A Hybrid Quasi-ARMAX Modeling Scheme for Identification of Nonlinear Systems

Jinglu Hu*, Kousuke Kumamaru**, Katsuhiro Inoue** and Kotaro Hirasawa*

This paper proposes a hybrid quasi-ARMAX modeling and identification scheme for nonlinear systems. The idea is to incorporate a group of certain nonlinear nonparametric models (NNMs) into a linear ARMAX structure. Particular effort is made to find a better compromise to the trade-off between the model flexibility and the model simplicity by using knowledge information efficiently. As the result, we obtain a model equipped with a linear ARMAX structure, flexibility and simplicity. The effectiveness and usefulness of the proposed hybrid model are examined by applying it to identification of a variety of nonlinear systems.

Key Words: nonlinear systems, quasi-ARMAX modeling, system identification, nonlinear nonparametric models, knowledge information

1. Introduction

The key problem in system identification is to find a suitable model structure, within which a good model is to be found. According to the levels of prior knowledge used, there are three types of models: white box models, grey box models, and black box models. When no physical insight is available or used, one has to choose black-box model structure which belongs to families that are known to have good flexibility and have been "successful in the past".

Under the assumption that the unknown system is linear, linear black-box models can be chosen for the system identification. The identification based on linear approximation has been extensively and successfully handled within some well known linear black-box structures. If the linear assumption is relaxed, one has to use nonlinear black-box models. For nonlinear black-box modeling, the "classical" literature seems to have concentrated on global basis function expansions, such as Volterra expansions. These have apparently had limited success. Recently, some authors have suggested the use of nonlinear structures based on neural networks (NN), wavelet networks (WN), radial basis function networks (RBFN), etc, and have achieved considerable success, see 6), 7). However, the latter ones emphasize only on the input-output representation ability and abandon some properties of the highly successful linear black-box modeling. Those properties such as structural linearity and simplicity for estimation are very useful in practical applications.

From a user's point of view, a nonlinear black-box model is preferred to have the following properties:

(1) A linear structure. In order to take advantage of linear system theory that is well developed, a linear structure may be useful. Therefore, one would benefit by constructing a nonlinear black-box model as an extension of linear model instead of abandoning the properties of linear model totally.

(2) Flexibility. Since a nonlinear system can be nonlinear in so many ways, a nonlinear black-box model structure, in general, must be feasible enough to deal with various nonlinear systems.

(3) Simplicity. A nonlinear black-box model usually offers a large amount of parameters. If the model is constructed to be linear in the parameters, its estimation becomes simple. Furthermore, if the model is constructed to be linear in the one-step past input variable, it is simple to derive a control law based on the model. Unfortunately, few existing nonlinear black-box models have those properties simultaneously. It therefore is highly motivated to develop a modeling scheme in order to obtain a nonlinear model equipped with a linear structure, flexibility and simplicity. For this purpose, we have the following motivations during the modeling:

(1) To find a better compromise to the trade-off between
the conflicting properties, since some of the properties, for instance flexibility and simplicity, usually appears to be conflicting ones.

(2) To divide the model parameters into two groups, and determine one of which by using knowledge information, since some of knowledge information are always available or can be obtained via some ways in practice.

It seems natural to consider the use of hybrid (linear-nonlinear) structure for realizing those motivations. In the literature, some authors have employed a "linear model + neural network" type hybrid scheme for identification and control of nonlinear systems, in which neural network is simply used as a compensator to describe the error due to nonlinear undermodeling\(^8\)-\(^{10}\). Such simple hybrid scheme does not, however, seem to have the properties of simplicity. In this paper, we will propose a new hybrid model structure based on an effective combination of a linear ARMAX structure and a group of certain nonlinear nonparametric models (NNMs) (neural networks, adaptive fuzzy systems, etc.). The basic idea of such hybrid modeling is first to increase the overall model flexibility by using a group of certain NNMs and then to restrict the flexibility in some ways for simplicity, e.g., by determining certain parameters using knowledge information. It is shown that a general nonlinear ARX system can be expressed in a linear ARX structure whose coefficients consist of constant parameters and nonlinear terms. Then a group of certain NNMs is incorporated into the linear ARX structure by using them to represent the nonlinear terms. In this way, we obtain a hybrid model structure which provides more freedoms so that particular effort can be made to find a better compromise between the model flexibility and the model simplicity by using knowledge information efficiently. The model built in this way is named as hybrid quasi-ARMAX model, which has a linear ARMAX structure, flexibility and simplicity.

The paper is organized as follows: in Section 2, we propose a hybrid quasi-ARMAX modeling scheme. Section 3 discusses the estimation of the hybrid model. Experimental studies using both real data and simulated data are carried out in Section 4. Finally, Section 5 is devoted to discussions and conclusions.

## 2. Hybrid Quasi-ARMAX Modeling

Let us consider an SISO general nonlinear system whose input-output can be described by

\[
y(t) = g(\varphi(t)) + v(t)\quad (1)
\]

\[
\varphi(t) = \begin{bmatrix} y(t-1) & \ldots & y(t-n) & u(t-1) & \ldots & u(t-m) \end{bmatrix}^T
\]

where \(y(t)\) is the output at time \(t\) \((t = 1, 2, \ldots)\), \(u(t)\) the input, \(\varphi(t)\) the regression vector, \(v(t)\) the system disturbance, and \(g(\cdot)\) the unknown nonlinear function which is assumed to be continuously differentiable.

Performing Taylor expansion to \(g(\varphi(t))\) around the region \(\varphi(t) = 0\)

\[
y(t) = g(0) + g'(0)\varphi(t) + \frac{1}{2} g''(0)\varphi(t)^2 + \ldots (2)
\]

Since \(g(\cdot)\) is assumed to be continuously differentiable, the derivative \(g^{(i)}(0)\) \((i = 1, 2, \ldots)\) exists. Then ignoring \(g(0)\) for simplicity and introducing two coefficient vectors \(\theta\) and \(\Delta \theta_t\) defined by

\[
\theta = g'(0)^T = [a_1 \ldots a_n b_1 \ldots b_m]^T
\]

\[
\Delta \theta_t = \begin{bmatrix} \frac{1}{2} g''(0) \varphi(t) + \ldots \end{bmatrix}^T
\]

\[
= \begin{bmatrix} \Delta a_{1,t} & \ldots & \Delta a_n,t & \Delta b_{1,t} & \ldots & \Delta b_{m,t} \end{bmatrix}^T
\]

we have

\[
\mathcal{M} : y(t) = \varphi^T(t)(\theta + \Delta \theta_t) + v(t) \quad (3)
\]

\[
v(t) = \epsilon(t) + c_1 \epsilon(t-1) + \ldots + c_2 \epsilon(t-n) \quad (4)
\]

where (4) is a moving average (MA) noise model introduced for the system disturbance, in which \(\epsilon(t)\) is white noise. It is clear that in the model (3) \(a_i, b_i\) and \(c_i\) are the constant parameters, while \(\Delta a_{i,t} = \Delta a_i(\varphi(t))\) and \(\Delta b_{i,t} = \Delta b_i(\varphi(t))\) are nonlinear functions of \(\varphi(t)\). We call (3) a quasi-ARMAX model in order to distinguish it from nonlinear ARMAX (NARMAX) model in the literature. In Fig.1, the quasi-ARMAX modeling is described graphically.

Next, let us see an example. The nonlinear system is assumed to be the type of Kolmogrov-Gabor polynomial\(^{11}\)
of order \( r \)

\[
y(t) = \sum_{i=1}^{r} \alpha_i x_i(t) + \sum_{i=1}^{r} \sum_{j=1}^{r} \alpha_{ij} x_i(t) x_j(t) + \ldots,
\]

(5)

in which the elements of \([x_i(t) \; i = 1, \ldots, r]\) are assumed to be the past input-outputs of system

\[
x(t) = y(t - i); \quad i = 1, \ldots, n
\]

\[
x_{j+n}(t) = u(t - j); \quad j = 1, \ldots, m
\]

\((r = n + m)\)

Using (6) in (5), we can express (5) in quasi-ARX form

\[
y(t) = \sum_{i=1}^{n} (a_i + \Delta a_{i,t}) y(t-i) + \sum_{i=1}^{m} (b_i + \Delta b_{i,t}) u(t-i)
\]

(7)

where the \( \Delta a_{i,t} \) and \( \Delta b_{i,t} \) are explicitly expressed as

\[
\Delta a_{i,t} = \sum_{j=1}^{n} \alpha_{ij} y(t-j) + \frac{1}{2} \sum_{j=n+1}^{r} \alpha_{ij} u(t-j+n) + \ldots
\]

(8)

\[
\Delta b_{i,t} = \sum_{j=1}^{r} \alpha_{ij} u(t-j+n) + \frac{1}{2} \sum_{j=1}^{n} \alpha_{ij} y(t-j) + \ldots
\]

(9)

In (7), \( a_i \) and \( b_i \) stand for \( \alpha_i \) in (5).

Obviously, the nonlinear terms \( \Delta a_{i,t} \) and \( \Delta b_{i,t} \) can not always be expressed in explicit forms when general nonlinear systems are considered. We here will represent them using the NNMs as follows:

\[
\Delta a_{i,t} = f_i(\varphi(t)) \quad (i = 1, \ldots, n)
\]

(10)

\[
\Delta b_{j,t} = f_{j+n}(\varphi(t)) \quad (j = 1, \ldots, m)
\]

(11)

\[
f_i(\varphi(t)) = \sum_{j=1}^{M} \omega_{ij} N_j(p_j, \varphi(t))
\]

(12)

where \( N_j(p_j, \varphi(t)) \)’s are the ‘basis functions’ \( (1) \), \( \omega_{ij} \)'s are the coordinate parameters, and \( p_j \)'s are the scale and position parameter vectors. A useful example for the NNMs is the adaptive fuzzy systems (AFS) introduced by L.X. Wang and J.M. Mendel in (12), which can be explicitly expressed as

\[
f_i(\varphi(t)) = \frac{\bigwedge_{k=1}^{M} \mu_{A_k}^i(x_k(t))}{\sum_{j=1}^{M} \mu_{A_k}^j(x_k(t))}
\]

(13)

where \( \bigwedge \) is the minimum operator, \( M \) is the number of rules, \( x_k(t) \) are the elements of \( \varphi(t) \), and \( \mu_{A_k} \) is the membership function of fuzzy set \( A_k \). In this case, \( N_j(p_j, \varphi(t)) \) has the expression

\[
N_j(p_j, \varphi(t)) = \frac{\bigwedge_{k=1}^{M} \mu_{A_k}^j(x_k(t))}{\sum_{j=1}^{M} \bigwedge_{k=1}^{M} \mu_{A_k}^j(x_k(t))}
\]

in which \( p_j \) is the parameter vector determining the membership functions \( \mu_{A_k}^j(k = 1 \ldots r) \), see (6) for an extensive discussion about the ‘basis function’ \( N_j(p_j) \). The model described by (3) and (10)-(12) is named as hybrid quasi-ARMAX model.

The NN is known to be flexible enough to represent most reasonable systems in practice. In our hybrid quasi-ARMAX model there are a group of such certain NNMs. Therefore, it becomes so flexible (complex) that it is difficult to estimate all of the parameters \( (a_i, b_i, c_i, \omega_{ij} \) and \( p_j) \) from observed data as usual. In order to solve this problem, we divide the estimation procession into two sub-procession; (1) determining the vector \( p_j \) specifying the ‘basis functions’ in the NN using knowledge information obtained from pre-procession or during the estimation; (2) estimating the parameters \( a_i, b_i, c_i \) and \( \omega_{ij} \) from observed data using optimization-based method. We will discuss these estimations in the next section. The followings are some interpretations for the hybrid model.

(1) Expression in Linear ARMAX Structure

Introducing a coefficient vector \( \Theta_t \)

\[
\Theta_t = \theta + \Delta \theta_t,
\]

(14)

we have an expression of the hybrid quasi-ARMAX model

\[
M: \quad y(t) = \varphi^T(t) \Theta_t + \nu(t).
\]

(15)

From (15), we can see that the hybrid quasi-ARMAX model has a linear ARMAX structure, which is useful when the model is applied to system analysis.

(2) Expression in Combined Structure

Using (10)-(12) in (3) we can obtain another expression of the hybrid quasi-ARMAX model

\[
M: \quad y(t) = \varphi^T(t) \theta + \nu(t)
\]

\[
+ \sum_{j=1}^{M} \varphi_{ARX}^T(t) \Omega_j N_j(p_j, \varphi(t))
\]

(16)

where \( \Omega_j = [\omega_{ij} \ldots \omega_{ij}]^T \). (16) shows that the hybrid quasi-ARMAX model is equivalent to a hybrid model combining a linear ARMAX model and a multi-ARX-model. The multi-ARX-model consists of \( M \) local ARX models and its overall performance is obtained via an interpolation using the ‘basis function’ \( N_j(x) \). It also implies that the proposed model can be shown to be able to describe any sufficiently smooth nonlinear function in (1) on a compact interval arbitrarily well by merely increasing the value of \( M \).
Fig. 2 The hybrid quasi-ARMAX model shown as an associative memory networks. (the MA noise model has been omitted for clarity.)

(3) Expression in Linear Regression Structure
Introduce a parameter vector \( \Theta \) and a regression vector \( \varphi_{NL}(t) \) defined as
\[
\Theta = [\theta^T, \omega_1, ..., \omega_M, c_1, ..., c_l]^T \tag{17}
\]
\[
\varphi_{NL}(t) = [\varphi^T(t), \varphi^T(t) \otimes \varphi_{NL}^T(t), e(t-1) ... e(t-1)]^T \tag{18}
\]
where \( \varphi_{NL}^T(t) = [N_j(p_j, \varphi(t), j = 1, ..., M)] \), and the symbol \( \otimes \) denotes Kronecker production. Then we have the third expression of the hybrid quasi-ARMAX model
\[
M: y(t) = \varphi_{NL}^T(t)\Theta + e(t) \tag{19}
\]
The expression (19) shows that the proposed model is simple for parameter estimation because it is linear in the parameters \( \Theta \) to be estimated.

(4) Expression in Associative Memory Networks
From a network’s point of view, the hybrid quasi-ARMAX model can also be seen as an associative memory networks, which consists of two hidden layers: the first layer (next to the input layer) with weights determined by a set of simplified NNMs; the second layer with weights simply taking the time delayed value of the system input and output, see Fig. 2. In this sense, we may consider the hybrid quasi-ARMAX models as a specially constructed associative memory network in favorable to identification, control design or system analysis of nonlinear dynamical systems. This is significant because in the network structure shown in Fig. 2, the input vector is not necessary to be \( \varphi(t) \). It will become very convenient in some applications (e.g. control design) if an appropriate vector rather than \( \varphi(t) \) is chosen as the input vector of associative memory networks.13)

Finally, from the above discussions we can conclude that the hybrid quasi-ARMAX model is equipped with a linear ARMAX structure, flexibility and simplicity. The modeling scheme proposed here is different from the “linear model + neural network” type hybrid schemes appeared in the literature8).

3. Estimation of the Hybrid Model
In this section, we first have a summary about what steps are needed in order to find a hybrid quasi-ARMAX model. Next, we discuss a scheme to determine the parameter vectors \( p_j \) based on knowledge information and the algorithm for estimating the parameters \( (a_i, b_i, c_i \text{ and } \omega_{ij}) \) using observed data.

3.1 Steps for Obtaining A Hybrid Model
In order to obtain a hybrid quasi-ARMAX model, the following steps are necessary:
(1) Select the regression vector \( \varphi(t) \). This is equivalent to determining the order \( n \) and \( m \). Since the hybrid quasi-ARMAX model is basically an extension of linear ARMAX model to the nonlinear one, \( \varphi(t) \) will be determined based on the results of identifying the system using a linear ARMAX model. Therefore, many existing approaches for determining the order of linear models such as Akaike criteria AIC and FPE can be applied as a reference. However, the order \( n \) and \( m \) for \( \varphi(t) \) should be chosen as small as possible, so far as the performance of the linear model is not significantly worse, instead limiting on the optimal order.
(2) Select a scalar ‘mother basis function’ \( N_j \). Theoretically, all the NNMs which are described by (12) can be used. However, the parameters \( p_j \) specifying the nature of the ‘basis function’ will be determined using knowledge information, so some of them (e.g. AFS, RBFN and B-spline based models) are more feasible, while some others (e.g. NN and WN) are less feasible.
(3) Determine the parameter vectors \( p_j \). For general NNMs, this is still an open problem to be solved, which depends on the kind of NNMs used. Section 3.2 describes a strategy which can be used for AFS, RBFN or B-spline based models.
(4) Determine the order \( M \). This is related with determining the parameter vectors \( p_j \). In Section 3.2 we will gives several hints to reduce the order \( M \). When the AFSs are used as the NNMs, the order \( M \) denotes the number of rules in the AFS. Therefore, this is a problem similar to building an adaptive fuzzy system using knowledge information and observed data. Many existing results for fuzzy system design can be applied.
(5) Estimate model parameters \( \Theta \). The \( \Theta \) denotes the unknown parameters to be estimated from the observed data. We will discuss the estimation problem in Section 3.3.
3.2 Determining $p_j$ Using Knowledge Information

As mentioned before, in order to achieve the model simplicity and the model flexibility simultaneously, the hybrid quasi-ARMAX modeling tries to increase the model overall flexibility by using a group of NNMs and then to restrict the flexibility in some way for the simplicity. The latter is done by determining the scale and position parameter vectors $p_j$ of the 'basis functions' in the NNMs using knowledge information. Since in a black-box modeling, the physical insight of system is assumed to be not available, the knowledge information are mainly obtained from the observed data and the prediction error in the pre-processing or during the estimation. Several kinds of knowledge information can be considered to be useful. They are

- the information concerning the operating region of $\varphi(t)$,
- the information about the structure of nonlinearity,
- the information concerning the relations among the elements in $\varphi(t)$, and
- the information about the size of prediction errors and their relations with the operating region of $\varphi(t)$.

However, how to use those kinds of information efficiently are still under investigation. Here only some suggestions can be given.

3.2.1 A Strategy for Determining $p_j$

How to determine the parameter vector $p_j$ depends on the kind of NNMs used. The following strategy is suitable for AFS, RBFN and B-spline based models.

Suppose the NNM has $r$ inputs, $X = [x_i, i = 1, ..., r]$ and the operating region is mostly located in $X_{\text{min}} \leq X \leq X_{\text{max}}$, $X_{\text{min}} = [x_{i_{\text{min}}}, i = 1, ..., r]$, $X_{\text{max}} = [x_{i_{\text{max}}}, i = 1, ..., r]$. $X \notin [X_{\text{min}}, X_{\text{max}}]$ is allowable in practice. We first partition the input hyperplane, that is, put nodes into the input hyperplane. As shown in Fig.3, if the number of nodes corresponding to $x_i$ is denoted as $n_i$, the total number of the nodes in the hyperplane will be $M = \prod_{i=1}^{r} n_i$. Next, the parameter vectors $p_j$ are chosen so that the 'basis functions' $N_j(p_j, X)$ have appropriate shape and are put onto each node. Without using other knowledge information, the nodes will be uniformly assigned in the hyperplane. Figure 3 shows an example for determining $p_j$ for AFS with $r = 2$ and $M = 4 \times 4$. It should however be noticed that this strategy may not be suitable for some NNMs such as neural networks and wavelet networks. Further research is needed for using neural networks and wavelet networks as NNMs in the hybrid quasi-ARMAX model.

3.2.2 Hints for Reducing the Number of Nodes

The prior knowledge concerning operating region $[X_{\text{min}}, X_{\text{max}}]$ is the least information required for determining the parameter vectors $p_j$. However, when dim($X$) is large, the total number of nodes ($M$) may be rather large. Therefore, further information should be used to reduce the number of nodes or to improve the node assignment. The following hints can be used to reduce the total number of nodes $M$:

(1) Hint A: If the system is linear with respect to $x_i$, $n_i$ may be chosen to be 1.

(2) Hint B: If no other useful information is available, $n_1$ and $n_{n+1}$ corresponding to $y(t-1)$ and $u(t-1)$ are assigned with appropriate values, while all other $n_i$'s are set to 1.

(3) Hint C: If the role of nodes can be replaced by employing interpolation of NNMs, those nodes may be removed from the input hyperplane. However, the above hints are only intuitive ones. In order to make the modeling scheme less heuristically dependent, further research is needed to develop an algorithm for incorporating knowledge information automatically.

3.3 Recursive Estimation Algorithm

The parameters to be estimated in the hybrid quasi-ARMAX model are ARMAX parameters $a_i, b_i, c_i$ and the coordinate parameters $\omega_{ij}$ of the NNMs. Recalling (19), we have

$$y(t) = \varphi^T_{NL}(t)\Theta + e(t).$$

It is well known that the estimation of $\Theta$ for (20) can be done by minimizing the following criterion function

$$V_N(\Theta) = \frac{1}{N} \sum_{k=1}^{N} \varphi^2(t) = \frac{1}{N} \sum_{k=1}^{N} |y(t) - \hat{y}(t|\Theta)|^2,$$

which is carried out using a recursive algorithm\(^2\).
\begin{align}
\hat{\Theta}(t) &= \hat{\Theta}(t-1) + L(t)\varepsilon(t, \hat{\Theta}(t-1)) \\
L(t) &= \frac{P(t-1)\psi(t, \hat{\Theta}(t-1))}{1 + \psi^T(t, \hat{\Theta}(t-1))P(t-1)\psi(t, \hat{\Theta}(t-1))} \\
P(t) &= P(t-1) - L(t)\psi^T(t, \hat{\Theta}(t-1))P(t-1)
\end{align}

where
\[\varepsilon(t, \hat{\Theta}(t-1)) = y(t) - \varphi_{NL}^T(t)\hat{\Theta}(t-1)\]
and
\[\psi(t, \hat{\Theta}(t-1)) = \varphi_{NL}(t|\hat{\Theta}(t-1))\]

for Extended Least Square (ELS) method or
\[
\psi(t, \hat{\Theta}(t-1)) = \varphi_{NL}(t|\hat{\Theta}(t-1))/C(q^{-1}, \hat{\Theta}(t-1))
\]
for Prediction Error Method (PEM). Here, \(\varphi_{NL}(t)\) whose elements \(\varepsilon(t-i)\) are replaced by \(\varepsilon(t-i, \hat{\Theta}(t-1))\). Since a better initial value can improve the convergence property very much, the above algorithm will benefit by implementing it in the following two stages.

1. Fix \(\omega_{ij} = 0\) and adjust \(a_i, b_i, c_i\) for appropriate steps. The number of parameters to be adjusted in this stage is usually small. A hybrid identification method using genetic algorithm may therefore be applied effectively.

2. Continue the estimation for all parameters \((a_i, b_i, c_i, \omega_{ij})\) using the results obtained in the stage (1) as initial values. It is found experimentally that fixing \(c_i\) in this stage gives better identified noise model.

4. Experimental Studies

In this section, we will apply the hybrid quasi-ARMAX model to identify a real system and a simulated system. These systems are well known in the literature, where they have been used to test nonlinear black-box models such as Neural Networks, Wavelet Networks and Hinging Hyperplanes. We will compare our results with those using nonlinear black-box models.

4.1 Modeling A Hydraulic Robot Actuator

The position of a robot arm is controlled by a hydraulic actuator. The oil pressure in the actuator is controlled by the size of the valve opening through which the oil flows into the actuator. The position of the robot arm is then a function of the oil pressure. Let us denote by \(u(t)\) and \(y(t)\) the position of the valve and the oil pressure at time \(t\), respectively. A sample of 1024 pairs of \(\{y(t), u(t)\}\) was registered. We divide it into two equal parts for estimating and for validating our model, respectively. The estimation data is depicted in Fig.4.

4.1.1 Using a linear model

Following the principle of ‘try simple things first’, we first use a linear ARX model to identify the system. A reasonable modeling has been obtained with \(n = 3\) and \(m = 2\), that is, the regression vector \(\varphi(t) = [y(t-1) y(t-2) y(t-3) u(t-1) u(t-2)]^T\). Figure 5(a) shows the result of a simulation with the obtained linear model on validation data, which gives a root mean square (RMS) error of 1.0160. The result is not very impressive.

4.1.2 Using neural networks and wavelet networks

The problem of modeling the hydraulic robot actuator has been discussed comprehensively by a Sweden and France group. J. Sjöberg modeled the system using neural network models, Q. Zhang using wavelet network models, and P. Pucar using hinging hyperplane models. For the sake of easy comparison, we carry out similar simulations using Matlab Neural Network toolbox and the package of Zhang (1993). First, a NARX model based on an one-hidden-layer sigmoid neural network with 10 hidden units, 5 input units and one output unit is considered. This gives a model with 71 parameters. The identified neural network NARX model is simulated on the validation data, which gives a root mean square (RMS) error of 0.617. Our result is a little worse than that obtained by J. Sjöberg, see Fig.9 in 6), where the RMS error is 0.467. The reason we think is that we have over-trained the model. J. Sjöberg and L. Ljung have developed an algorithm using regularization to solve the over-training problem. Next, another NARX model based on a wavelet network is considered to model the hydraulic actuator in a similar way, with the same regressors. The wavelet function used is \(\psi(\varphi) = (d - \varphi^T \varphi)e^{-\varphi^T \varphi/2}\).
with $d = \dim \varphi$. Since this identification is realized using the package of Zhang, we obtained the same result as that obtained by Q. Zhang, see Fig.10 in 6). The result of the identified wavelet network model simulated on the validation data is shown in Fig.5(c), which gives a RMS error of 0.5285.

In 6), Sjoberg et al. reported that they have also identified the hydraulic robot actuator using several other non-linear black-box models with various nonlinear structures. The best model they obtained simulated on the validation data gives a RMS error of 0.328, referred to 6) for details.

4.1.3 Using a hybrid quasi-ARMAX model

Now we use the proposed hybrid quasi-ARMAX model to identify the hydraulic robot actuator. For easy comparison, we do not consider the noise model first, i.e., $n = 3$, $m = 2$, $l = 0$ are chosen for the regression vector. From the estimation data shown in Fig.4, we choose $X_{\text{min}}=[-4 -4 -4 -2 -2]$ and $X_{\text{max}}=[4 4 4 2 2]$. Since no other useful information available, we choose $n_1 = n_4 = 4$, $n_2 = n_3 = n_5 = 1$ using Hint B, which gives $M = 16$. The model obtained thus has 85 parameters to be estimated. After estimating the model using the estimation data for 2048 steps, the simulation of the model on the validation data is shown in Fig.6(a), which gives a RMS error of 0.5445. Comparing this result with those using neural networks and wavelet networks (see Fig.5 (b) and (c)), we can see that the hybrid quasi-ARMAX model has given a compatible performance.

In 1), J. Sjoberg reported that for this example, separate noise models could not improve the fit substantially. Since a noise model has to be identified from the residual of system model, it is difficult to identify it if the system model is nonlinear in the parameters to be estimated. However, our hybrid quasi-ARMAX model is linear in the parameters and includes a MA noise model naturally, so that the noise model can be identified easily. Next, we use a hybrid quasi-ARMAX model with $n = 3$, $m = 2$, $l = 1$ and $M = 16$ to identify the system in a similar way. Figure 6(b) shows the simulation of the model on the validation data. The RMS error is 0.1360, which is better than the best results in 1). We think that the main reason for the superior performance of the proposed model is that it has better property in dealing with correlated noise.

4.2 Modeling A Mathematical System

The mathematical system is taken from Narendra (1990) 19), which contains rather strong nonlinearity. The system is governed by

\[ y(t) = f[y(t-1), y(t-2), y(t-3), u(t-1), u(t-2)] \] (28)

where

\[ f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2}. \]

Estimation data are sampled when system is excited us-
Fig. 7 Simulations of identified models on validation data. The solid line shows the system true output and the dashed line the simulated model output.

using random input uniformly distributed in the interval \([-1, 1]\), while validation data are sampled from system using an input \(u(t) = \sin(2\pi t/250)\) for \(t \leq 500\) and \(u(t) = 0.8\sin(2\pi t/250) + 0.2\sin(2\pi t/25)\) for \(t > 500\).

First, an ARX model with \(n = 3\), \(m = 2\) is used to identify the system. The simulation of ARX model on the validation data is shown in Fig. 7(a), which gives a RMS error of 0.0866. The result is not very impressive. Next, a four-layer neural network of \(N_{5,20,10,1}\), which contains 341 parameters to be estimated, is used to identify the system. Similar to K.S. Narendra in 19), we train the model for 100,000 steps using Matlab Neural Network toolbox. The simulation of the model on the validation data is shown in Fig. 7(b). The RMS error is 0.0678, which is compatible to the result obtained in 19), but is not very impressive too.

The reason we think is that the identification algorithm has been stuck at a local minimum. Finally, we use the hybrid quasi-ARMAX model with \(n = 3\), \(m = 2\), \(l = 0\) to identify the system. \(X_{\text{min}} = [-1 -1 -1 -1 -1]\) and \(X_{\text{max}} = [1 1 1 1 1]\) are chosen based on the information obtained from the estimation data. Since the system is linear with respect to \(u(t - i)\), we choose \(n_1 = n_2 = n_3 = 3\), \(n_4 = n_5 = 1\) using Hint A. \(M = 18\) is then obtained by removing some nodes using Hint C. We thus obtain a model with 95 parameters to be estimated. After estimating the model for 5000 steps, the simulation of the model on the validation data is shown in Fig.7(c). The RMS error is 0.0270. We can see that the hybrid quasi-ARMAX model represents the system very well. We think that the reason for the better performance of the proposed model is that our estimator could find the global minimum.

5. Discussions and Conclusions

We have proposed a hybrid quasi-ARMAX modeling scheme, in which a group of NNMs are embedded into the coefficients of a linear ARMAX structure. Because of the use of a group of NNMs, we may have some extra freedoms, which make it possible to identify the model by using an appropriate modeling procedure and an appropriate estimation algorithm so that the identified models are favorable to various objectives such as control design, system analysis or prediction and simulation of nonlinear systems. In this paper, we have mainly concentrated on the modeling scheme and have demonstrated its simulation ability experimentally. The applications of hybrid quasi-ARMAX modeling schemes for control design and fault detection of nonlinear systems will be discussed in separate papers, which are under preparation. The interested reader may also be referred to 20)~23). On the other hand, in order to make our hybrid modeling scheme less heuristically dependent, further research is needed to develop an algorithm for incorporating knowledge information automatically.

Based on the result of numerical simulations and the discussion throughout the paper, we may conclude that the hybrid quasi-ARMAX model is equipped with a linear ARMAX structure, flexibility and simplicity. It will be useful in real applications.

References


