{ki} &= \frac{\langle \mathbf{w}_k, \mathbf{p}_i^{(k-1)} \rangle}{\langle \mathbf{w}_k, \mathbf{w}_k \rangle} \\ i &= k+1, \ldots, M \\ \mathbf{p}_i^{(k)} &= \mathbf{p}_i^{(k-1)} - \alpha_{ki}\mathbf{w}_k \\ i &= k+1, \ldots, M \end{aligned}\right\} k = 1, \ldots, M-1 \left.\begin{aligned} \\ \\ \\ \\ \\ \mathbf{w}_M &= \mathbf{p}_M^{(M-1)} \end{aligned}\right\} \quad (61)$$

where $\mathbf{p}_i^{(0)} = \mathbf{p}_i$ and $i = 1, \ldots, M$.

In the first stage, the only difference with the CGS method is that the CGS orthogonalization is replaced with MGS method. An initial model $\mathbf{P}_{M_s}$ is selected and its orthogonal decomposition, including an upper matrix $\mathbf{A}$ (62), shown at the top of the next page and $\mathbf{Q}$ (63), shown at the top of the next page is produced, where the $\mathbf{p}_i^{(0)} = \mathbf{p}_i$, $i = 1, \ldots, M$ and $\mathbf{p}_k^{(k-1)} = \mathbf{w}_k, k = 1, \ldots, M_s$.

In the second stage, suppose moving $\mathbf{p}_n$ to the last column $\mathbf{p}_{M_s}$. The above matrices become $\mathbf{A}'$ (64) and $\mathbf{Q}'$ (65), shown at the top of the next page respectively.

The variables with $'$ are necessary to be recomputed and they can be computed as follows.

1) Update variables from the first row to the nth row by

$$\left.\begin{aligned} \alpha'_{ij} &= \begin{cases} \alpha_{i(j+1)}, i = 1, \ldots, n-1, \\ \qquad j = n, \ldots, M_s - 1 \\ \alpha_{in}, i = 1, \ldots, n-1, j = M_s \end{cases} \\ \mathbf{p}_j^{(i)\prime} &= \begin{cases} \mathbf{p}_{j+1}^{(i)}, i = 0, \ldots, n-1, \\ \qquad j = n, \ldots, M_s - 1 \\ \mathbf{p}_n^{(i)}, i = 0, \ldots, n-1, j = M_s \end{cases} \\ \mathbf{w}_n{}' &= \mathbf{p}_n^{(n-1)\prime} = \mathbf{p}_{n+1}^{(n-1)} \end{aligned}\right\} \quad (66)$$

where only the elements from column n to M_s are changed and they can be directly obtained from the previous results without recalculation while other elements remain unchanged.

$$\mathbf{A}=\begin{bmatrix} 1 & \alpha_{12} & \cdots & \alpha_{1(n-1)} & \alpha_{1n} & \cdots & \alpha_{1M_s} & \alpha_{1(M_s+1)} & \cdots & \alpha_{1M} \\ & 1 & \cdots & \alpha_{2(n-1)} & \alpha_{2n} & \cdots & \alpha_{2M_s} & \alpha_{2(M_s+1)} & \cdots & \alpha_{2M} \\ & & \ddots & \vdots & \vdots & \cdots & \vdots & & & \\ & & & 1 & \alpha_{(n-1)n} & \cdots & \alpha_{(n-1)M_s} & \alpha_{(n-1)(M_s+1)} & \cdots & \alpha_{(n-1)M} \\ & & & & 1 & \cdots & \alpha_{nM_s} & \alpha_{n(M_s+1)} & \cdots & \alpha_{nM} \\ & & & & & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & & & & & & 1 & \alpha_{M_s(M_s+1)} & \cdots & \alpha_{M_sM} \end{bmatrix} \tag{62}$$

$$\mathbf{Q}=\begin{bmatrix} \mathbf{p}_1^{(0)} & \mathbf{p}_2^{(0)} & \cdots & \mathbf{p}_{n-1}^{(0)} & \mathbf{p}_n^{(0)} & \mathbf{p}_{n+1}^{(0)} & \cdots & \mathbf{p}_{M_s}^{(0)} & \mathbf{p}_{M_s+1}^{(0)} & \cdots & \mathbf{p}_M^{(0)} \\ & \mathbf{p}_2^{(1)} & \cdots & \mathbf{p}_{n-1}^{(1)} & \mathbf{p}_n^{(1)} & \mathbf{p}_{n+1}^{(1)} & \cdots & \mathbf{p}_{M_s}^{(1)} & \mathbf{p}_{M_s+1}^{(1)} & \cdots & \mathbf{p}_M^{(1)} \\ & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ & & & \mathbf{p}_{n-1}^{(n-2)} & \mathbf{p}_n^{(n-2)} & \mathbf{p}_{n+1}^{(n-2)} & \cdots & \mathbf{p}_{M_s}^{(n-2)} & \mathbf{p}_{M_s+1}^{(n-2)} & \cdots & \mathbf{p}_M^{(n-2)} \\ & & & & \mathbf{p}_n^{(n-1)} & \mathbf{p}_{n+1}^{(n-1)} & \cdots & \mathbf{p}_{M_s}^{(n-1)} & \mathbf{p}_{M_s+1}^{(n-1)} & \cdots & \mathbf{p}_M^{(n-1)} \\ & & & & & \mathbf{p}_{n+1}^{(n)} & \cdots & \mathbf{p}_{M_s}^{(n)} & \mathbf{p}_{M_s+1}^{(n)} & \cdots & \mathbf{p}_M^{(n)} \\ & & \mathbf{0} & & & & \ddots & \vdots & \vdots & \cdots & \vdots \\ & & & & & & & \mathbf{p}_{M_s}^{(M_s-1)} & \mathbf{p}_{M_s+1}^{(M_s-1)} & \cdots & \mathbf{p}_M^{(M_s-1)} \end{bmatrix} \tag{63}$$

$$\mathbf{A}'=\begin{bmatrix} 1 & \alpha_{12} & \cdots & \alpha_{1(n-1)} & \alpha'_{1n} & \cdots & \alpha'_{1M_s} & \alpha_{1(M_s+1)} & \cdots & \alpha_{1M} \\ & 1 & \cdots & \alpha_{2(n-1)} & \alpha'_{2n} & \cdots & \alpha'_{2M_s} & \alpha_{2(M_s+1)} & \cdots & \alpha_{2M} \\ & & \ddots & \vdots & \vdots & \cdots & \vdots & & & \\ & & & 1 & \alpha'_{(n-1)n} & \cdots & \alpha'_{(n-1)M_s} & \alpha_{(n-1)(M_s+1)} & \cdots & \alpha_{(n-1)M} \\ & & & & 1 & \cdots & \alpha'_{nM_s} & \alpha'_{n(M_s+1)} & \cdots & \alpha'_{nM} \\ & & & & & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & & & & & & 1 & \alpha'_{M_s(M_s+1)} & \cdots & \alpha'_{M_sM} \end{bmatrix} \tag{64}$$

$$\mathbf{Q}'=\begin{bmatrix} \mathbf{p}_1^{(0)} & \mathbf{p}_2^{(0)} & \cdots & \mathbf{p}_{n-1}^{(0)} & \mathbf{p}_n^{(0)\prime} & \mathbf{p}_{n+1}^{(0)\prime} & \cdots & \mathbf{p}_{M_s}^{(0)\prime} & \mathbf{p}_{M_s+1}^{(0)} & \cdots & \mathbf{p}_M^{(0)} \\ & \mathbf{p}_2^{(1)} & \cdots & \mathbf{p}_{n-1}^{(1)} & \mathbf{p}_n^{(1)\prime} & \mathbf{p}_{n+1}^{(1)\prime} & \cdots & \mathbf{p}_{M_s}^{(1)\prime} & \mathbf{p}_{M_s+1}^{(1)} & \cdots & \mathbf{p}_M^{(1)} \\ & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ & & & \mathbf{p}_{n-1}^{(n-2)} & \mathbf{p}_n^{(n-2)\prime} & \mathbf{p}_{n+1}^{(n-2)\prime} & \cdots & \mathbf{p}_{M_s}^{(n-2)\prime} & \mathbf{p}_{M_s+1}^{(n-2)} & \cdots & \mathbf{p}_M^{(n-2)} \\ & & & & \mathbf{p}_n^{(n-1)\prime} & \mathbf{p}_{n+1}^{(n-1)\prime} & \cdots & \mathbf{p}_{M_s}^{(n-1)\prime} & \mathbf{p}_{M_s+1}^{(n-1)} & \cdots & \mathbf{p}_M^{(n-1)} \\ & & & & & \mathbf{p}_{n+1}^{(n)\prime} & \cdots & \mathbf{p}_{M_s}^{(n)\prime} & \mathbf{p}_{M_s+1}^{(n)\prime} & \cdots & \mathbf{p}_M^{(n)\prime} \\ & & \mathbf{0} & & & & \ddots & \vdots & \vdots & \cdots & \vdots \\ & & & & & & & \mathbf{p}_{M_s}^{(M_s-1)\prime} & \mathbf{p}_{M_s+1}^{(M_s-1)\prime} & \cdots & \mathbf{p}_M^{(M_s-1)\prime}. \end{bmatrix} \tag{65}$$

2) Update variables from the $(n+1)$th row by

$$\left.\begin{aligned} \alpha'_{nj}&=\begin{cases}\dfrac{\langle \mathbf{w}_n', \mathbf{p}_j^{(n-1)\prime}\rangle}{\langle \mathbf{w}_n', \mathbf{w}_n'\rangle}, & j=n+1,\ldots,M_s \\ \dfrac{\langle \mathbf{w}'_n, \mathbf{p}_j^{(n-1)}\rangle}{\langle \mathbf{w}'_n, \mathbf{w}'_n\rangle}, & j=M_s+1,\ldots,M\end{cases} \\ \mathbf{p}_j^{(n)\prime}&=\begin{cases}\mathbf{p}_j^{(n-1)\prime}-\alpha'_{nj}\mathbf{w}_n', & j=n+1,\ldots,M_s \\ \mathbf{p}_j^{(n-1)}-\alpha'_{nj}\mathbf{w}'_n, & j=M_s+1,\ldots,M\end{cases} \\ \mathbf{w}_{n+1}{}'&=\mathbf{p}_{n+1}^{(n)\prime} \end{aligned}\right\}. \tag{67}$$

It is clear that all the variables from column $n+1$ to column M are changed due to the changed $\mathbf{w}'_n$.

3) Update variables from the $(n+2)$th row to the M_sth row by

$$\left.\begin{aligned} \alpha'_{ij}&=\frac{\langle \mathbf{w}_i', \mathbf{p}_j^{(i-1)\prime}\rangle}{\langle \mathbf{w}_i', \mathbf{w}_i'\rangle}, \\ & j=i+1,\ldots,M \\ \mathbf{p}_j^{(i)\prime}&=\mathbf{p}_j^{(i-1)\prime}-\alpha'_{ij}\mathbf{w}_i', \\ & j=i+1,\ldots,M \\ \mathbf{w}_{i+1}{}'&=\mathbf{p}_{i+1}^{(i)\prime} \end{aligned}\right\} i=n+1,\ldots,M_s-1 \tag{68}$$

where variables both in $\mathbf{A}'$ and $\mathbf{Q}'$ are updated one row at a time.

After the column $\mathbf{p}_n$ is moved to the last column, it is compared with the remaining terms in the candidate pool. This process continues until no term can be replaced by others.

Under the ERR criterion, the two skills used by two-stage CGS OLS, namely storing intermediate variables and avoiding

calculating the orthogonal basis, are also suitable for the two-stage MGS OLS to reduce computations. Using the orthogonal property $\langle \mathbf{w}_i, \mathbf{w}_j \rangle = 0, i \neq j$ and the formula in (61), calculate the inner products by (69)–(71). It is clear that the CGS and MGS have the same formula for the inner product operations. Therefore, the fast two-stage MGS OLS is the same as the fast two-stage CGS OLS in terms of computational complexity

$$\begin{aligned}\langle \mathbf{w}_k, \mathbf{p}_j^{(k-1)} \rangle &= \langle \mathbf{w}_k, \mathbf{p}_j^{(k-2)} - \alpha_{(k-1)j}\mathbf{w}_{k-1} \rangle \\ &= \langle \mathbf{w}_k, \mathbf{p}_j - \sum_{i=1}^{k-1} \alpha_{ij}\mathbf{w}_i \rangle = \langle \mathbf{w}_k, \mathbf{p}_j \rangle = \langle \mathbf{p}_k^{(k-1)}, \mathbf{p}_j \rangle \\ &= \langle \mathbf{p}_k^{(k-2)} - \alpha_{(k-1)k}\mathbf{w}_{k-1}, \mathbf{p}_j \rangle \\ &= \langle \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik}\mathbf{w}_i, \mathbf{p}_j \rangle \\ &= \langle \mathbf{p}_k, \mathbf{p}_j \rangle - \sum_{i=1}^{k-1} \alpha_{ik} \langle \mathbf{w}_i, \mathbf{p}_j \rangle \end{aligned} \tag{69}$$

$$\begin{aligned}\langle \mathbf{w}_k, \mathbf{w}_k \rangle &= \langle \mathbf{w}_k, \mathbf{p}_k^{(k-2)} - \alpha_{(k-1)k}\mathbf{w}_{k-1} \rangle \\ &= \langle \mathbf{w}_k, \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik}\mathbf{w}_i \rangle \\ &= \langle \mathbf{w}_k, \mathbf{p}_k \rangle = \langle \mathbf{p}_k^{(k-1)}, \mathbf{p}_k \rangle \\ &= \langle \mathbf{p}_k, \mathbf{p}_k \rangle - \sum_{i=1}^{k-1} \alpha_{ik} \langle \mathbf{w}_i, \mathbf{p}_k \rangle \end{aligned} \tag{70}$$

$$\begin{aligned}\langle \mathbf{w}_k, \mathbf{y} \rangle &= \langle \mathbf{p}_k^{(k-2)} - \alpha_{(k-1)k}\mathbf{w}_{k-1}, \mathbf{y} \rangle \\ &= \langle \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik}\mathbf{w}_i, \mathbf{y} \rangle = \langle \mathbf{p}_k, \mathbf{y} \rangle - \sum_{i=1}^{k-1} \alpha_{ik} \langle \mathbf{w}_i, \mathbf{y} \rangle. \end{aligned} \tag{71}$$

VI. Numerical Examples

Three numerical experiments were carried out using MATLAB R2011a on a desktop Intel E8400 PC with Windows 7 system.

A. Polynomial NARX Modeling

This example of modeling a nonlinear time series using a polynomial NAR model, which is a special case of NARX model, is taken from [16]

$$\begin{aligned} y(t) = &(0.8 - 0.5e^{-y^2(t-1)})y(t-1) \\ &- (0.3 + 0.9e^{-y^2(t-1)})y(t-2) \\ &+ 0.1\sin(\pi y(t-1)) + \xi(t). \end{aligned} \tag{72}$$

Here, y denotes the system output with respect to the time series t, and $\xi(t)$ is a zero-mean Gaussian white noise sequence with variance 0.02^2. A set of 1000 data samples were generated, for two initial conditions $y(0) = 0.01$ and $y(1) = 0.1$, half being used for training and the remainder for testing. Here, polynomials with orders up to 3 for variables $y(t-q)$ $(q = 1, 2, 3, 4)$ were used for the nonlinear functions $\mathbf{p}$ in (1). All the four variables form 34 polynomial terms with 4 first-, 10 second-, and 20 third-order terms, which is shown in Table II.

TABLE II
34 Polynomial Terms

No.	terms
1-5	$y(t-1), y(t-2), y(t-3), y(t-4), y^2(t-1)$
6-8	$y(t-1)y(t-2), y(t-1)y(t-3), y(t-1)y(t-4)$
9-12	$y^2(t-2), y(t-2)y(t-3), y(t-2)y(t-4), y^2(t-3)$
13-16	$y(t-3)y(t-4), y^2(t-4), y^3(t-1), y^2(t-1)y(t-2)$
17-19	$y^2(t-1)y(t-3), y^2(t-1)y(t-4), y(t-1)y^2(t-2)$
20-21	$y(t-1)y^2(t-3), y(t-1)y^2(t-4)$
22-23	$y(t-1)y(t-2)y(t-3), y(t-1)y(t-2)y(t-4)$
24-25	$y(t-1)y(t-3)y(t-4), y^3(t-2)$
26-28	$y^2(t-2)y(t-3), y^2(t-2)y(t-4), y(t-2)y^2(t-3)$
29-31	$y(t-2)y^2(t-4), y(t-2)y(t-3)y(t-4), y^3(t-3)$
32-34	$y^2(t-3)y(t-4), y(t-3)y^2(t-4), y^3(t-4)$

TABLE III
Monte Carlo Simulation-Based Model Performance Using Different Methods for Polynomial NARX Modeling

Methods	Size	Time(ms)	Training SSE	Test SSE
OLS [22]	18	18	0.20 ± 0.01	0.87 ± 0.07
LAR [41]	19	**10**	0.22 ± 0.03	0.87 ± 0.11
OOMP [27]	18	**10**	0.20 ± 0.01	0.87 ± 0.07
LASSO-bc [37]	34	13	0.19 ± 0.00	0.87 ± 0.08
LASSO-gr [38]	28	2299	0.19 ± 0.00	0.87 ± 0.08
LASSO-ir [39]	32	**10**	0.19 ± 0.00	0.88 ± 0.07
LASSO-pl [40]	19	26	0.21 ± 0.01	0.87 ± 0.09
TSFRA [16]	**15**	46	0.19 ± 0.01	0.87 ± 0.07
TSOLS	**15**	26	0.19 ± 0.01	0.87 ± 0.07

The SSEs over the training data and testing data sets are given by training and test SSEs, respectively, and their standard deviations are also given. To demonstrate the superiority of the proposed TSOLS algorithm, it was compared with the forward OLS [22], LAR [41], OOMP [27], TSFRA [16], and four LASSO methods including LASSO-bc [37], LASSO-gr [38], LASSO-ir [39], and LASSO-pl [40]. To make a fair comparison, the regularization parameters used in LASSO methods are carefully tuned. To obtain repeatable results, the Monte Carlo simulation with 100 trials was conducted. As shown in Table III, all these methods produced the similar training and test SSEs. However, their average model sizes and running time are quite different.

Although LAR, OOMP, and LASSO-ir used the least computation time among all the methods, they did not produce the most compact model with the smallest model size, while both the TSFRA and the proposed TSOLS method were able to build a more compact model than OLS, LAR, OOMP, and four LASSO methods. Another advantage of the new technique is that it needs less computing time than the TSFRA approach. According to the parsimonious principle, a simple model that has a good generalization performance (smallest test SSE) is always preferable. From this point of view, the model built by TSOLS is better than those constructed by the other methods.

TABLE IV
MONTE CARLO SIMULATION-BASED MODEL PERFORMANCE USING DIFFERENT METHODS FOR RBF NETWORKS CONSTRUCTION WITH NOISE VARIANCE 0.2^2

Methods	Size	Time(ms)	Training SSE	Test SSE
OLS [22]	15	75	7.47 ± 0.24	8.70 ± 0.59
LAR [41]	20	**25**	7.35 ± 0.23	8.66 ± 0.47
OOMP [27]	15	42	7.47 ± 0.24	8.70 ± 0.59
LASSO-bc [37]	200	88	7.13 ± 0.21	8.68 ± 0.56
LASSO-gr [38]	124	4362	7.03 ± 0.19	8.73 ± 0.56
LASSO-ir [39]	26	550	6.99 ± 0.19	8.76 ± 0.56
LASSO-pl [40]	18	125	7.45 ±0.17	8.62 ± 0.37
TSFRA [16]	**13**	122	7.59 ± 0.18	8.60 ± 0.50
TSOLS	**13**	79	7.59 ± 0.18	8.60 ± 0.50

TABLE V
MONTE CARLO SIMULATION-BASED MODEL PERFORMANCE USING DIFFERENT METHODS FOR RBF NETWORKS CONSTRUCTION WITH NOISE VARIANCE 0.3^2

Methods	Size	Time	Training SSE	Test SSE
OLS [22]	7	51	17.44 ± 0.47	19.13 ± 0.58
LAR [41]	12	**24**	17.11 ± 0.57	19.12 ± 0.52
OOMP [27]	7	31	17.44 ± 0.47	19.17 ± 0.58
LASSO-bc [37]	200	150	16.23 ± 0.62	19.52 ± 1.24
LASSO-gr [38]	148	4794	16.05 ± 0.56	19.71 ± 1.11
LASSO-ir [39]	27	932	15.99 ± 0.56	19.79 ± 1.10
LASSO-pl [40]	10	99	17.29 ±0.51	19.08 ± 0.40
TSFRA [16]	**5**	98	17.62 ± 0.54	19.11 ± 0.69
TSOLS	**5**	50	17.62 ± 0.54	19.11 ± 0.69

TABLE VI
MONTE CARLO SIMULATION-BASED MODEL PERFORMANCE USING DIFFERENT METHODS FOR WAVELET NETWORKS CONSTRUCTION WITH NOISE VARIANCE 0.2^2

Methods	Size	Time (ms)	Training SSE	Test SSE
OLS [22]	25	47	11.34 ± 0.46	13.79 ± 0.73
LAR [41]	43	65	10.63 ± 0.39	13.92 ± 0.63
OOMP [27]	25	17	11.34 ± 0.46	13.79 ± 0.73
LASSO-bc [37]	49	**15**	10.72 ± 0.33	13.80 ± 0.59
LASSO-gr [38]	37	21621	10.83 ± 0.38	13.85 ± 0.63
LASSO-ir [39]	41	249	10.58 ± 0.33	13.84 ± 0.60
LASSO-pl [40]	40	296	11.84 ±4.37	15.04 ± 4.83
TSFRA [16]	**23**	341	11.38 ± 0.47	13.77 ± 0.69
TSOLS	**23**	72	11.38 ± 0.47	13.77 ± 0.69

TABLE VII
MONTE CARLO SIMULATION-BASED MODEL PERFORMANCE USING DIFFERENT METHODS FOR WAVELET NETWORKS CONSTRUCTION WITH NOISE VARIANCE 0.3^2

Methods	Size	Time(ms)	Training SSE	Test SSE
OLS [22]	22	39	25.95 ± 1.64	31.02 ± 1.63
LAR [41]	40	52	23.96 ± 1.06	31.13 ± 1.50
OOMP [27]	22	17	25.95 ± 1.64	31.02 ± 1.63
LASSO-bc [37]	49	**15**	23.70 ± 0.75	30.97 ± 1.37
LASSO-gr [38]	41	29018	23.80 ± 0.84	31.02 ± 1.40
LASSO-ir [39]	42	198	23.40 ± 0.74	31.19 ± 1.38
LASSO-pl [40]	37	178	35.09 ±29.07	42.24 ± 31.13
TSFRA [16]	**20**	190	25.95 ± 1.66	31.01 ± 1.70
TSOLS	**20**	55	25.95 ± 1.66	31.01 ± 1.70

B. RBF Neural Networks Modeling

The second example is from [49] and concerns approximating the following by an RBF neural network:

$$y(t) = 0.1t + \frac{\sin(t)}{t} + \sin(0.5t) + \xi(t), \quad -10 \le t \le 10. \quad (73)$$

Here, t is the recursive step and it is uniformly distributed in $[-10, 10]$ and $\xi(t)$ is a Gaussian white noise with zero mean. A total of 400 data samples were generated, the first 200 data used for training and the rest for testing. The RBF network employed a Gaussian kernel function $p(\mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{c}_i\|/\sigma^2)$, where the $\mathbf{c}_i$s and σ are the RBF centers and width, respectively. The width was chosen as $\sigma = 2$ by trial-and-error, while the centers were initialized on the training data points, and to produce a total of 200 candidate terms.

The same nine methods used in constructing the polynomial NARX model were also employed again for the RBF network construction. Two different noise variances with 0.2^2 and 0.3^2 were added separately to test the robustness of these methods. Their average Monte Carlo results are listed in Tables IV and V. In both cases, all the methods performed similarly well in terms of the training and test SSEs. However, their model complexity is quite different. In both cases, the simplest models were built by the two-stage methods, namely the TSFRA and TSOLS, with results of only 13 and 5 terms, respectively. Unlike the most complex models with 200 terms by the LASSO-bc method, the TSFRA and TSOLS significantly improve the model compactness. Furthermore, the proposed TSOLS is computationally more efficient than the TSFRA.

C. Wavelet Networks Modeling

The following nonlinear single variable function is modeled using wavelet networks [50]:

$$f(x) = \begin{cases} -2.186x - 12.846 + \xi(x) & \text{if } x \in [-10, -2) \\ 4.246x + \xi(x) & \text{if } x \in [-2, 0) \\ \sin(x(0.03x + 0.7)) & \\ \cdot 10e^{(-0.05x - 0.5)} + \xi(x) & \text{if } x \in [0, 10] \end{cases} \quad (74)$$

where $\xi(x)$ is a zero-mean Gaussian white noise sequence. A set of 400 data samples were generated with x uniformly distributed in the range $[-10, 10]$, where half of them were used for network training and the other half were used for testing. The model employed the Mexican function $p(x) = 2^{-(1/2)n}(1 - \|2^{-n}x - m\|^2 e^{-(\|2^{-n}x - m\|^2/2)})$ as the wavelet function, where n and m are the dilation and translation parameters, respectively. Following [45], both m and n are integers and selected between $-3 \le m, n \le 3$ where the wavelet networks can converge, so there are 49 wavelet candidate terms in total.

Again, the same nine methods used in the above two examples were used to produce different wavelet network models under noise variances 0.2^2 and 0.3^2, respectively. The average Monte Carlo results are shown in Tables VI and VII. Although the TSFRA and TSOLS performs best in terms of the testing SSE in both cases, they are only slightly better than the others. In other words, all the methods can capture the true characteristics of the nonlinear behavior. Therefore, there is no major difference for model generalization among different methods. Their major differences lie in the model complexity and computing time. The OOMP runs fastest, but it does not

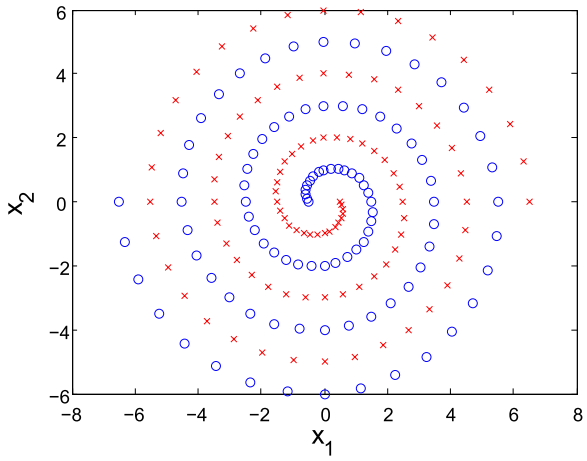


Fig. 1. Plot of two spirals data.

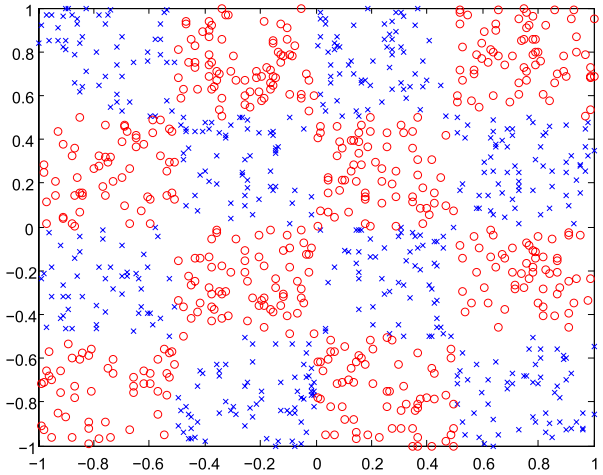


Fig. 2. Plot of chess board data.

TABLE VIII
COMPARISON OF TRAINING ERRORS FOR TWO SPIRALS CLASSIFICATION

Methods	Size	Time(s)	Misclassification rate
OLS [22]	75	2.31	00%
LAR [41]	160	**1.93**	00%
OOMP [27]	75	2.18	00%
LASSO-bc [37]	180	3.79	00%
LASSO-gr [38]	162	3.18	00%
LASSO-ir [39]	166	1.46	00%
LASSO-pl [40]	144	2.75	00%
TSFRA [16]	**72**	35.15	00%
TSOLS	**72**	3.05	00%

TABLE IX
COMPARISON OF TRAINING ERRORS

Methods	Size	Time(s)	Misclassification rate
OLS [22]	13	0.26	00%
LAR [41]	13	**0.11**	00%
OOMP [27]	75	0.21	00%
LASSO-bc [37]	180	3.79	00%
LASSO-gr [38]	162	3.18	00%
LASSO-ir [39]	166	1.46	00%
LASSO-pl [40]	144	2.75	00%
TSFRA [16]	**9**	0.45	00%
TSOLS	**9**	0.16	00%

produce a compact model. Compared with the OOMP, the TSOLS needs extra more computations to improve the model compactness, leading to an improved model with the smallest model size.

D. Classification of Two Spirals

To further test the performance of the proposed and other existing methods, classification problems are considered. The two spirals problem is a widely used classification benchmark. Fig. 1 shows the 194 data samples from the two spirals data set. The RBF networks are used for the two spirals classification problem where the width is predetermined to be 1 and the centers are initialized by 194 data points, leading to a candidate pool with 194 terms. The training performance is measured by the rates of unsuccessfully misclassified data points. The classification results produced by the different methods are shown in Table VIII. It can be observed that all the nine methods can classify the two sets successfully with 0 misclassification rate. Their differences lie in the model compactness and running time. Although the LAR uses the least computing time, it produces the most complex model with 160 terms. The TSFRA and TSOLS build the most compact model with the smallest size of 72 model terms. Compared with the TSFRA method, which used 35.15 s, the proposed TSOLS retains the computational efficiency with only 3.05 s. The saving time is much more significant than the previous examples. The reason for this is that this example is a large-scale problem. The larger the resultant model size is, the more computational time is saved by the TSOLS. This has also been confirmed by the computational complexity analysis shown in Table I.

E. Chessboard Classification

The chessboard data set contains 1000 samples from two categories, which forms a pattern similar to a chess board. The distribution of samples is shown in Fig. 2. RBF networks are again employed for the chessboard classification problem. The width is predetermined to be 0.2. The 1000 samples are selected as the centers of RBF networks, which produce a candidate term pool with 1000 terms. The classification results produced by the different methods are shown in Table IX. Similar to the previous example, all the methods can classify the two sets successfully with 0 misclassification rate. The TSFRA and TSOLS again build the simplest model with nine terms and TSOLS performs faster than TSFRA.

F. Discussion

1) It has been shown that all the nine methods can build good models with similar generalization performance in the five examples. Although the proposed method is not the fastest algorithm, it can build the most compact model among the nine methods. According to the parsimonious principle, the proposed TSOLS method is recommended if the running time is not a critical factor. 2) Although the detailed information for neural networks initialization has been given in each

example, it should be noted that many different initialization schemes are available in the literature, and the ELM approach is one of the recent developments for initializing the nonlinear parameters in the neural networks, including the width for RBF neural networks [19]. This will generate a large candidate term pool and may provide more opportunities to build a better model. However, computational time is inevitably significantly increased.

VII. Conclusion

A unified TSOLS for constructing linear-in-the-parameters models has been introduced for the first time. In the first stage (orthogonal forward selection), an initial model is generated using a forward selection procedure by selecting one term at a time from a large pool of candidates built from prior knowledge. In the second stage (backward model refinement), the contribution of each previously selected term to an ERR cost function is reviewed by shifting it to the last position and replacing with more significant ones from the candidate pool if necessary. Two implementation algorithms for the OLS-based two-stage methods are given. The computational complexity analysis has shown that the proposed method is computationally efficient than the previously proposed TSFRA. Numerical results have shown that the proposed method is able to produce a more compact neural model than conventional OLS, LAR, OOMP, and four LASSO methods, and executes faster than the TSFRA approach.

Acknowledgment

The authors would like to thank Dr. W. Zhao from the Chinese Academy of Sciences of China for his constructive comments.

References

[1] S.-W. Oh, W. Pedrycz, and B.-J. Park, "Polynomial neural networks architecture: Analysis and design," *Comput. Electr. Eng.*, vol. 29, no. 6, pp. 703–725, 2003.

[2] I. J. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems part I: Deterministic non-linear systems," *Int. J. Control*, vol. 41, no. 2, pp. 303–328, 1985.

[3] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[4] S. A. Billings, H. L. Wei, and M. A. Balikhin, "Generalized multiscale radial basis function networks," *Neural Netw.*, vol. 20, no. 10, pp. 1081–1094, 2007.

[5] L.-X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 807–814, Sep. 1992.

[6] Q. Zhang, "Using wavelet network in nonparametric estimation," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 227–236, Mar. 1997.

[7] S. A. Billings and H.-L. Wei, "A new class of wavelet networks for nonlinear system identification," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 862–874, Jul. 2005.

[8] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1997.

[9] D.-S. Huang, "Radial basis probabilistic neural networks: Model and application," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 13, no. 7, pp. 1083–1101, 1999.

[10] C.-Y. Lu and D.-S. Huang, "Optimized projections for sparse representation based classification," *Neurocomputing*, vol. 113, pp. 213–219, Aug. 2013.

[11] L. Zhu and D.-S. Huang, "Efficient optimally regularized discriminant analysis," *Neurocomputing*, vol. 117, pp. 12–21, Oct. 2013.

[12] B. Li, C. Wang, and D.-S. Huang, "Supervised feature extraction based on orthogonal discriminant projection," *Neurocomputing*, vol. 73, nos. 1–3, pp. 191–196, 2009.

[13] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. 19, no. 6, pp. 716–723, Dec. 1974.

[14] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[15] D. Burshtein and E. Weinstein, "Some relations between the various criteria for autoregressive model order determination," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 4, pp. 1071–1079, Aug. 1985.

[16] K. Li, J.-X. Peng, and E.-W. Bai, "A two-stage algorithm for identification of nonlinear dynamic systems," *Automatica*, vol. 42, no. 7, pp. 1189–1197, 2006.

[17] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.

[18] N. B. Karayiannis, "Reformulated radial basis neural networks trained by gradient descent," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 657–671, May 1999.

[19] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[20] A. Miller, *Subset Selection in Regression*, 2nd ed. London, U.K.: Chapman & Hall, 2002.

[21] O. Nelles, *Nonlinear System Identification*. New York, NY, USA: Springer-Verlag, 2001.

[22] S. Chen, S. A. Billing, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *Int. J. Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[23] D.-S. Huang and W.-B. Zhao, "Determining the centers of radial basis probabilistic neural networks by recursive orthogonal least square algorithms," *Appl. Math. Comput.*, vol. 162, no. 1, pp. 461–473, 2005.

[24] K. Li, J.-X. Peng, and G. W. Irwin, "A fast nonlinear model identification method," *IEEE Trans. Autom. Control*, vol. 50, no. 8, pp. 1211–1216, Aug. 2005.

[25] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[26] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Conf. Rec. 27th Asilomar Conf. Signals, Syst., Comput.*, Nov. 1993, pp. 40–44.

[27] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Signal Process. Lett.*, vol. 9, no. 4, pp. 137–140, Apr. 2002.

[28] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 306–314, Mar. 2000.

[29] A. Sherstinsky and R. W. Picard, "On the efficiency of the orthogonal least squares training method for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 195–200, Jan. 1996.

[30] X. Hong and C. J. Harris, "Nonlinear model structure detection using optimum experimental design and orthogonal least squares," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 435–439, Mar. 2001.

[31] X. Hong and C. J. Harris, "Nonlinear model structure design and construction using orthogonal least squares and D-optimality design," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1245–1250, Sep. 2002.

[32] A. Atkinson, A. Donev, and R. Tobias, *Optimum Experimental Designs, With SAS*. New York, NY, USA: Oxford Univ. Press, 2007.

[33] X. Hong, R. J. Mitchell, S. Chen, C. J. Harris, K. Li, and G. W. Irwin, "Model selection approaches for non-linear system identification: A review," *Int. J. Syst. Sci.*, vol. 39, no. 10, pp. 925–946, 2008.

[34] S. Chen, E. S. Chng, and K. Alkadhimi, "Regularized orthogonal least squares algorithm for constructing radial basis funtion networks," *Int. J. Control*, vol. 64, no. 5, pp. 829–837, 1996.

[35] G. A. F. Seber and A. J. Lee, *Linear Regression Analysis*, 2nd ed. New York, NY, USA: Wiley, 2003.

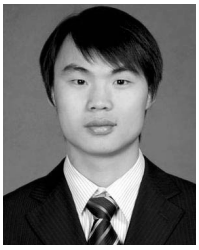
[36] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B (Methodol.)*, vol. 58, no. 1, pp. 267–288, 1996.

[37] S. Sardy, A. G. Bruce, and P. Tseng, "Block coordinate relaxation methods for nonparametric wavelet denoising," *J. Comput. Graph. Statist.*, vol. 9, no. 2, pp. 361–379, 2000.

[38] S. Perkins, K. Lacker, and J. Theiler, "Grafting: Fast, incremental feature selection by gradient descent in function space," *J. Mach. Learn. Res.*, vol. 3, pp. 1333–1356, Jan. 2003.

[39] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. Statist. Assoc.*, vol. 96, no. 456, pp. 1348–1360, 2001.

[40] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *Ann. Statist.*, vol. 35, no. 3, pp. 1012–1030, 2007.
[41] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.
[42] K. Z. Mao and S. A. Billings, "Algorithms for minimal model structure detection in nonlinear dynamic system identification," *Int. J. Control*, vol. 68, no. 2, pp. 311–330, 1997.
[43] S. Chen, X. Hong, and C. J. Harris, "Particle swarm optimization aided orthogonal forward regression for united data modeling," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 477–499, Aug. 2010.
[44] D.-S. Huang and J.-X. Du, "A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2099–2115, Dec. 2008.
[45] Y. Oussar and G. Dreyfus, "Initialization by selection for wavelet network training," *Neurocomputing*, vol. 34, nos. 1–4, pp. 131–143, 2000.
[46] Q. M. Zhu and S. A. Billings, "Fast orthogonal identification of nonlinear stochastic models and radial basis function neural networks," *Int. J. Control*, vol. 64, no. 5, pp. 871–886, 1996.
[47] J. R. Rice, "Experiments on Gram–Schmidt orthogonalization," *Math. Comput.*, vol. 20, no. 94, pp. 325–328, 1966.
[48] Å. Björck, "Solving linear least squares problems by Gram–Schmidt orthogonalization," *BIT Numer. Math.*, vol. 7, no. 1, pp. 1–21, 1967.
[49] J.-X. Peng, K. Li, and D.-S. Huang, "A hybrid forward algorithm for RBF neural network construction," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1439–1451, Nov. 2006.
[50] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 889–898, Nov. 1992.



Long Zhang (M'13) received the B.Eng. and M.Eng. degrees in electrical engineering and automation from the Harbin Institute of Technology, Harbin, China, in 2008 and 2010, respectively, and the Ph.D. degree in electronics, electrical engineering and computer science from Queen's University Belfast, Belfast, U.K., in 2013.

He is currently a Research Associate with the Department of Automatic Control and System Engineering, University of Sheffield, Sheffield, U.K. His current research interests include system identification, neural networks, statistical regression, and fault-diagnosis in both time and frequency domains.

Kang Li (M'05–SM'11) received the B.Sc. degree from Xiangtan University, Xiangtan, China, in 1989, the M.Sc. degree from the Harbin Institute of Technology, Harbin, China, in 1992, and the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 1995.

He is currently a Professor of Intelligent Systems and Control with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, U.K. He is involved in bioinformatics with applications on food safety and biomedical engineering. He is also a Visiting Professor with the Harbin Institute of Technology, Shanghai University, Shanghai, China, and the Ningbo Institute of Technology, Zhejiang University, Zhejiang, China. He held a Visiting Fellowship or Professorship with the National University of Singapore, Singapore, the University of Iowa, Iowa City, IA, USA, the New Jersey Institute of Technology, Newark, NJ, USA, Tsinghua University, Beijing, China, and the Technical University of Bari, Taranto, Italy. He has authored more than 200 papers in his areas of expertise, and edited 12 conference proceedings (Springer). His current research interests include nonlinear system modeling, identification and control, bio-inspired computational intelligence, and fault-diagnosis and detection, with recent applications in power systems and renewable energy, and polymer extrusion processes.

Dr. Li serves in the editorial boards of *Neurocomputing*, the *Transactions of the Institute of Measurement and Control*, *Cognitive Computation, and International Journal of Modelling, Identification and Control*.

Er-Wei Bai (F'03) received the B.Sc, M.Eng and Ph.D Degrees from Fudan University, Shanghai Jiaotong University, both in Shanghai, China, and the University of California at Berkeley, respectively.

He is currently a Professor of Electrical and Computer Engineering, and Radiology with the University of Iowa, Iowa City, IA, USA, where he is involved in identification, control, signal processing and their applications in engineering, and life science. He also holds the rank of World Class Research Professor with Queen's University Belfast, Belfast, U.K.

Dr. Bai was a recipient of the President's Award for Teaching Excellence and the Board of Regents Award for Faculty Excellence.

George W. Irwin (F'04) received the B.Sc degree (Hons.) in electrical and electronic engineering, in 1972, the Ph.D. degree in control theory, in 1976, and the D.Sc. degree, in 1998, all from Queen's University Belfast, Belfast, U.K.

He has held the Personal Chair in control engineering with Queen's University Belfast since 1989. He has authored more than 350 research publications, including 125 peer-reviewed journal papers. His current research interests include identification, monitoring, and control, including neural networks, fuzzy neural systems, and multivariate statistics.

Prof. Irwin was a fellow of the U.K. Royal Academy of Engineering in 2002, and a member of the Royal Irish Academy in 2002. He holds Fellowships of the IEEE in 2004 and International Federation of Accountants (IFAC) in 2009. He was a Editor-in-Chief of *Control Engineering Practice*. He serves on the Technical Board and the Publications Management Board of IFAC. He received four IEE Premiums and two medals from the U.K. Institute of Measurement and Control.