Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Genetic programming-based chaotic time series modeling*

ZHANG Wei (张 伟)†, WU Zhi-ming (吴智铭), YANG Gen-ke (杨根科)

(*Department of Automation, Shanghai Jiaotong University, Shanghai 200030, China*)

†E-mail: zhang_wi@sjtu.edu.cn

Received Sept. 18, 2003; revision accepted Dec. 12, 2003

Abstract: This paper proposes a Genetic Programming-Based Modeling (GPM) algorithm on chaotic time series. GP is used here to search for appropriate model structures in function space, and the Particle Swarm Optimization (PSO) algorithm is used for Nonlinear Parameter Estimation (NPE) of dynamic model structures. In addition, GPM integrates the results of Nonlinear Time Series Analysis (NTSA) to adjust the parameters and takes them as the criteria of established models. Experiments showed the effectiveness of such improvements on chaotic time series modeling.

Key words: Chaotic time series analysis, Genetic programming modeling, Nonlinear Parameter Estimation (NPE), Particle Swarm Optimization (PSO), Nonlinear system identification

doi:10.1631/jzus.2004.1432 **Document code:** A **CLC number:** TN914

INTRODUCTION

Chaos is based on the principle that simple deterministic laws can exhibit complex external behavior (Hegger and Kantz, 2000); although it is quite difficult to reveal such simple laws from the system's external behavior only. This is mainly due to the dissipation and sensitivity to initial conditions (SIC) properties of the chaotic systems (Wei *et al*., 2002). Linear system theory has developed for many years, but has proved to be incapable of sufficiently modeling the chaotic time series. The identification and modeling of nonlinear systems is still a very active field of research at present.

Artificial Neural Networks (ANN) and Polynomials (Jian and Zheng, 2002) are two methods for global modeling. But they both cannot give simple and elegant model representations. Essentially speaking, they are more appropriate to be regression and approaching tools than to be modeling tools. They are less powerful in revealing the system dynamic laws and are difficult to be integrated with the pre-discovered knowledge on chaotic systems. Generic Programming (GP) (Pan *et al*., 1998) proposed in this paper is integrated with Nonlinear Time Series Analysis (NTSA) to model the chaotic time series. A prototype named GPM is implemented by mix-programming of C++ and MATLAB. The NTSA results not only guide the evolution of GPM, but also give the final criteria of the model quality according to chaotic system invariants instead of generally used mean square error (MSE). Particle Swarm Optimization (PSO) technique is introduced in GPM to overcome the difficulty of Nonlinear Parameter Estimation (NPE) of dynamic model structures, which can reduce the computations and improve the efficiency of GPM.

This paper is organized as follows: Section 2 introduces the architecture design of GPM; Section 3 gives the detail descriptions of the GP-based chaotic modeling techniques in GPM; Section 4 gives several modeling examples and summarizes

* Project (Nos. 60174009 and 70071017) supported by the National Natural Science Foundation of China

some modeling experiments, and finally Section 5 presents the conclusion.

ARCHITECTURE OF GPM

The GPM is implemented with C++ and MATLAB. The data flow is illustrated in Fig.1.

Generally, there should be some stationary validation and noise reduction steps on the actual measured time series before further analysis. GPM uses wavelets analysis to filter the non-stationary trends and noises. The results of GPM may not converge if there is too much noise in the time series. Then the time series should be analyzed by NTSA techniques to reveal the necessary information such as the embedding dimension m, the largest Lyapunov exponent λ and the maximum prediction length L on the chaotic system. This information will further guide the evolution of the GP's modeling process. The GP here is responsible for exploring appropriate model structures. The unknown parameters in the model are estimated through the NPE module. Usually the GPM will output a group of candidate models, so we should examine these models to select the best one.

Fig.2 illustrates the organization and hierarchy of each module in GPM.

PRINCIPLE AND DETAIL DESCRIPTION OF GPM

Model structure selection and model parameter estimation are two fundamental problems in system identification fields. Generally, the modeling process will assume some kinds of model structure, then further efforts focus on model refinement and parameter estimation, for example, the ARMA modeling and polynomial modeling (Jian and Zheng, 2002). But these methods are less effective when dealing with nonlinear chaotic systems because of the dissipation and SIC property. Chaotic systems usually contain positive feedback components, delay components or interaction components. The ordinary model structures are usu-

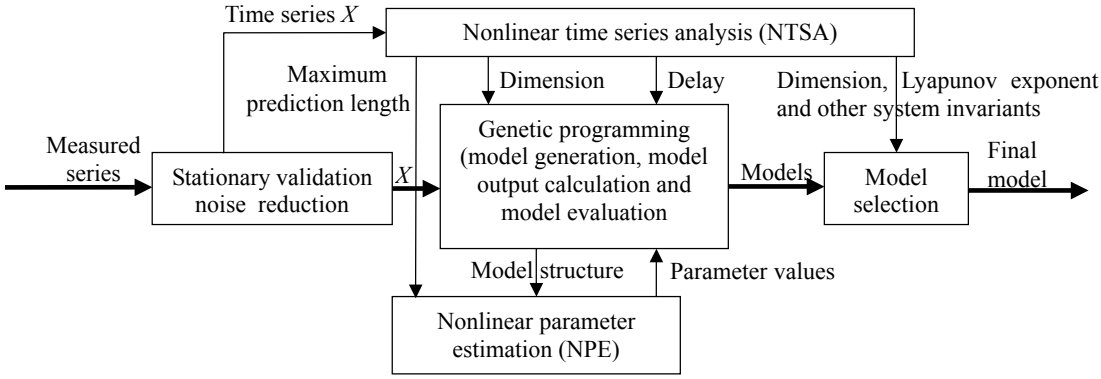

Fig.1 Architecture of GPM

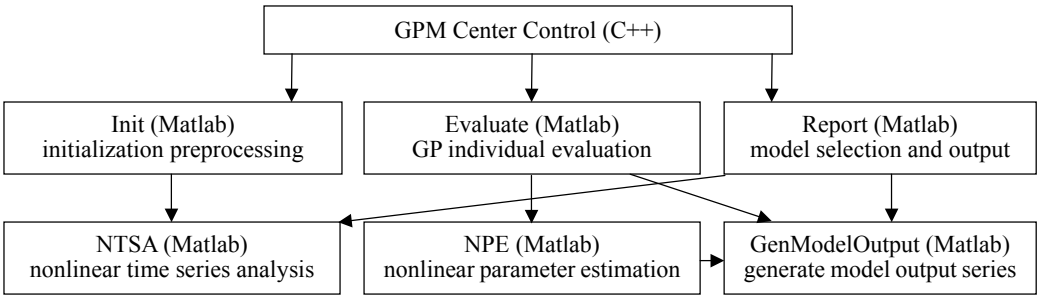

Fig.2 Organization and hierarchy of each module in GPM

ally less powerful in expressing such models. Furthermore, these methods tend to be diverged and the results are too complex when the input dataset becomes large. So we had better explore more function space to get better models for chaotic systems, especially for those models with simple expressions. This can be achieved through genetic programming in GPM.

The next problem after establishing the model structure is the model parameter estimation. Since the model structures generated by GPM are highly dynamic and usually nonlinear, they are difficult for most of the classical parameter estimation algorithms to be used. So the PSO algorithm (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998) is introduced to assume this duty. It has relatively less computation than other random algorithms such as GA.

Genetic programming

Genetic Programming (GP) is responsible for searching appropriate model structures in the function space. In GPM, each model is represented as a symbol tree structure (Koza, 1990). The leaf nodes are selected from Terminator Set (*TS*), and the internal nodes are selected from Function Set (*FS*). The model expression can be generated through the tree's first-order traverse algorithm. Essentially speaking, the maximum model space explored is the span of both function set and terminator set. The process of modeling is mainly the search process in this span space for appropriate models.

In GPM, $TS_{\max}=\{CONSTANT, X\}$, $FS_{\max}=\{$add, multiply, sub, divide, cos, sin, acos, exp$\}$. X denotes the independent variables and *CONSTANT* denotes the constant coefficient in the model. For a specific problem, we can choose the operators that have the most ability to express system dynamics according to our knowledge of the system.

Some GP modeling systems introduced z operator, namely, the *delay* operator (Leung and Varadan, 2002; Varadan and Leung, 2001), to increase the model dimensions, which can enhance the model ability in expressing complex behaviors. In fact, the dimension information can be analyzed approximately through NTSA techniques.

The GP operators are designed as follows (Pan *et al*., 1998):

a) Initialize: the process of GP individual's initialization is essentially the process of tree random generation. Half of the individuals are generated according to the depth-first principle, and the others are generated according to width-first principle. Some pre-defined patterns can also be added into the initial population to improve the modeling efficiency.

b) Crossover: the crossover operator involves two parent trees. Firstly, two crossover points are selected randomly in the parents respectively, then the two offsprings are generated by exchanging the two sub-trees rooted with the crossover points.

c) Mutate: Similar to the crossover operator, one mutation point must be chosen in the tree randomly at first. If the mutation point is a leaf node, do the following operations in equal probability: replace the current node with another valid terminator operator in *TS* or replace the current node with a randomly generated tree. While, if the mutation point is not a leaf node, do the following in equal probability: remove the sub-tree rooted as the mutation point and replace it with a terminator operator, or remove the sub-tree and replace it with a randomly generated tree.

Nonlinear parameter estimation of dynamic models

Since a good model structure may be diminished or even die out in GP's evolution due to improper parameters, GPM separates the process of parameter estimation from model exploring in order to match the models with their appropriate parameter values. The model formula with unknown parameters is generated from the tree structure according to the following rules:

a) The model formula is generated from the tree structure according to first-order tree traversing;

b) Each function operator and terminator operator can be attached with a multiplicative coefficient as the model parameter;

c) If the terminator is a *CONSTANT* operator,

then replace *CONSTANT* with a model parameter. No additional multiplicative coefficient should be attached;

d) If the function operator is linear such as plus, minus, multiply and divide, no additional coefficients should be attached, because such coefficients are redundant when considering their child nodes.

```
.1.plus
  .2.times
    .3.minus
      .4.X(1)
      .4.constant
    .3.X1
  .2.constant
```

Fig.3 Tree representation of a GP individual

For example, the model formula generated from the tree structure in Fig.3 is: plus(times(minus($A(1)$*$X(1)$, $A(2)$), $A(3)$*$X(1)$), $A(4)$). In GPM, $X(k)$ denotes delaying the input series to k time units.

Different from the NPE algorithms on known model structure, the model formula generated in GPM is highly dynamic. Furthermore, improper model structures may be generated during the process of crossover and mutation. Even the appropriate model structures may generate improper output because of invalid input or parameter values. They all lead to the difficulties of parameter estimation in GPM. The Particle Swarm Optimization (PSO) algorithm (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998) is introduced in GPM in order to get better parameter values to overcome such difficulties.

PSO is a kind of group intelligence algorithm like GA (Xie *et al.*, 2003). Each individual in PSO is represented as a flying particle without volume in the parameter space. The particle can adjust its flying speed and directions according to its own history and the group's experience. Compared with the classical GA algorithm, the PSO usually employs small population (4–30) and fewer generations (usually<200). Moreover, the PSO is relatively simple and can be efficiently implemented, which can reduce the computations and improve the performance of GPM.

Evaluating the model output

After being given the model structure, the parameter values and the model input, we can parse the model formula and evaluate its output. For simplicity and efficiency, we still adopt the mean square error (MSE) as the criterion to judge the model quality during GP's evolution. In addition, the traditional GP-based modeling methods can only deal with small datasets, because the noise in the model input tends to cause the modeling results to diverge when the input dataset becomes large. For chaotic systems, the SIC property enhances such trends. The chaotic nature allows only short-term prediction of future values of the model output (Kantz and Schreiber, 1997). In order to deal with large datasets and their chaotic nature, we adopt multi-stage prediction (Fig.4) to calculate the model output series.

Essentially the model output series in GP is the concatenation of the prediction values with model input draw-out from the original measured series. Since the largest Lyapunov exponent λ describes the diverging rate of chaotic trajectories, the maximum prediction length L can be estimated approximately as $L\approx 1/\lambda$ (Lv *et al.*, 2002). Considering the chaotic property SIC, the value of L can be extended moderately larger than $1/\lambda$ when the model input series has high precision and little noise.

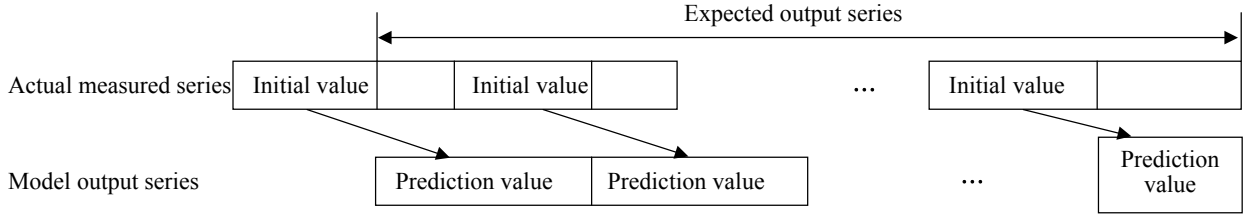

Fig.4 Evaluating model output with multi-stage prediction

Evaluation of genetic individuals

The individual evaluating strategies decide the GP evolution directions. In GPM, the fitness value of individual i is calculated as follows:

$$fitness(i)=\frac{1}{1+precision(i)}/\alpha(h)$$

where $precision(i)=\sum_{k=1}^{n}[x(k)-\hat{x}(k)]$, which denotes the model precision, $\{x(k), k=1,2,\ldots,n\}$ is the model input series and $\{\hat{x}(k), k=1,2,...n\}$ is the model output series. The transformation $1/(1+precision(i))$ makes the individuals with small precision values have high fitness. It can also smooth the large fluctuations of model precisions, which can improve the GP's effectiveness. $\alpha(h)$ is a tuning parameter, with its value decided by the tree depth h. It controls the model complexity and makes the GPM tend to search for those models with simpler structures.

$$\alpha(h)=\begin{cases}\text{abs}(h-h')/2^{h-2} & h\geq 4\\ \text{abs}(h-h')\times 4 & h\leq 2\\ \text{abs}(h-h') & \text{else}\end{cases}$$

where h' denotes the expected model tree depth. The default value is h'=2.5. When the depth is larger than 4, the exponential gene 2^{h-2} will further punish the individual in order to decrease the model complexity.

Model comparison

Can the established model faithfully reflect the system dynamics or not? Most of the modeling methods prove this equivalence through the difference of model output series and expected output series, for example, the MSE criterion. But for chaotic systems, the established model should not only approach the expected output series under specific initial values, but also should have the same dynamic properties as that of the model input series. So it is necessary to introduce some chaotic system's invariants as the criteria of model quality. Here, the largest Lyapunov exponent λ and the correlation dimension D are chosen to serve as such criteria. They can be estimated from time series directly (Rosenstein *et al*., 1993).

Selection of candidate models

Usually, GP can generate a group of models to be further investigated. In addition, the best model output by GP may not be the best one that reflects system dynamics well because of the noise, bias and system's internal complexity. So we should select the most appropriate model from GP's outputs. The model should reveal the system dynamics and the chaotic nature as stated in Section "Model comparison". The selection process is essentially the process of judging which model is the best one to fit the system dynamics. Different from the model output calculation in Section "Evaluating the model output", the model output series here is generated by iterative calculation to arbitrary length from any valid initial values. If the model output has similar chaotic invariants as the measured series does, we can determine that this model reflects the chaotic dynamics well.

MODELING EXPERIMENTS AND DISCUSSION

Logistic series modeling

The Logistic map

$$x(n)=rx(n-1)(1-x(n-1)) \quad x\in(0, 1), n\in N$$

is a typical discrete chaotic map. It exhibits complex behaviors when r>3.5699. This experiment investigates the performance of GPM modeling from known chaotic systems. The model input series are generated from the logistic map with r=4 and initial value $x(0)$=0.2 (Fig.5a). The length of the series is 10000 so that it is long enough to estimate the chaotic invariants. The reconstruction delay τ, embedding dimension m, correlation dimension D, and the largest Lyapunov exponent λ, can be estimated from the logistic series respec-

tively as 8, 4, 3.13 and 0.51. Thus the maximum prediction length is about $L \approx 1/\lambda \approx 2$ steps. Because this series has high data precision and S/N ratio, the L can be set a bit longer. In this experiment it is set to 3. Considering that most chaotic systems are governed by their internal low-dimensional laws, and the embedding dimensions m estimated from time series tend to be larger than the modeling dimension necessary, we can try the modeling dimension from small values first. It is first set to 1 in the experiment.

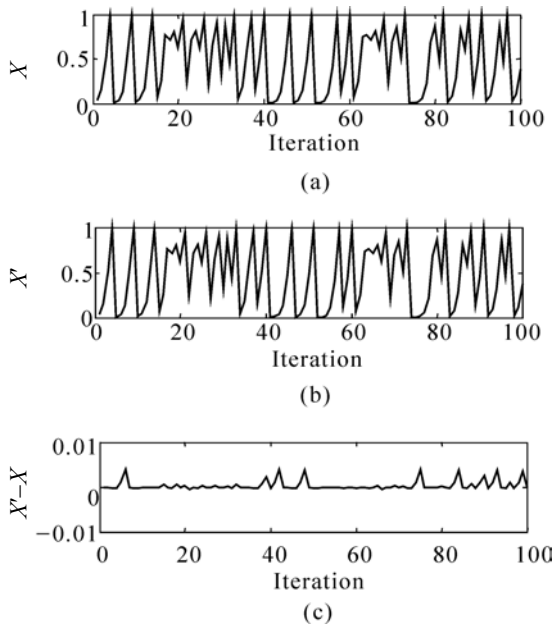

Fig.5 Model's expected output series (a), actual output series (b), and their differences (c)

The model output is generated according to multi-stage prediction algorithm in Section "Evaluating the model output". The maximum prediction length L=3, model dimension m=1

Table 1 lists the candidate models found by GPM on logistic series. The GP population size M=5, maximum generation Gen=10, the crossover probability Pc=0.8, the mutation probability Pm=0.6, the expected tree depth h'=2.5, function set FS={plus, minus, times}, and terminator set TS={$CONSTANT$, $X(1)$}. These individuals are selected because they have relatively high fitness values, and they contain the nonlinear structure that may exhibit chaotic behaviors. For high efficiency, the PSO-NPE algorithm embedded in GP usually has fewer particles and generations. We can increase the particle count and generations in the model-selection stage to get better results. For example, the estimated parameter set of model 4 in Table 1 is A={10.0000, −10.0000, 0.3993} which uses 20 particles and 80 generations, and the MSE between expected output and model output is as low as 0.0739. The model's output and the differences between actual output and expected output are illustrated in Figs. 5b and 5c respectively. The model formula can be re-organized as $x(n)=3.993x(n-1)(1-x(n-1))$, which is quite similar to the logistic model above. The model series also has similar Lyapunov exponent 0.54 as the original series has. This fact confirms the model 4 can reveal logistic dynamics. In addition, the parameter set in model 3 after elaborate estimation is A={−8.2138, −4.7130, 1.1438, 4.0018}, and MSE=0.0704. The model can be re-organized as $x(n)=4.0018x(n-1)-4.0042x2(n-1)$. It is also equiva-

Table 1 Candidate models of logistic series

	Candidate models (Parameter value ignored)
1	plus(times(minus(A(1)*X(1), A(2)), A(3)*X(1)), A(4))
2	plus(times(minus(A(1)*X(1), A(2)), A(3)), A(4)*X(1))
3	plus(times(minus(A(1)*X(1), A(2)*X(1)), A(3)*X(1)), A(4)*X(1))
4	times(plus(A(1), A(2)*X(1)), A(3)*X(1))
5	plus(times(minus(A(1)*X(1), A(2)*X(1)), plus(A(3), A(4))), A(5)*X(1))
6	times(plus(A(1), A(2)*X(1)), times(plus(A(3), A(4)*X(1)), A(5)*X(1)))
7	times(plus(A(1), plus(A(2), A(3)*X(1))), times(plus(A(4)*X(1), A(5)*X(1)), minus(A(6)*X(1), A(7)*X(1))))
8	times(plus(A(1), A(2)*X(1)), plus(A(3)*X(1), A(4)))
9	plus(minus(A(1)*X(1), A(2)), minus(A(3)*X(1), A(4)*X(1)))

A denotes the parameter vector, X denotes the independent variable, and "plus", "times", "minus" denote the arithmetic operator "+", "*", "–" respectively

lent to the logistic model. So the GPM can reconstruct correct models successfully from logistic series.

Chebyshev series modeling

The chaotic Chebyshev-map (Leung and Varadan, 2002) is as follows:

$$x(n)=\cos(1.8\cos^{-1}(x(n-1)))$$

In order to investigate the effect of GPM with noisy input, we pollute the Chebyshev series with white noise up to *SNR*=35 db. The best model achieved by the LS-GP algorithm (Leung and Varadan, 2002) with such input after 51 generations is:

$$x(n)=1.9832\cos(0.6545x(n)+0.0183\sin^{-1}x(n-3))$$

Leung and other authors improved the standard LS-GP to ILS-GP and got the following results (Leung and Varadan, 2002):

$$x(n)=1.0025\cos(1.8003\cos^{-1}(1.0000x(n)))$$

For the same data set, GPM can rapidly converge to the optimal model structure:

$$x(n)=A(1)\cos(A(2)\cos^{-1}(A(3)x(n-1)))$$

The parameter set estimated by PSO is $A=\{0.9954, 1.8024, 1.0000\}$. So we conclude that GPM can get satisfactory results with moderately noise-polluted input series. This is mainly because GPM integrates the results of NTSA. The system's information such as the system dimension and the maximum prediction length can be analyzed in advance. So we need not depend on the modeling algorithm to "search and try" the appropriate dimensions; thus the efficiency and accuracy can be improved.

Some experience on GPM parameter tuning and modeling

Different from other global modeling method such as ANN, the GPM can easily integrate known knowledge and experience about the system, so that it can get better modeling results. Another advantage is that the GPM may find simple and elegant model expressions. But the quality of final results depends heavily on the GPM parameters. Correct and exact knowledge on system dimension, model structure pattern, maximum prediction length, operators in *FS* and *TS* and the quality of the initial population will be beneficial for improving the modeling quality.

During the early stage, GPM may generate large number of invalid models and some meaningless trivial models. But they will be decreased under appropriate evaluation strategies. Editing operator can help to reduce the number of "illegal" individuals. Avoiding too complex structure can also be of some help because complex structures have relatively higher probability to be "illegal". Furthermore, the simple models are often superior to complex models in revealing system dynamic laws.

The nonlinear parameter estimation of the dynamical structures is another difficulty in GPM, because the estimation algorithms often encounter senseless model output such as invalid value, infinity output, etc. Besides these, the searching ranges of parameter space also affect the NPE results, especially under nonlinear and chaotic conditions.

In most cases, the population size and evolution generation need not to be too large. Most of the above experiments have population size of 10–30 and generation number of 10–50. This is because the GPM emphasize model space exploring. It does not require the last population to converge to the strived for optimal model. As described in Section 3.5, the optimal model of GP under the criterion of MSE may not be the most appropriate model.

CONCLUSION

Chaos is based on the principle that simple deterministic laws can generate complex behaviors. GPM provides a "try-and-test" method to reveal simple laws in the model space from the system's external measurements. It can get simple model expressions and can easily integrate known system

knowledge. The experiment results on chaotic data series proved that the GPM can find appropriate models successfully. The models have similar system invariants as the measured series indicates, which shows the effectiveness of this method.

References

Hegger, R., Kantz, H., 2000. Practical Implementation of Nonlinear Time Series Methods, The TISEAN Software Package Online Documentation. http://www.mpiipks-dresden.mpg.de/~tisean.

Jian, X.C., Zheng, J.L., 2002. A chaotic global modeling method based on orthogonal polynomials. *Acta Electronica Sinica*, **30**(1):76-78.

Kantz, H., Schreiber, T., 1997. Nonlinear Time Series Analysis. Cambridge University Press.

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. Proc IEEE Int. Conf on Neural Networks, p.1942-1948.

Koza, J.R., 1990. Genetic Programming, A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. Stanford University Report, Report No. STAN-CS-90-1394, http://www.genetic-programming.com/jkpubs72to93.html#anchor484765.

Leung, H., Varadan, V., 2002. System Modelling and Design Using Genetic Programming. The 1st IEEE International Conference on Cognitive Informatics, Banff, Canada.

Lv, J.H., Lu, J.N., Chen, S.H., 2002. Nonlinear Time Series Analysis and Applications. Wuhan University Press, Wuhan (in Chinese).

Pan, Z.J., Kang, L.S., Chen, Y.T., 1998. Evolutionary Computation. Tsinghua University Press and Guangxi Scientific and Technology Press (in Chinese).

Rosenstein, J.R., Collins, J.J., Luca, C.J., 1993. A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D*, **65**:117-134.

Shi, Y.H., Eberhart, R., 1998. A Modified Particle Swarm Optimizer. Proc IEEE Int. Conf on Evolutionary Computation, p.69-73.

Varadan, V., Leung, H., 2001. Reconstruction of polynomial systems from noisy time series measurements using genetic programming. *IEEE Trans. Industrial Electronics*, **48**(4):742-748.

Xie, X.F., Zhang, W.J., Yang Z.L., 2003. Overview of particle swarm optimization. *Control and Decision*. **18**(2):129-134 (in Chinese).

Wei, R., Lu, J.G., Li, J., Wang, Z.Q., 2002. A new wavelet model for identification of discrete chaotic systems and qualitative analysis of model. *Acta Electronica Sinica*, **30**(1):73-75.