A new adaptive multiple modeling approach for non-linear and non-stationary systems

Hao Chen, *Student Member, IEEE,* Xia Hong, *Senior Member, IEEE* and Yu Gong, *Member, IEEE,*

Abstract

This paper proposes a novel adaptive multiple modeling algorithm for non-linear and non-stationary systems. This simple modeling paradigm comprises K candidate sub-models which are all linear. With data available in an online fashion, the performance of all candidate sub-models are monitored based on the most recent data window, and M best sub-models are selected from the K candidates. The weight coefficients of the selected sub-model are adapted via the recursive least square (RLS) algorithm, while the coefficients of the un-selected sub-models remain unchanged. These M model predictions are then optimally combined to produce the multi-model output. We propose to minimize the mean square error based on a recent data window, and apply the sum to one constraint to the combination parameters, leading to a close form solution so that maximal computational efficiency can be achieved. In addition at each time step the model prediction is chosen from either the resultant multiple model or the best sub-model, whichever is the best. Simulation results are given in comparison with some typical alternatives including the linear RLS algorithm and a number of on-line non-linear approaches, in terms of modeling performance and time consumption.

Index Terms

On-line modeling, non-stationary, non-linear, multiple model.

This research is sponsored by the UK Engineering and Physical Sciences Research Council and DSTL under the grant number EP/H012516/1.

Hao Chen and Xia Hong are with the School of Systems Engineering, University of Reading, UK (E-mail: hao.chen@pgr.reading.ac.uk, x.hong@reading.ac.uk).

Yu Gong is with the School of Electronic, Electrical and Systems Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK, E-mail: y.gong@lboro.ac.uk.

I. Introduction

Many systems exhibit both non-linear and non-stationarity properties. In order to capture the non-linear properties of a system from data, model representations using non-linear basis functions are often used. A typical example is the popular radial basis function (RBF) neural network. With a cluster of non-linear "kernels" (or nodes) imposed on the input data, the RBF network can approximate any continuous function to an arbitrary degree of accuracy [1]. On the other hand, the non-stationarity refers to the fact the a system is slowly changing in real time operations. On-line modeling which keeps updating with the incoming data is of great importance for non-stationary systems. For a linear system, a common approach is to use adaptive algorithms such as the least mean square (LMS) and the recursive least squares (RLS) algorithms to track the temporal variation of the systems [2], [3]. By constantly updating the output layer weights, the RLS and LMS are directly applicable to a RBF model due to its linear-in-the-parameters structure.

In many applications with strong non-stationarity, the weight adaptation alone may be insufficient and often the structure of the RBF network should also be updated on-line. This can be seen in the well-known resource allocating network (RAN) [4], where the network model starts from empty and grows with the input data based on the *nearest neighbour* method. Many improvements have been proposed since then, where a typical example is the GAP-RBF algorithm in which the model size can both grow and prune with the input data [5].

Alternatively, following the well-known fact that a non-linear curve can be approximated with piecewise linear functions, non-linear modeling can also be realized with multiple linear models. The early work of this kind includes Tong's threshold autoregressive (TAR) model for non-linear time series analysis, in which the overall non-linear time series is split into several "regimes" and one or more linear models are then chosen from a group of pre-set linear AR models for each of the regimes [6], [7]. There are generally two kinds of switching rules to determine the linear models for each regime. In the first kind, the switching indicator is observed based on the input data. For example, in the self-exciting threshold autoregressive (SETAR) model, the modeling switches among different pre-set AR models according to the current input [6]. On the other hand, the switching indicator may also be hidden such that it does not depend on the input data. This can be seen in the exponential AR (EAR) model in which the switching indicator is a sequence of independently and identically distributed random variables which form a hidden finite state Markov chain [7].

The TAR modeling approach provides an attractive way of modeling non-linear systems due to the associated low complexity since only piecewise linear models are involved. The data driven model based adaptive control [**?**] is closely related to the modeling of nonlinear time series. Using multiple piecewise linear models leads to the successful gain scheduling technique where a bank of linear controllers are used to control a nonlinear system. In the similar spirit, the operating point dependent locally linear model representation has been introduced that lends to use of linear control methods with provable convergence [**?**]. Switching based multiple linear models and control have been well researched with provable convergence. Two kinds of switching schemes have been proposed for multiple models: In the first class, called direct switching [8]–[12], the switching rule is based on the output of the system which was predetermined. In the second class, known as indirect switching schemes [13]–[15], the switching activities depend upon the multiple models at every instant and were found to be more attractive than the direct switching for practical applications [16], [17]. The candidate multiple models using fixed models [18], [19] or combinations of fixed and adaptive models [20], [21] were well studied. Currently, switching mechanism with neural networks sub-models was proposed to improve the accuracy of nonlinear systems estimation [22]. However, the performance of all these switching based approaches well depends on the pre-set candidate models and switching rules which require substantial a-priori knowledge of the systems. This makes it inflexible in highly non-stationary systems where the "a-priori knowledge" may keep changing over time. The switching model also has difficulty in handling systems containing heavily continuous non-linear portions [23].

The aforementioned switching based modeling approaches select a single candidate model at every instant. Clearly better modeling performance can be achieved with an optimally combined candidate models. In machine learning community, there are consideridable researches concerning producing superior modeling performance via a bank of weak/base learners. Representatives ensemble learning are classification and regression trees (CART), multivariate adaptive regression splines (MARS), mixture of experts (MoE) etc [24]–[27]. Cherkassky et al. [28], [29] proposed to partition available training data into several subsets and to estimate the corresponding models based on the double application of support vector machine (SVM) approach for multiple model regression estimation and classification (MME and MMC). Other techniques based on the concept of data resampling include bagging and boosting have also attract attentions primarily in pattern recognition tasks [30]–[32]. However most of the approaches the superior performance come with the associated high computational costs, so they are not practical to be applied as online learning algorithms for nonstationary systems, in particular real time signal processing and adaptive control applications.

In this paper, we propose a novel adaptive multiple modeling approach. The proposed approach has a two-level structure. The lower level is composed of a bank of K linear candidate sub-models which can track the system output by updating the associated parameters independently. The higher level uses a linear combinator to combine a subset of the M best sub-models which are selected from the K candidate sub-models based on the recent modeling performance. At each time step, the parameters of the selected linear sub-models are updated using the recursive-least-square (RLS) algorithms at the lower level; the linear combining coefficients are obtained at higher level by minimizing the norm of recent modeling errors of the combined model, subject to the constraint that combining coefficients sum to one. The proposed combinator scheme is advantageous since it leads to a closed form solution for maximum computational efficiency. Additionally the modeling performance of the higher level is monitored in comparison with the best sub-linear model, and the better model prediction is selected as the final model output.

The proposed multiple modeling approach retains the low-complexity advantage of the traditional multiple TAR models [6], [7] as all sub-models are linear, making it particularly suitable for on-line applications. On the other hand, the proposed approach does not depend on the a-priori information of the systems, because the candidate sub-models keep adapting with the new input data. Moreover, the proposed approach can also well handle the heavily continuous non-linearity with M sub-models being optimally combined to give the multiple modeling output. Simulations show that the proposed adaptive multiple modeling approach not only has significantly superior performance to the linear adaptive approaches, but also outperforms on-line RBF approaches such as the RAN [4] and GAP-RBF algorithms [5]. It provides a fast yet accurate way for real time non-linear and non-stationary system modeling.

II. Adaptive multiple linear modeling

A discrete time non-linear system can be represented by an NARX model as:

$$y(t) = F_t(\mathbf{x}(t)) + \xi(t), \tag{1}$$

where F_t is the model function representing the input/ouput relationship at time t, $\mathbf{x}(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^{\mathrm{T}} = [y(t-1), \cdots, y(t-n_y), u(t-1), \cdots, u(t-n_u)]^{\mathrm{T}}$ is the model input vector with dimension of $n = n_y + n_u$, $u(t)$ and $y(t)$ are the system input and output at time t respectively, n_y and n_u are the input and output lags respectively, and $\xi(t)$ is the uncorrelated observation noise with mean zero and variance σ^2. The object of the on-line modeling is, given the observation data set $\{\mathbf{x}(t), y(t)\}_{t=1}^{N}$, to construct an estimator to approximate the underlying dynamic $F_t(.)$ at every sampling time t.

In this paper, an adaptive multiple linear model is described to estimate the non-linear system on-line. We assume without losing generality that there are K candidate sub-models which are all linear. At time t, the output of the kth sub-model is given by

$$\begin{aligned} \hat{y}^{(k)}(t) &= \theta_1^{(k)}(t) + \sum_{i=2}^{n+1} \theta_i^{(k)}(t) x_i(t) \\ &= \boldsymbol{\varphi}^{\mathrm{T}}(t)\boldsymbol{\theta}^{(k)}(t), \qquad k = 1, ...K \end{aligned} \tag{2}$$

where $\boldsymbol{\theta}^{(k)}(t) = \left[\theta_1^{(k)}(t), \cdots, \theta_{n+1}^{(k)}(t)\right]^{\mathrm{T}}$, $\theta_i^{(k)}(t)$ is the ith coefficient for the kth model at time t, $\boldsymbol{\varphi}(t) = [1, \mathbf{x}^{\mathrm{T}}(t)]^{\mathrm{T}}$ which is the input vector plus a DC offset 1.

The performance for every of the K candidate sub-models is constantly monitored for every input data. In order to avoid the random disturbance from a single input, the performance is measured based on the last p inputs. To be specific, at every time step, the performance index for the k-th sub-model is calculated as

$$J^{(k)}(t) = \|\mathbf{e}^{(k)}(t)\|^2, \qquad k = 1, ..., K \tag{3}$$

where $\mathbf{e}^{(k)}(t) = [e^{(k)}(t), \cdots, e^{(k)}(t-p+1)]^{\mathrm{T}}$, and $e^{(k)}(t)$ is the the modeling error of the k-th sub-model at time t which is given by

$$e^{(k)}(t) = y(t) - \hat{y}^{(k)}(t). \tag{4}$$

At every time t, a fixed number of M ($M \leq K$) sub-models with the smallest $J_k(t)$ are selected from the K candidates. We assume without losing generality that the j-th selected sub-model is the k_j-th candidate sub-model, where $j = 1, \cdots, M$ and $k_j \in \{1, \cdots, K\}$. As is shown in Fig. 1-(a), the multi-modeling output is the weighted summation of the M selected sub-models which is given by

$$\hat{y}(t) = \sum_{j=1}^{M} \omega^{(j)}(t) \hat{y}^{(k_j)}(t) \tag{5}$$

where $\omega^{(j)}(t)$ is the combining coefficient for the j-th selected sub-model.

We propose that the combining coefficients are constrained as $\sum_{j=1}^{M} \omega^{(j)}(t) = 1$. Thus the optimum combining coefficients at time t minimize the least squares cost function as

$$
\begin{aligned}
& V(t) = \frac{1}{2}\sum_{i=0}^{p-1} e^2(t-i) = \frac{1}{2}\sum_{i=0}^{p-1}[y(t-i) - \hat{y}(t-i|t)]^2, \\
& \text{s.t.} \sum_{j=1}^{M} \omega^{(j)}(t) = 1
\end{aligned} \tag{6}
$$

where $\hat{y}(t-i|t) = \sum_{j=1}^{M} \omega^{(j)}(t)\hat{y}^{(k_j)}(t-i)$ and $p \geq 1$. We point out that the cost function $V(t)$ is minimized with respect to the current combining vector $\omega^{(j)}(t)$ only.

III. Multiple model optimization

In this section, we present how the proposed multiple linear modeling is optimized on line. In Section III-A, the RLS algorithm is applied as the algorithm for updating M sub-models selected at each time step; In Section III-B, we derive the analytical formula for optimal combining coefficients, and then propose to use the combined output in competition with the best sub-model output in order to achieve best available modeling performance. Finally in Section III-C, the important aspect about the sub-model initialization is discussed.

A. Sub-model adaptation

The coefficients for each of the selected sub-models are adapted with the RLS algorithm based on the corresponding error signal $e^{(j)}(t)$ respectively, as is shown in Fig. 1-(a). To be specific, the coefficient for the j-th selected sub-model $\boldsymbol{\theta}^{(j)}(t)$ are adapted as

$$
\begin{aligned}
& \boldsymbol{\Psi}^{(j)}(t) = \mathbf{P}^{(j)}(t-1)\boldsymbol{\varphi}(t)\left[\lambda + \boldsymbol{\varphi}^{\mathrm{T}}(t)\mathbf{P}^{(j)}(t-1)\boldsymbol{\varphi}(t)\right]^{-1} \\
& \mathbf{P}^{(j)}(t) = \left[\mathbf{P}^{(j)}(t-1) - \boldsymbol{\Psi}^{(j)}(t)\boldsymbol{\varphi}^{\mathrm{T}}(t)\mathbf{P}^{(j)}(t-1)\right]\lambda^{-1} \\
& \boldsymbol{\theta}^{(j)}(t) = \boldsymbol{\theta}^{(j)}(t-1) + e^{(j)}(t)\boldsymbol{\Psi}^{(j)}(t)
\end{aligned} \tag{7}
$$

where $0.9 < \lambda \leq 1$ is the forgetting factor, $\mathbf{P}^{(j)}(t)$ is the covariance matrix of the j-th sub-model which is often initialized as $\mathbf{P}^{(j)}(0) = \delta\mathbf{I}$, δ is a large constant and $\mathbf{I}$ is the identity matrix with appropriate dimension.

For the unselected candidate sub-models, the performance is constantly monitored but the coefficients remained unchanged. Thus the computation complexity is kept at a low level since only the selected sub-models are adapted with time. More importantly, this also makes the overall candidate sub-models have good data coverage. Since an unselected sub-model remains unchanged, it fits into some "previous" data rather than the current input. This makes sure that, if the data falls back into the old range, this unselected sub-model will be activated again. As a result, each sub-model only takes care of one part of the overall data range, which is actually the basic point in applying the multiple model.

B. Fast determining combining coefficients

When the M best sub-models are selected, they are weightily combined to produce the multi-modeling output. The optimum combining coefficients are obtained by minimizing the cost function in (6). Because of the constraint $\sum_{j=1}^{M} \omega^{(j)}(t) = 1$, (6) can be expressed as:

$$\begin{aligned} V(t) &= \frac{1}{2}\sum_{i=0}^{p-1}\left[\sum_{j=1}^{M}\omega^{(j)}(t)y(t-i) - \sum_{j=1}^{M}\omega^{(j)}(t)\hat{y}^{(k_j)}(t-i)\right]^2 \\ &= \frac{1}{2}\sum_{i=0}^{p-1}\left[\sum_{j=1}^{M}(\omega^{(j)}(t)e^{(j)}(t-i))\right]^2 \\ &= \frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}(t)\mathbf{C}(t)\boldsymbol{\omega}(t) \end{aligned} \tag{8}$$

where $e^{(j)}(t-i) = y(t-i) - \hat{y}^{(k_j)}(t-i|t)$, $\boldsymbol{\omega}(t) = [\omega^{(1)}(t), \cdots, \omega^{(M)}(t)]^{\mathrm{T}}$ which is the combining vector, and $\mathbf{C}(t)$ is the error covariance matrix which is given by

$$\begin{aligned} \mathbf{C}(t) &= \sum_{i=0}^{p-1}\mathbf{e}(t-i)\mathbf{e}^{\mathrm{T}}(t-i) \in \Re^{M\times M} \qquad (9) \\ &= \sum_{i=0}^{p-1}\begin{bmatrix} |e^{(1)}(t-i)|^2 & \cdots & e^{(M)}(t-i)e^{(1)}(t-i) \\ e^{(1)}(t-i)e^{(2)}(t-i) & \cdots & e^{(M)}(t-i)e^{(2)}(t-i) \\ \cdots & \cdots & \cdots \\ e^{(1)}(t-i)e^{(M)}(t-i) & \cdots & |e^{(M)}(t-i)|^2 \end{bmatrix} \end{aligned}$$

where $\mathbf{e}(t-i) = [e^{(1)}(t-i), \cdots, e^{(M)}(t-i)]^{\mathrm{T}}$.

The Lagrange function for the constrained minimization problem in (6) is thus given by

$$L(\boldsymbol{\omega}(t)) = \frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}(t)\mathbf{C}(t)\boldsymbol{\omega}(t) + \gamma[\mathbf{1}^{\mathrm{T}}\boldsymbol{\omega}(t) - 1], \tag{10}$$

where $\gamma > 0$ is Lagrange multiplier, and $\mathbf{1} = [1, ..., 1]^{\mathrm{T}}$.

Letting $\frac{\partial}{\partial \omega(t)} L = \mathbf{0}$ yields

$$\mathbf{C}(t)\boldsymbol{\omega}(t) + \gamma \mathbf{1} = 0 \tag{11}$$

This suggests that the optimum combining vector is obtained

$$\boldsymbol{\omega}(t) = \mathbf{C}^{-1}(t) \cdot \mathbf{1} \tag{12}$$

divided by its own sum $\sum_{j=1}^{M} \omega^{(j)}(t)$. It is clear the resultant vector satisfies the constraint $\sum_{j=1}^{M} \omega^{(j)}(t) = 1$.

We particularly highlight that this multiple model output may not always be better (though it often is) than that from a single sub-model. This is because we only let $\sum_{j=1}^{M} \omega^{(j)}(t) = 1$ without any positive constraint on individual combining coefficients. While this has the advantage of leading to the closed-form solution of the combining vector as in (12), it may yield some negative $w^{(j)}(t)$ which corresponds to "negative" contribution to the multi-modeling. Therefore, as is shown in Fig. 1-(b), it is necessary to compare the output for the multi-model and those for every individual sub-models, and choose best one as the final output. Specifically, the final modeling output is given by

$$\begin{cases} \text{If } \sum_{i=0}^{p-1} |e(t-i)|^2 < \sum_{i=0}^{p-1} |e^{(k)}(t-i)|^2 \text{ for any } k \\ \qquad \hat{z}(t) = \sum_{j=1}^{M} \omega^{(j)}(t) \hat{y}^{(k_j)}(t), \\ \text{Otherwise} \\ \qquad \hat{z}(t) = \hat{y}^{(k_q)}(t), \end{cases} \tag{13}$$

where $\hat{y}^{(k_q)}(t)$ is the output of the k_q-th sub-model which has the smallest residual error. Thus the final modeling error is given as

$$\epsilon(t) = y(t) - \hat{z}(t). \tag{14}$$

C. Sub-model initialization

The performance of the multi-model system well depends on the coverage of the candidate sub-models. Ideally the candidate sub-models should seamlessly and evenly cover the overall dynamic range of the input data, so that different sub-models are selected for different input range. This requires careful initialization of the candidate sub-models. If, for example, the coefficients of all candidate sub-models are simply initialized to zero as in many adaptive filters, the initially selected sub-models will have the same coefficient vector as they follow the same RLS adaptation. Moreover, the unselected sub-models will have no chance to be selected at any later time because they have zero output. Then the multi-model system reduces to a single linear model. On the other hand, if the coefficients of the sub-models are randomly initialized, it is also highly possible that some candidate sub-models are not selected at any time. As a result, some dynamic range of the input data may not be well covered, leading to significant performance degradation.

In practice, even if the overall dynamic range of the input data is known, it is still not straightforward to initialize the coefficients so that each sub-model covers one part of the overall dynamic range. In many applications, moreover, the input dynamic range is often neither a priori known nor easily obtained. An adaptive initialization scheme is thus described below.

At first, we only let the coefficients of the first candidate sub-model be trained by the RLS algorithm. After the coefficients converge which is usually within several iterations for the RLS algorithms, the first sub-model roughly covers the input dynamic range. The other candidate sub-models can then be randomly initialized around the pre-trained sub-model. Because the poles and zeros of the transfer function describe the intrinsic properties of a dynamic system [33], the initialization of other sub-models are through poles and zeros of their transfer functions, rather than directly from the coefficients.

As is shown in (2), the sub-model input includes both data and a constant DC offset 1. After removing the DC offset, the pulse transfer function of the k-th sub-model is given by

$$\begin{aligned} H^{(k)}(z) &= \frac{\theta^{(k)}_{n_y+2} + \theta^{(k)}_{n_y+3} \cdot z^{-1} + \cdots + \theta^{(k)}_{n_y+n_u+1} \cdot z^{-n_u+1}}{1 - \theta^{(k)}_{2} \cdot z^{-1} - \cdots - \theta^{(k)}_{n_y+1} \cdot z^{-n_y}} \\ &= \frac{\prod_{m=0}^{n_u-1}\left(z - O^{(k)}_m\right)}{\prod_{l=1}^{n_y}\left(z - P^{(k)}_l\right)}, \quad k = 1, \cdots, K. \end{aligned} \tag{15}$$

where z is the shift operator, $P^{(k)}_l$ and $O^{(k)}_m$ are the l-th and m-th pole and zero for the k-th sub-model respectively. It is clear that $P^{(k)}_l$ and $O^{(k)}_m$, together with the coefficient for the DC offset $\theta^{(k)}_1$, determine the k-th sub-model.

With these observation, we have the following initialization procedure:

1) At the beginning, train the first candidate sub-model with $\bar{N}$ numbers of data using the RLS algorithm.
2) Obtain the the poles $P^{(1)}_l$ and zeros $O^{(1)}_m$ of the transfer function (after removing the DC offset) for the pre-trained sub-model, and find the DC coefficient $\theta^{(1)}_1$.

3) The l-th and m-th pole and zero of the k-th ($k \neq 1$) sub-model are randomly initialized as

$$\begin{aligned} P_l^{(k)} &= P_l^{(1)} \cdot N(1, \sigma_1), \\ O_m^{(k)} &= O_m^{(1)} \cdot N(1, \sigma_1), \qquad k = 2, ..., K \end{aligned} \tag{16}$$

respectively, where $N(a, b)$ represents one random realization of the Gaussian process with mean a and standard-deviation of b.

4) The DC coefficient of the k-th ($k \neq 1$) sub-model is randomly initialized as

$$\theta_1^{(k)} = \theta_1^{(1)} \cdot N(1, \sigma_2), \qquad k = 2, ..., K \tag{17}$$

5) From (16) and (17), we can easily set the initial weight coefficients for all candidate sub-models.

The aforementioned initialization process requires little a priori information of the input data, and can automatically initialize the candidate sub-models with good data coverage.

D. The algorithm summary

The flow-chart of the proposed adaptive multi-model algorithm is shown in Fig. 2. In general, it consists of two phases: *initialization phase* and *on-line learning phase*.

In the initialization phase, only the first sub-model is trained by the RLS algorithm. After the training, other $(K-1)$ sub-models are randomly initialized around the pre-trained sub-model. Extensive simulations show that more than 5 training samples are necessary to pre-train the first sub-model.

In the learning phase, for every new input data, the performance of all candidate sub-models are monitored. The best M sub-models are selected and weightily combined to give the multi-modeling output. Only the weight coefficients of the selected sub-models are adapted with the RLS and those for unselected sub-models remain unchanged. Finally, the output of the multi-model and the best of the candidate sub-models are compared, and whichever the better result is used as the final model prediction.

It is clear that the proposed model has a two level structure, each of them being essentially linear. The motivation is to enable the applicability for the efficient linear learning algorithm to nonlinear and nonstationary systems, which can be locally linearly approximated. In the proposed algorithm, the parameters M,K and p are predetermined. Particularly p represents the length of the recent data window for evaluating the modeling performance in selecting sub-models, as well as computing combining coefficients. For nonstationary systems, the system dynamics are changeable therefore old data should be forgotten so the model parameters can be adaptive to the current data. However a sufficient large p is also needed in order to make the parameter variance as small as possible. We use K candidate linear models in order to cope with nonlinear systems, since this provides modeling capability if these linear models are significantly diverse. The strategy of selecting M out of K sub-models is useful for maintaining the diversity of these candidate linear models, hence the nonlinear approximation capability of the proposed approach.

IV. SIMULATION

In this section, computer simulations are given to compare the proposed adaptive multi-model algorithm with some typical on-line modeling approaches including the linear RLS algorithm [34], and the non-linear RBF based RAN [4] and GAP-RBF [5] algorithms, where both the RAN and GAP-RBF apply Gaussian nodes.

For the proposed multi-model approach, the number of total candidate sub-models is set as $K = 10$, at every time step the number of selected sub-models M is set as either 3 or 5, the number of data input used to pre-train the first sub-model is set as $\bar{N} = 50$, and the number of recent errors in the least function cost function (6) is set as $p = 3$. For all other on-line approaches in the simulation, the parameters are carefully chosen based trial-and-error to achieve the best possible performance.

In this section, the algorithms are compared in the application of on-line time series prediction. To be specific, the T-step ahead prediction is to use the past four samples

$$\mathbf{x}(t) = [y(t),\ y(t-6),\ y(t-12),\ y(t-18)]^{\mathrm{T}} \tag{18}$$

to estimate the future sample $y(t+T)$. The prediction performance is measured by the root mean square error (RMSE). At time t, the RMSE is defined as

$$\mathrm{RMSE}(t) = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(y(i) - \hat{z}(i)))^2} \tag{19}$$

Two benchmark chaotic time series are considered: Mackey-Glass and Lorenz time series. In order to fully verify the proposed approach, both stationary and non-stationary cases are considered. In the stationary case, the parameters controlling the chaotic time series behavior are fixed. While in the non-stationary case, these controlling parameters are changing either abruptly (piecewise function) or continuously. An additive white Gaussian noise with mean zero and variance σ^2 is used in the data to further validate the robustness and stability of the proposed method.

The simulations are carried out by the MATLAB 7.14 (R2012a) running on an Intel Core i7 2600K, 4.3 GHz processor.

A. Mackey-Glass time series

The Mackey-Glass time series is generated from the differential delay equation as

$$\frac{dx(t)}{dt} = \frac{ax(t-c)}{1+x^{10}(t-c)}) - bx(t) \tag{20}$$

where a, b and c are controlling parameters, and especially when $c \geq 17$ the equation shows typical chaotic behavior. In this simulation, 5000 samples are generated by the 4th order Runge-Kutta method with the step size of 1, and the last 3000 samples are used for the prediction. The forward prediction step is set as $T = 60$.

1) Mackey-Glass time series with fixed parameters: First, we let $a = 0.2$, $b = 0.1$ and $c = 30$. Table I and Fig. 3 compare the final prediction performance and RMSE learning curves for different approaches respectively. In Table I, the values with brackets are the results in noise environment with noise variance as $\sigma^2 = 0.01$. The time consumption for learning without noise is also shown in Table I[1]. It is clear that the linear RLS has the worst performance because a single linear model can not model the non-linear system. The RAN and GAP-RBF have comparable performance, while the GAP-RBF has far fewer number of nodes than the RAN. This is because the GAP-RBF can prune the model size and the RAN can not. It is observed that, when the noise is present, the model size for both the RAN and GAP-RBF approaches increase significantly. In contrast, the proposed approach with the same model size has similar performance and thus it is robust against the noise. In general, the proposed algorithm with only 3 or 5 selected linear models not only has better performance than both the linear and non-linear approaches, but also is very fast with the time consumption comparable to that of the linear RLS. In fact, as is also shown in other simulations, the time consumption for the proposed approach with M selected nodes is approximated M times of that for the linear RLS algorithm.

[1]The noise has little affect on the time consumption.

2) Mackey-Glass time series with piecewise function: In this simulation, the above Mackey-Glass time series are weighted by the piecewise function as

$$y_1(t) = y(t) \cdot \Psi(t), \tag{21}$$

where

$$\Psi(t) = \begin{cases} 0.0005 \cdot t^{\frac{1}{0.98}} & 0 \leq t \leq 999 \\ 0.0006 \cdot t^{\frac{1}{0.95}} & 1,000 \leq t \leq 1,999 \\ 0.0003 \cdot t^{\frac{1}{0.97}} & 2,000 \leq t \leq 2,999 \end{cases} \tag{22}$$

$y_1(t)$ is used for the time series prediction. As is shown in Fig. 4, $y_1(t)$ is clearly non-stationary as it has different dynamic range at different time intervals.

Table II compare the final prediction performance in this non-stationary case for different approaches. Comparing these results with those in the previous simulation for the stationary case, we can observe that, while all approaches have worse prediction performance than those in the stationary case, the comparison between different approaches are similar. It is clear that the proposed multiple model still has the best performance. It is interesting to observe that both the non-linear RAN and GAP-RBF algorithms have very large model size.

Fig. 4 shows the on-line prediction performance of the proposed algorithm with 3 selected linear sub-models, where it is clearly shown that the proposed algorithm well predicts this non-stationary time series on-line.

B. Lorenz time series

The Lorenz chaotic time series prediction is another widely used benchmark application [35]. As a three dimensional and highly non-linear system, the Lorenz system is governed by three differential equations as

$$\begin{aligned} \frac{dx(t)}{dt} &= ay(t) - ax(t) \\ \frac{dy(t)}{dt} &= cx(t) - x(t)z(t) - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t) \end{aligned}$$

where a, b and c are parameters that control the behavior of the Lorenz system. In the simulations, the 4th order Runge-Kutta approach with the step size of 0.01 is used to generate the Lorenz samples, and only the Y-dimension samples, $y(t)$, are used for the time series prediction. For the 5000 data samples generated for $y(t)$, the last 3000 stable samples are used for the prediction since Lorenz system is very sensitive to the initial condition. The prediction step is set as $T = 50$.

1) Lorenz time series with fixed parameters: We first consider the Lorenz time series with fixed parameters as $a = 10$, $b = 8/3$ and $c = 28$. The trajectory of this Lorenz system is shown in Fig. 5.

Table III and Fig. 6 compare the final prediction performance and the RMSE learning curves for different approaches respectively. In Table III, the values with brackets are the results in noise environment with noise variance as $\sigma^2 = 0.1$. It is clear that the linear RLS has the worst prediction performance. The RAN and GAP-RBF have comparable prediction performance, and their model sizes increase clearly with noise. The proposed multi-model approach has the best performance and yet is still very fast.

2) Lorenz time series with time base drift: In this simulation, the parameters of the Lorenz systems are fixed as $a = 10$, $b = 8/3$, $c = 28$, but the samples of $y(t)$ are weighted by an exponential time based drift to obtain

$$y_2(t) = 1.1^{0.01t} \cdot y(t), \tag{23}$$

and $y_2(t)$ is used for the time series prediction.

Fig. 7 shows the on-line prediction performance of the proposed approach with 3 selected sub-models. It is shown that $y_2(t)$ is even more non-stationary than the time series in Fig. 4, where the dynamic range of $y_2(t)$ goes from around [-20,20] initially to about [-2000,2000] in the end. The proposed approach clearly well tracks this highly non-stationary Lorenz time series.

Table IV compare the final prediction performance for different approaches. It is clearly shown that the proposed multiple model has significantly better performance than the others. In fact, the multiple model is not only effectively the only approach here that can well track this highly non-stationary Lorenz time series, but also requires little time consumption.

V. Conclusion

In this paper, we have introduced a simple and effective approach aimed at the modeling of non-linear and non-stationary systems using multiple linear models. Each of the linear sub-models is able to track the system output independently. At each time and based on modeling performance over a short period, we select a subset of linear models, whose parameters are adapted using the RLS. Their outputs are fused using a set of linear combining coefficients. We propose the sum to one constraint and minimize the mean square errors over the recent data window in order to achieve a closed form solution for the combining coefficients. Since the multiple model output cannot be guaranteed to be better than that of the sub-linear models, the final model prediction is chosen from either the resultant multiple model or the best sub-model, whichever is the better. Numerical examples have been used to verify the proposed novel approach in comparison with the RLS and a number of nonlinear modeling methods. Because the proposed modeling algorithm is extremely fast without any compromise on modeling performance, it can be potentially utilized in many real time signal processing and adaptive control applications with nonlinear and non-stationary properties.

References

[1] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, Summer 1991.

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

[3] M. Birgmeier, "A fully kalman-trained radial basis function network for nonlinear speech modeling," in *Neural Networks, 1995. Proceedings, IEEE International Conference on*, vol. 1, Nov./Dec. 1995, pp. 259 –264.

[4] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, Summer 1991.

[5] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 6, pp. 2284 –2292, Dec. 2004.

[6] H. Tong and K. S. Lim, "Threshold autoregression, limit cycles and cyclical data," *Journal of the Royal Statistical Society, Series B*, vol. 42, no. 3, pp. 245–292, 1980.

[7] H. Tong, *Threshold models in non-linear time series analysis*, ser. Lecture notes in statistics. Springer-Verlag, 1983.

[8] B. M. artensson, "Adaptive stabilization," Ph.D. dissertation, Lund Insti-tute of Technology, Lund, Sweden, Tech. Rep., 1986.

[9] M. Fu and B. Barmish, "Adaptive stabilization of linear systems via switching control," *Automatic Control, IEEE Transactions on*, vol. 31, no. 12, pp. 1097–1103, Dec 1986.

[10] K. Poolla and S. J. Cusumano, "A new approach to adaptive robust controlparts i and ii," Coordinated Science Laboratory, Univ. Illinois, Urbana, Tech. Rep., 1988.

[11] D. E. Miller and E. Davison, "An adaptive controller which provides lyapunov stability," *Automatic Control, IEEE Transactions on*, vol. 34, no. 6, pp. 599–609, Jun 1989.

[12] D. E. Miller, "Adaptive stabilization using a nonlinear time-varying controller," *Automatic Control, IEEE Transactions on*, vol. 39, no. 7, pp. 1347–1359, Jul 1994.

[13] R. Middleton, G. Goodwin, D. Hill, and D. Mayne, "Design issues in adaptive control," *Automatic Control, IEEE Transactions on*, vol. 33, no. 1, pp. 50–58, Jan. 1988.

[14] A. Morse, D. Mayne, and G. Goodwin, “Applications of hysteresis switching in parameter adaptive control,” *Automatic Control, IEEE Transactions on*, vol. 37, no. 9, pp. 1343–1354, Sep 1992.

[15] S. Weller and G. Goodwin, “Hysteresis switching adaptive control of linear multivariable systems,” *Automatic Control, IEEE Transactions on*, vol. 39, no. 7, pp. 1360–1375, Jul 1994.

[16] K. Narendra and J. Balakrishnan, “Adaptive control using multiple models,” *Automatic Control, IEEE Transactions on*, vol. 42, no. 2, pp. 171–187, Feb 1997.

[17] K. Narendra and C. Xiang, “Adaptive control of discrete-time systems using multiple models,” *Automatic Control, IEEE Transactions on*, vol. 45, no. 9, pp. 1669–1686, Sep 2000.

[18] A. Morse, “Supervisory control of families of linear set-point controllers part i. exact matching,” *Automatic Control, IEEE Transactions on*, vol. 41, no. 10, pp. 1413–1431, Oct 1996.

[19] ——, “Supervisory control of families of linear set-point controllers. 2. robustness,” *Automatic Control, IEEE Transactions on*, vol. 42, no. 11, pp. 1500–1515, Nov 1997.

[20] K. Narendra, J. Balakrishnan, and M. Ciliz, “Adaptation and learning using multiple models, switching, and tuning,” *Control Systems, IEEE*, vol. 15, no. 3, pp. 37–51, Jun 1995.

[21] K. Narendra and J. Balakrishnan, “Adaptive control using multiple models,” *Automatic Control, IEEE Transactions on*, vol. 42, no. 2, pp. 171–187, Feb 1997.

[22] Y. Fu and T. Chai, “Intelligent decoupling control of nonlinear multivariable systems and its application to a wind tunnel system,” *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 6, pp. 1376–1384, Nov. 2009.

[23] P. Young, “Stochastic, dynamic modelling and signal processing: time variable and state dependent parameter estimation,” in *Cambridge University Press: Cambridge*. University Press, 2000, pp. 74–114.

[24] R. O. L. Breiman, J. Friedman and C. Stone, “Classification and regression trees,” Belmont, CA: Wadsworth, Tech. Rep., 1984.

[25] J. H. Friedman, “Multivariate adaptive regression splines,” *The Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[26] M. I. Jordan, “Hierarchical mixtures of experts and the em algorithm,” *Neural Computation*, vol. 6, pp. 181–214, 1994.

[27] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction, Second Edition*. Springer Series in Statistics, 2009.

[28] V. Cherkassky and Y. Ma, “Multiple model regression estimation,” *Neural Networks, IEEE Transactions on*, vol. 16, no. 4, pp. 785–798, July 2005.

[29] Y. Ma and V. Cherkassky, “Multiple model classification using svm-based approach,” in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 2, July 2003, pp. 1581–1586 vol.2.

[30] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

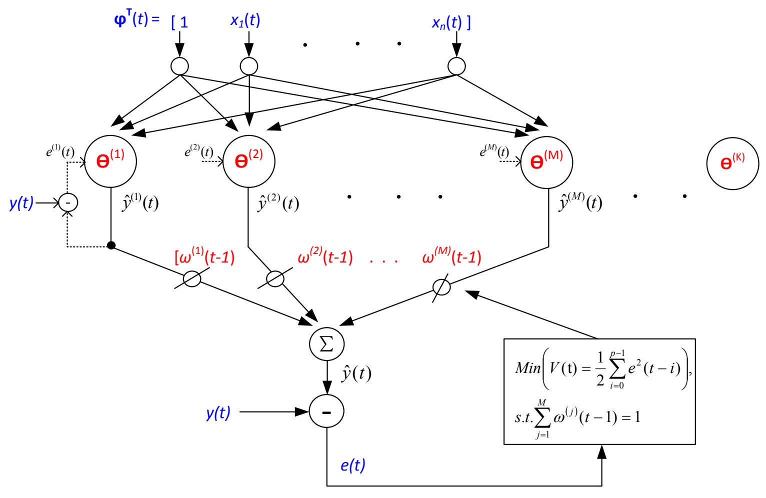
[31] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.

[32] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting,” *Annals of Statistics*, vol. 28, pp. 337–407, 2000.

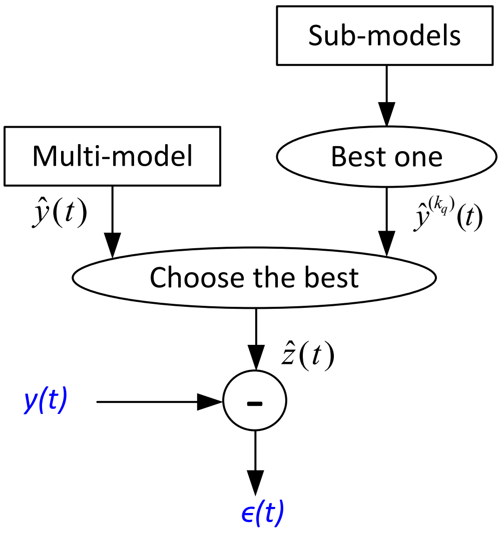
[33] K. Ogata, *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[34] L. Ljung, *System Identification - Theory For the User*. PTR Prentice Hall, 1999.

[35] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130 – 141, Mar. 1963.



(a) Multiple model structure



(b) Choose the best result from the multi-model and sub-models.

Fig. 1. Adaptive multiple linear modeling.

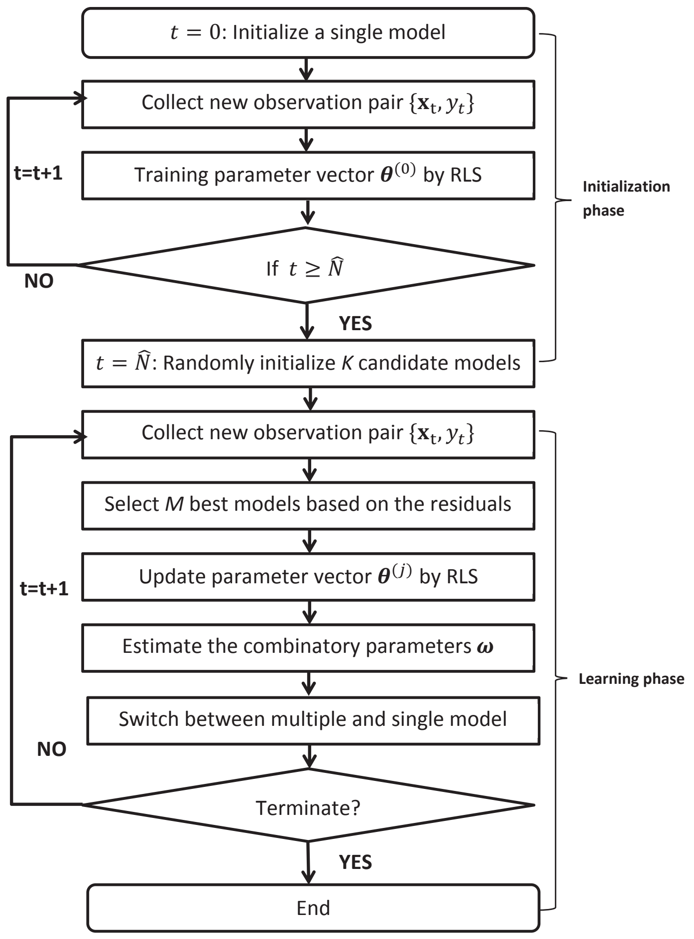

$t = 0$: Initialize a single model
Collect new observation pair $\{\mathbf{x}_t, y_t\}$
Training parameter vector $\boldsymbol{\theta}^{(0)}$ by RLS
t=t+1
If $t \geq \widehat{N}$
NO
YES
$t = \widehat{N}$: Randomly initialize K candidate models
Initialization phase
Collect new observation pair $\{\mathbf{x}_t, y_t\}$
Select M best models based on the residuals
Update parameter vector $\boldsymbol{\theta}^{(j)}$ by RLS
t=t+1
Estimate the combinatory parameters $\boldsymbol{\omega}$
Switch between multiple and single model
NO
Terminate?
YES
End
Learning phase

Fig. 2. **Flowchart of the adaptive multi-model algorithm**

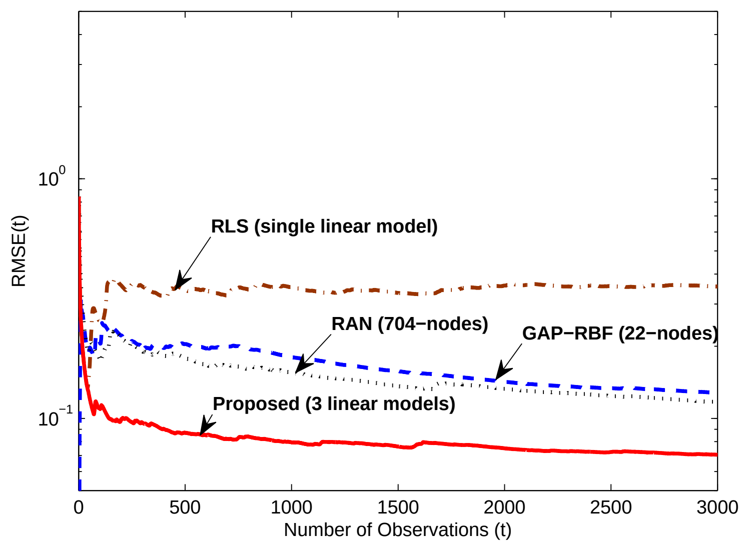


Fig. 3. **Mackey-Glass (fixed parameters)**: RMSE learning curves.

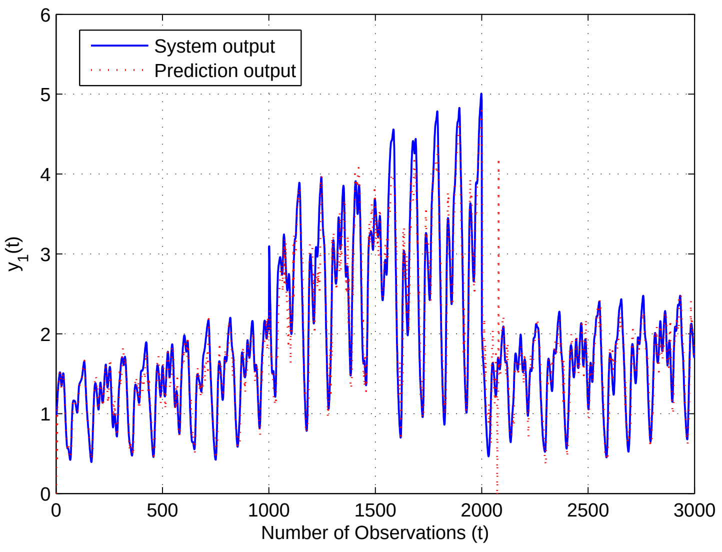


Fig. 4. **Mackey-Glass (piecewise function)**: prediction performance of the proposed algorithm with 3 selected sub-models.

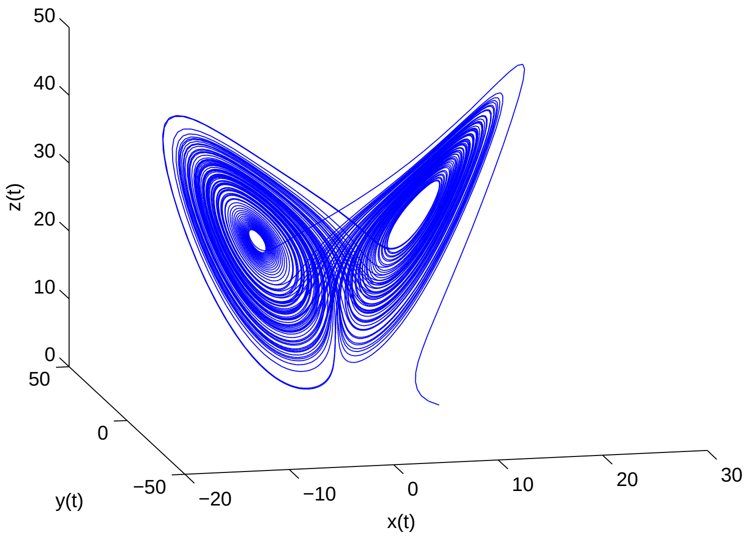


Fig. 5. Lorenz Attractor.

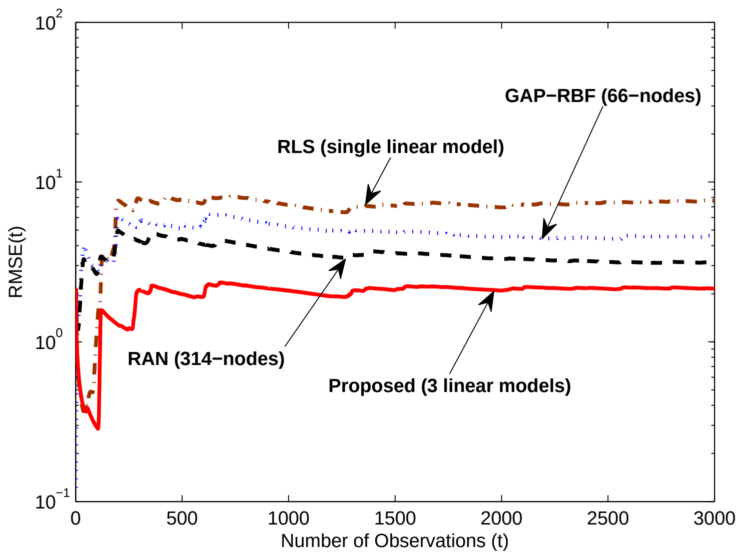


Fig. 6. **Lorenz time series (fixed parameters)**: RMSE learning curves.

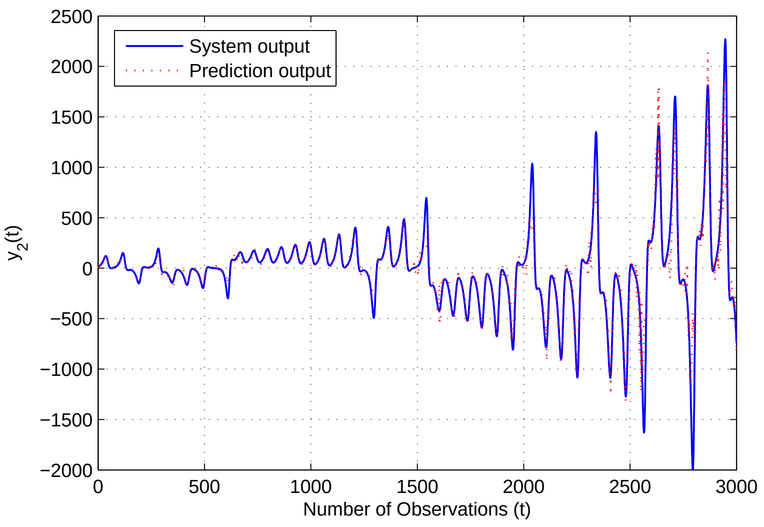


Fig. 7. **Lorenz time series (time base drift)**: prediction performance of the proposed algorithm with 3 selected sub-models.

TABLE I

Mackey-Glass (fixed parameters): FINAL PREDICTION PERFORMANCE, T=60

Algorithm	RMSE		Model size	Time(sec.)
	without noise	(with noise)		
RLS	0.3551	(0.3611)	single linear model	0.07
RAN	0.1170	(0.1731)	704 (774) RBF nodes	50.6
GAP-RBF	0.1278	(0.1648)	22 (32) RBF nodes	1.36
Proposed	0.0705	(0.1035)	3 selected sub-models	0.36
Proposed	0.0874	(0.1081)	5 selected sub-models	0.49

TABLE II

Mackey-Glass (piecewise function): FINAL PREDICTION PERFORMANCE, T=60

Algorithm	RMSE	Model size	Time(sec.)
RLS	0.8868	single linear model	0.0798
RAN	0.6694	2382 RBF nodes	39.7381
GAP-RBF	0.5323	195 RBF nodes	87.9364
Multiple models	0.2332	3 selected sub-models	0.4275
Multiple models	0.3364	5 selected sub-models	0.5444

TABLE III

Lorenz time series (fixed parameters): FINAL PREDICTION PERFORMANCE, T=50

Algorithm	RMSE Without noise	RMSE (with noise)	Model size	Time(sec.)
RLS	7.6369	(7.6498)	single linear model	0.08
RAN	3.1353	(3.2738)	314 (328) RBF nodes	14.6
GAP-RBF	4.6065	(4.7363)	66 (70) RBF nodes	11.4
Proposed	2.1510	(2.3431)	3 selected sub-models	0.39
Proposed	2.6711	(2.9388)	5 selected sub-models	0.49

TABLE IV

Lorenz time series (time base drift): FINAL PREDICTION PERFORMANCE, T=50

Algorithm	RMSE	Model size	Time(sec.)
RLS	399.6533	single linear model	0.0898
RAN	337.6048	2877 RBF nodes	22.1255
GAP-RBF	325.1256	1570 RBF nodes	1292.2099
Multiple models	156.1221	3 selected sub-models	0.4321
Multiple models	165.1258	5 selected sub-models	0.5233