Time Series Perturbation by Genetic Programming

G. Y. Lee
Assistant Professr, Department of Computer and Information Engineering, Young-San University
San 150, Ju-Nam-Ri, Ung-Sang-Eup,
Yang-San-Shi, Kyung-Nam, South Korea
sky@java-tech.com

Abstract- This paper presents a new algorithm that combines perturbation theory and genetic programming for modeling and forecasting real-world chaotic time series. Both perturbation theory and time series modeling have to build symbolic models for very complex system dynamics. Perturbation theory does not work without well-defined system equation. Difficulties in modeling time series lie in the fact that we can't have or assume any system equation. The new algorithm shows how genetic programming can be combined with perturbation theory for time series modeling. Detailed discussions on successful applications to chaotic time series from practically important fields of science and engineering are given. Computational resources were negligible as compared with earlier similar regression studies based on genetic programming. Desktop PC provides sufficient computing power to make the new algorithm very useful for real-world chaotic time series. Especially, it worked very well for deterministic or stationary time series, while stochastic or nonstationary time series needed extended effort, as it should be.

1 Introduction

Time series is a scalar sequence of numerical data, $x_0, x_1, x_2, \ldots$. Time series modeling usually starts from generating vector time series, $\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots$. An appropriately arranged set of the scalar time series data, $x_{t-(n-1)\tau}, \ldots, x_{t-2\tau}, x_{t-\tau}, x_t, x_w$, constitutes a vector. The subscript t stands for current time, τ for delay time (also called lag time or lag spacing), $w = t + T$ for forecast time, T for future time (also called lead time or prediction horizon), and n for embedding dimension of the Euclidean state space where each vector is a point. Determination of these state space parameters for a given time series is critical for good model. However, it is not the scope of this paper. See Weigend 1993 for various data characterization techniques.

Time series modeling is to find the functional approximation $\tilde{f}$ to f in Eq. (1) that relates $x_w^{(i)}$ with remaining components of vector. The approximation error $e^{(i)}$ should be minimal for all vectors, $\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots$.

$$x_w^{(i)} = f\left(\vec{x}_t^{(i)}\right) \cong \tilde{f}\left(\vec{x}_t^{(i)}\right) + e^{(i)}, \; i = 1, 2, 3, \ldots$$

$$\vec{x}_t^{(i)} \equiv \left(x_{t-(n-1)\tau}, \ldots, x_{t-2\tau}, x_{t-\tau}, x_t\right)^{(i)} \in R^n \quad (1)$$

As ARMA (Box 1994) and many other techniques such as GMDH (Ivakhnenko 1971) do, genetic programming applied to symbolic regression (Koza 1994) tries to find explicit model that is written in mathematical symbols. It has achieved interesting performance for stationary scientific time series (Koza 1994, Oakeley 1994, Iba 1994). But, real-world time series is very chaotic and usually highly nonstationary (Box 1994). Also, noise makes it difficult to have practically useful model. Expensive computations resulted in only moderately performing models for them.

The author noted that time series modeling have some similarities with perturbation theory of quantum mechanics (Rae 1992). They need useful approaches for formulating nonstationary or stochastic system dynamics. Of course, the system behaviors are represented differently. Perturbation theory assumes well-defined system behavior, e.g. the wave equation (Rae 1992, Nayeh 1993), while time series modeling should work without such kind of equations.

Note that symbolic regression based on genetic programming (Koza 1994) provides evolutionary ways to do the essentially same works, i.e. formulation of complex system dynamics, as perturbation theory does, even without explicit system equation. The algorithm presented here takes several viewpoints to model complex system dynamics from perturbation theory. And, genetic programming is requested to play the role of ill-defined system equation for chaotic time series.

Section 2 formulates the new algorithm, followed by Section 3 to show application examples to many real-world chaotic time series. Section 4 concludes this paper and lists up topics for further study.

2 Time Series Perturbation Algorithm

2.1 Development

The inherently nonstationary dynamics of the wave equation does not allow for exact solution. Perturbation theory (Nayeh 1993) involves two types of Hamiltonians for such equation. Now, let $\Psi_u(\vec{x})$ be an unperturbed Hamiltonian and $\Psi_p(\vec{x})$ be a perturbed Hamiltonian. Then

$$\Phi(\vec{x}) = \Psi_u(\vec{x}) + \Psi_p(\vec{x}) \quad (2)$$

meets the system equation, i.e. the wave equation. Equation (2) states that nonstationary system behavior can be described by linear combination of Hamiltonians. The same thing holds again for the perturbed Hamiltonian. This means that the perturbed Hamiltonian $\Psi_p(\vec{x})$ can be expanded as another linear combination of different Hamiltonians, and the expansion repeats for all newly available sequence of perturbed Hamiltonians. Perturbation technique provide systematic ways to find the unperturbed Hamiltonians that are integrated into a final solution to the complex system equation. If Eq. (2) meets the system equation, any linear combination of the Hamiltonians also meets the system equation.

Note that the time series carry system dynamics for source system as the wave equation does for a quantum system. Then, we believe that the same procedure for obtaining solutions to quantum system can be applied to time series modeling. Given below is the detailed analogical development of the time series modeling algorithm based on the perturbation theory.

Now, the time series model or the function *f* in Eq. (1) can be expressed as a sum of the unperturbed and perturbed *Hamiltonians,* on the analogy of Eq. (2). That is,

$$f(\vec{x}) = f_u(\vec{x}) + f_p(\vec{x}) \tag{3}$$

Equation (3) can be rewritten with appropriate constants, *a* and *b*

$$f(\vec{x}) = af_u(\vec{x}) + b + f_p(\vec{x}) \tag{4}$$

since if $f_u(\vec{x})$ and $f_p(\vec{x})$ meets the *system equation* (= *system dynamics carried by time series*), their linear combination also meets the equation as Hamiltonians do for the wave equation. The constant *b* is a kind of large number such as the average. Seeing differently, *b* is a deterministic value for every datum in the sequence of time series.

Equation (4) states that time series consist of the unperturbed and the perturbed time series. The unperturbed time series consists of the constant and the variable expressed by the explicit function $f_u(\vec{x})$. The *perturbed* time series is given by

$$f(\vec{x}) - af_u(\vec{x}) - b = f_p(\vec{x}) \tag{5}$$

Here, note that the perturbed time series can be easily calculated if we substract the numuerical values returned by linear combination of the unperturbed time series $f_u(\vec{x})$ from the original time series $f(\vec{x})$.

Equation (5), $f_p(\vec{x})$ represents still another time series to model. We can proceed to model it, again with the same algorithm for the original time series. In general, we will have in theory a sequence of the unperturbed time series, $f_u(\vec{x})^j$ where *j* stands for *j-th* modeling procedure. In the long run, the general form of *time series perturbation model* becomes

$$f(\vec{x}) = \sum_{j=0}^{J} a^j f_u(\vec{x})^j + f_p(\vec{x})^J \tag{6}$$

where the *0-th* unperturbed time series $f_u(\vec{x})^0 = 1$, and $f_p(\vec{x})^J$ is negligible. Now, note that our problem is reduced to how we get the series of the unperturbed time series models. As you may already know, it is by GP (Genetic Programming), see next section.

2.2 Implementations

Everytime we apply the classical GP-based symbolic regression (Koza 1994) to the sequence of the unperturbed time series, we can get the functional forms for $f_u(\vec{x})^0$, $f_u(\vec{x})^1$,..., $f_u(\vec{x})^J$ in Eq.(6). This paper suggests that the constants a^j, $j = 0, 1, 2, \ldots, J$ are determined by the least square regression that makes $f_p(\vec{x})^J$ negligible.

Next several subsections describe some general and special implementation details this paper is based on. For more classic techniques about GP-based symbolic regression, see (Koza 1994).

General Implementation Issues

1) *Construction, Selection, and Integration of Models*

For time series modeling based on genetic programming, population of symbolic forms is subject to genetic evolution. Initially, structures and contents of the symbolic forms are determined at random. Genetic evolution alters the structures and contents of symbolic forms such that they can capture the time series dynamics. Usually, the best symbolic form in a population is taken as a model for the time series. As a corollary, we are expected to have *P* models if we use *P* populations in the evolution.

Then, a question arises. What should we select one among the multiple models to represent the time series dynamics ? A commonsense selection may be the one that produces the minimum approximation error for all time series vectors, See Eq. (1). However, it is very hard to build a perfect model within practical limit on computational resources. The fact is that even the best one selected among *P* multiple models must be only partially successful in capturing the time series dynamics. And, moreover, the other $P - 1$ unselected models do capture partial dynamics of the time series, so that we can save computational resources if we can reuse them in some ways. We may reasonably integrate the partially successful models. Paragraphs below explains how.

Now, let $g_{pp}(\vec{x})$ be the model constructed in population *pp*. Then, the *j-th* unperturbed time series model $f_u(\vec{x})^j$ has, in this paper, the form

$$f_u(\vec{x})^j = \alpha_0 + \sum_{pp=1}^{P} \alpha_{pp} g_{pp}(\vec{x}) \quad (7)$$

where the numerical coefficients for each $g_{pp}(\vec{x})$ are calculated by the least square regression with respect to training data set of the time series. Note that Eq. (6) can be rewritten. Insert Eq. (7) into Eq. (6). We get

$$f(\vec{x}) = \sum_{j=0}^{J} a^j f_u(\vec{x})^j + f_p(\vec{x})^J = \sum_{j=0}^{J} a^j \left[\alpha_0 + \sum_{pp=1}^{P} \alpha_{pp} g_{pp}(\vec{x}) \right]^j + f_p(\vec{x})^J$$

which becomes

$$f(\vec{x}) = \sum_{j=0}^{J} a^j \alpha_0^j + \sum_{j=1}^{J} \sum_{pp=1}^{P} a^j \alpha_{pp}^j g_{pp}(\vec{x})^j + f_p(\vec{x})^J \quad (8)$$

Changing notations for the numerical coefficients, Eq. (8) can be rewritten as

$$f(\vec{x}) = \beta_0 + \sum_{K=1}^{J \times P} \beta_K g_K(\vec{x}) + f_p(\vec{x})^J \quad (9)$$

From (3) and (9), we can see that the unperturbed time series model up to *J-th* modeling procedure, see Section 2.1, takes the form

$$f_u(\vec{x})^J = \beta_0 + \sum_{K=1}^{J \times P} \beta_K g_K(\vec{x}) \quad (10)$$

2) *Interpretation of the Perturbation Modeling by GP*

We are now at a point to clarify overall procedure of the time series perturbation modeling procedure based on genetic programming. First, the vector time series, Eq. (1), are divided into three data regions for training, validation, and forecasting. For the first modeling procedure, genetic programming is run with respect to the training region and we have the unperturbed time series model as expressed by Eq. (10), $J = 1$.

Once the unperturbed time series model is determined, the *j-th* model building procedure is formally over. Ideally, the model would capture the time series dynamics and therefore can forecast the time series beyond the training region. But, our model would fail only after a few acceptable forecasts due to several practical reasons. For example, the training region may not be sufficient to provide information needed to capture the dynamics beyond it. Even if the training region contains sufficient information, computational resources may fall short of what is necessary to make full use of the information.

There may be several ways to keep the model performance as high as possible beyond the training region. The most straightforward one would be to re-construct the model with respect to newly defined training region that includes the latest available true time series data. But, the new model should be constructed in time to become useful forecaster for the time series.

In this paper, the numerical coefficients in the linear combination of the models are simply updated with respect to the data region that is close to forecast point. For example, assume that we have 200 data in the training region. Then, the 201st datum is forecasted with the numerical coefficients calculated with respect to data between 1st and 200th positions of the time series. The model is given by Eq. (6) if j =1, and by Eq. (10) if j is greater than 1. For the 202nd datum, the numerical coefficients are updated with respect to data between 2nd and 201st positions. We can proceed further only if the true time series datum at position v is known before we try to forecast the position $fp = v + 1$. The value $S = fp - v \geq 1$ is termed as *the impact step* in this paper. The data region that comes after the training region is called the validation region. The model with changing coefficients is used to forecast the data in the validation region. After forecasting the last datum in the validation region, the forecasting performance, e.g. Eq. (11), Eq. (12) is recorded. The forecasting performance in the validation region is an important criterion to determine if we stop or proceed to another perturbation modeling procedure by genetic programming. For this reason, we call it *the validation performance*.

Another modeling procedure starts if the validation performance got improved as compared with former modeling procedure. By default, the first modeling procedure is followed by the second modeling procedure. Otherwise, the perturbation modeling procedure stops on the assumption that the model has started to capture spurious or excessive perturbation such as noise or disturbance. This type of criterion to terminate learning or modeling algorithm was used in the field of artificial neural network (Geman 1992) and called *early stopping policy*. Section 4 summarizes above interpretation of the proposed GP-based time series perturbation modeling procedure.

Several Implementation Issues

1) Performance of individual model

A model or any individual in a population should be given numerical performance value that measures how well the time series is approximated or simulated by the symbolic forms represented by the individual or the model. Popular performance value is the normalized mean squared error (Weigend 1993), and the coefficient of variation (Iba 1994). They are defined by

$$CV(N) = \frac{1}{\bar{x}} \left[\frac{1}{N} \sum_{i=1}^{N} \left(x^{(i)} - \tilde{x}^{(i)} \right)^2 \right]^{0.5} \quad (11)$$

$$NMSE(N) = \frac{\sum_{i=1}^{N} \left(x^{(i)} - \tilde{x}^{(i)} \right)^2}{\sum_{i=1}^{N} \left(x^{(i)} - \bar{x} \right)^2} \cong \frac{\sum_{i=1}^{N} \left(x^{(i)} - \tilde{x}^{(i)} \right)^2}{N \hat{\sigma}^2} = \frac{MSE(N)}{\hat{\sigma}^2} \quad (12)$$

where $\tilde{x}^{(i)}$ and $x^{(i)}$ are model evaluation and true datum at the position i. $\bar{x}$ and $\hat{\sigma}^2$ denote the sample average and sample variance of the time series. N is the number of data over which $NMSE(N)$ or $CV(N)$ is calculated.

MSE stands for Mean Squared Error. Model fitness of an individual is given by the inverse of $NMSE(N)$ or $CV(N)$ in this paper.

2) Super population and Migration

Genetic programming designed for time series perturbation modeling in this paper has a special-purpose population called the super population of which sole service is to select and keep the best individuals, $g_{pp}(\vec{x})$ in Eq. (7), from each of multiple populations. The best individual from a population is allowed to replace a super population member at the end of each generation if and only if its fitness excels that of the member being replaced.

Migration does occur between the multiple populations that undergo evolution. Only those members in the super population that survived the whole generations become the final model $g_{pp}(\vec{x})$ in Eq. (7). The super population is different from the multi-agent team (Luke 1996) where agents or team members are somehow engaged in the evolutionary processes.

3) Three Symbolic Anomalities

Symbolic structures and contents that can not be translated into mathematically valid equations define *mathematical anomality*. Division-by-zero, negative arguments given to square root are typical examples of the mathematical abnormality. Symbolic structures and contents that cause computer software error such as the overflow or underflow define *computational anomality*. The last symbolic anomality is the *semantic replication*. When multiple symbolic forms are mathematically equivalent, they are semantic replications of each other. For example, *(+ x3 (sin (/ x2 x2)))* is the semantic replication of *(+ x3 (sin 1)) = x3*. Especially, semantic replication between the best individual from a population and the super population member should be avoided at the end of each generation and each modeling procedure. If not, mathematical error of singular matrix occurs during calculation of the numerical coefficients in Eq. (6) or Eq. (10) using the least square regression.

Any type of symbolic anomalities consumes computational resources because they blemish genetic diversity in a population. Also, they cause unexpected error in the program run. There should be systematic techniques to detect and repair the cases of symbolic anomalities. If any individual in a population has symbolic anomality, its fitness is arbitrarily assigned a very small value in this paper to reduce the chance of breeding offspring in the next generation.

4) Derived Terminal Set

Function set and terminal set for regression problem based on genetic programming are very important because they are basic symbols to represent time series datum in explicit form. Terminal set provides argument symbols for function symbols. For example, the symbolic form $(\boldsymbol{sin}\, x_3)$ has the argument symbol x_3 for the function symbol ***sin***.

The concept of *derived terminal set* (DTS) is introduced in this paper for the purpose of saving computational resources for initializing and processing various kinds of primitive functions such as $(\boldsymbol{sin}\, x_3)$, $(\boldsymbol{cos}\, x_3)$. DTS is a collection of symbols that represent primitive functions. DTS is used, along with the terminal set and the function set, for creating the initial population for genetic programming.

Desirable characteristics of DTS include the capability of approximating a mathematical function when linearly combined. Time series modeling is in a sense an approximation of the unknown function f in Eq. (1). Orthogonal functions (Sansone 1991) satisfy these conditions. In addition to trigonometric functions, we made DTS by applying Tschebyshev function to elements of $\vec{x}_t$ in Eq. (1). The Tschebyshev terminal $T_{order,i}$ is given by

$$T_{order,i} \equiv \boldsymbol{cos}\left[order \times \boldsymbol{arccos}\left(x_i^{**}\right)\right] \quad x_i^{**} \in \vec{x} \tag{13}$$

where *order* is an integer, the double asterisks in x_i^{**} indicate that x_i should be appropriately adjusted to be in the interval [-1, 1]. The following linear mapping is used.

$$x_i^{**} = sx_i - t, \quad s = 2\left(x_i^{MAX} - x_i^{MIN}\right)^{-1},$$
$$t = 1 + 2x_i^{MIN}\left(x_i^{MAX} - x_i^{MIN}\right)^{-1} \tag{14}$$

In Eq. (14), x_i^{MAX} and x_i^{MIN} are the global maximum and minimum. Now, let $x_i^{\boldsymbol{max}}$ and $x_i^{\boldsymbol{min}}$ be local maximum and minimum observable in the training region. The global x_i^{MAX} and x_i^{MIN} are estimated by introducing arbitrary expansion ratio η. Equation (15) assume that the global interval is $2\eta + 1$ times broader than the local interval observable in the training region.

$$x_i^{MAX} = x_i^{\boldsymbol{max}} + \eta \cdot \Delta, \quad x_i^{MIN} = x_i^{\boldsymbol{min}} - \eta \cdot \Delta,$$
$$\Delta = x_i^{\boldsymbol{max}} - x_i^{\boldsymbol{min}} \tag{15}$$

3 Applications

3.1 Human Body Blood Flow Dynamics

Time series data obtained by solving the Mackey – Glass equation have been used by several works (Oakeley 1994, Iba 1994, Casdagli 1989). The equation simulates the nonlinear dynamics of human blood flow, and is given by

$$\frac{dx_t}{dt} = \frac{bx_{t-\Delta}}{1 + x_{t-\Delta}^c} - ax_t,$$
$$x_{t+1} = (1-a)x_t + bx_{t-\Delta}\left(1 + x_{t-\Delta}^c\right)^{-1} \tag{16}$$

With appropriately assigned constant values a, b, and Δ, the difference equation in Eq. (16) is used to generate very chaotic time series from the initial random seeds of

predetermined size, about 40. Table below compares results of this study with those of earlier works.

Forecating Performance	Earlier Works		This Study
	Casdagli 1993	Iba 1994	
NMSE(20)	0.063	0.031	0.019
NMSE(30)	0.159		0.009
NMSE(40)	0.316	0.158	0.004
NMSE(50)	0.631	0.371	0.003
NMSE(60)	0.990	0.617	0.005

Time series perturbation algorithm outperforms the earlier works based on genetic programming. See section 17.5.1 and section 17.5.3.2, Lee 1999, for further details.

3.2 Santa Fe and ASHRAE Time Series Competitions

Santa Fe Institute and ASHRAE (American Society for Heating, Refrigerating, and Air-conditioning Engineers) held worldwide competitions for time series analysis and forecast. Time series was chosen from very stationary to highly volatile system. See Weigend 1993 for more details on the categorization and characterization of the time series data for the competition. This paper has fixed computational parameters for them to the followings:

- Number of populations = 5, Population size = 30, Generation Limit = 9, Maximum number of perturbation modeling allowed = 5
- Function set = { +, –, ×, /, sin, cos, exp, log, expt }, Terminal set = { $x_{t-(n-1)\tau},\ldots,x_{t-2\tau},x_{t-\tau},x_t$ } ∪ { $T_{order,i} \mid order = 1 \sim 10$ for each x_i }, Depth of regression trees = Initial 6, after-crossover 18
- Crossover fraction = 0.8, Mutation fraction = 0.1, Reproduction fraction = 0.1, Maximum number of migrating individuals allowed = 1 % of total individuals
- Lag spacing = 1, Lead time = 1, Embedding dimension = 1 or 4, Impact step = 1
- Total number of time series vector = 400, Training region ***T*** = first 200 data, Validation region ***V*** = next 100 data after region ***T***, Forecast region ***F*** = the last 100 data after region *V*, Terminating NMSE = 0.01.

Table below summarizes the modeling and forecasting performances of the perturbation modeling explored in this study. Symbols for each time series are used to save space here, and they are : Sun = Time series for the true solar beam isolation flux, Energy = Energy consumption rate in a building, Laser = Intensity fluctuation of NH3 laser, Heart = Heart rate of a human patient, Curr. = Currency exchange rate for Swiss franc vs. US dollor, Part. = Quantum particle position in 4D potential well, and Star = Surface brightness of a white dwarf star, PG1195. See Weigend 1993. For Santa Fe competition series, numerical values added to the symbols are used to represent the embedding dimensions.

Com-	Time	Performances in each Region			
		T	***V***	***F*** (Forecasting)	
		Modeling With DTS		With DTS	No DTS
ASHRAE	Sun	0.005	0.001	0.002	0.002
	Energy	0.032	0.039	0.054	0.075
Santa Fe	Laser, 1	0.007	0.018	0.015	0.016
	Laser, 4	0.001	0.003	0.004	0.004
	Heart, 1	0.065	0.190	0.165	>> 1
	Heart, 4	0.178	0.259	0.355	Infinite
	Curr., 1	1.542	1.666	1.247	35.88
	Curr., 4	8.364	7.878	15.39	Infinite
	Part., 1	0.023	0.033	0.076	12.54
	Part., 4	0.699	0.354	0.154	Infinite
	Star, 1	0.006	0.008	0.033	0.028
	Star, 4	0.002	0.001	0.002	0.002

Note that Sun, Laser, and Star are all from physics systems that have stationary system dynamics. These time series are categorized into the *stationary* or *deterministic time series* (Weigend 1993, Box 1994). Stationary or deterministic time series is well modeled and forecasted by the perturbation modeling based on genetic programming. The terminating NMSE for forecasting was achieved before the maximum number of perturbation modeling ran out.

Embedding dimension *n*, lag spacing τ, and lead time *T* are simplistically assumed and kept fixed in this study. Two different values of the embedding dimension, *n* = 1 and 4, are chosen only to see if the embedding dimension works differently for stationary and nonstationary time series. Perturbation modeling worked much better for deterministic time series when we use increased embedding dimension. Of course, there must be a limit on the embedding dimension over which model performance deteriorates.

On the other hand, the more a time series is nonstationary, the more unstable and irregular dynamics it will have. At the extreme, time series dynamics may be pure random. So, the increased embedding dimension for nonstationary time series might introduce increased randomness in dynamics that is hard to capture within practical, small computational resource limit. This point of view explains the different performances in the above table between the stationary and the other nonstationary time series.

The foregoing table also reveals that DTS has minimal effects on the performances for stationary time series, while it does contribute to improve performances for nonstationary time series. Introduction of DTS saves computational resources to generate, evaluate, and process primitive functions.

The algorithm for time series perturbation modeling based on genetic programming uses the update extension (Smith 1993) of time series beyond the training region. In the update extension, we must know true values of time series continuation that is ***S***, the impact step, position behind the forecast position. The state space (Kailath 1996)

parameters were simplistically assumed for all aforementioned tables. Analysis to determine such parameters is not the scope of this paper. Moreover, extremely small quantity of training data, 200, makes very coarsely constructed state space. The competition contestants were required to use the runaway extension (Weigend 1993, Smith 1993) that do not rely on the true values to make forecasting extension of time series, and it is best successful when mass amount of data is used to construct very dense state space.

From the practical points of view, each method of forecasting extension has the long and the short. Runaway extension requires expensive modeling burdens but the forecasting is easy with established self-extending models. It is beneficial to use the runaway extension when computational time is limited for timely forecast but there is a lot of stored data. On the other hand, the update extension is good when we have little data but good computing power. With modern high-speed computer, it will be more practical to use update extension.

3.3 Major US Economic Time Series

Economic time series constitute a very difficult but highly important category of time series. But, most short-term economic time series are extremely nonstationary, which has caused ever-continuing disputes on the existence of any order or chaotic structure we can simulate or model. See Dechert 1996 for related discussion.

Time series perturbation modeling based on genetic programming was tested with respect to 18 major US economic time series data available from http:// www.economagic. com.

Embedding dimensions are all fixed to 1 based on the experience from Santa Fe competition that increased embedding dimension may be risky for nonstationary economic time series with limited computational resources.

Computational parameters were same as those for ASHRAE and Santa Fe competitions. Here, the concept of *pre-modeling* is introduced. Pre-modeling is the application of the algorithm with very small data size, i.e. 100 training data, for the purpose of grasping data characteristics. Pre-modeling is based on the experience from Santa Fe competition. That is, the more stationary a time series is, the higher the model performance would be.

Federal fund rates FFR, Japanese yen to US dollar currency exchange rate YENDOL, and 30-year Treasury Constant Maturity 30YTCM were relatively difficult to forecast. They are relatively more nonstationary, and need increased computational resources.

Table below shows how the forecasting performances for FFR improve with increasing size of training data. All other parameters were fixed. NT stands for NMSE in the training (= modeling) region, and NF for the forecasting region, i. e., 100 data after the end of NT.

Data	τ =1 Month		τ =6 Month		τ =12 Month	
Size	NT	NF	NT	NF	NT	NF
100	0.096	0.176	0.534	10.62	0.812	3.602
200	0.022	0.082	0.244	0.485	0.503	0.748
300	0.030	0.012	0.206	0.232	0.352	0.627

Evolution with increased number of training data, i. e. high density state space results in better modeling and forecasting performances. The more dense the state space is, the more system dynamics can be captured (Smith 1993).

4 Conclusion

This paper presented a new algorithm that combines perturbation theory (Rae 1992, Nayeh 1993) with genetic programming. Perturbation theory provides efficient ways to get solution to complex system equation that usually does not allow for exact solution. Genetic programming provides evolutionary processes to get symbolic forms that model time series dynamics. In the time series perturbation algorithm, the sequence of time series plays the role of the system equation in perturbation theory. Symbolic models obtained by genetic programming plays the role of Hamiltonians in perturbation theory.

A sequence of the unperturbed time series models is obtained for a time series much like the unperturbed Hamiltonians were obtained for a wave equation by perturbation techniques. They are linearly combined with numerical coefficients calculated with respect to given time series. Forecasting beyond the training region is performed based on the update extension which requires numerical coefficients updated by the least square regression with respect to the latest data. The update extension of forecast data, and the introduction of DTS saved computation resources. Stationary time series are more easily modeled and forecasted.

The algorithm was successfully applied to many real-world chaotic time series, covering physics to economic ones. Noticeable performance was achieved even with the simplistically assumed values of state space parameters and the limited computational resource.

The algorithm should be coupled with time series characterization techniques to get optimized set of the state space parameters, if any. In this paper, a pre-modeling or the run of the algorithm with reduced size of training data is suggested to classify data characteristics. Time series with good forecasting performances in the pre-modeling were assumed to be stationary, based on the application experience for Santa Fe competition. The new algorithm produced consistent results on the time series data characteristics with the analyses by other techniques (Weigend 1993). Effects of various genetic programming parameters on the modeling performance should be studied further.

Bibliography

Weigend, A. S. and Gershenfeld, N. A., Eds (1993), Time Series Prediction – Forecasting the future and Understanding the Past, SFI Studies in the Science of Complexity, Vol. XV, Addison Wesley Publishing Co.

Box, G. E. P. , Jenkins, G. M. and Reinsel, G. C. (1994), *Time Series Analysis*, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1994.

Ivakhnenko, A. G. (1971), "Polynomial Theory of Complex Systems," *IEEE Trans. Syst. Man Cybern.*, vol. 1(4), pp. 364-378, 1971.

Koza, J. R. (1994), *Genetic Programming II*, MIT Press.

Oakeley, H. (1994), "Two Scientific Application of Genetic Programming: Stack Filters and Non-Linear Equation Fitting to Chaotic Data," in *Advances in Genetic Programming*, K. E. Kinnear Jr., Eds. MIT Press, pp. 369-389.

Iba, H., de Garis, H. and Sato, T. (1994), "Genetic Programming using a Minimum Description Length Principle," in *Advances in Genetic Programming*, K. E. Kinnear Jr., Eds. MIT Press, pp. 265-284.

Rae, A. I. M. (1992), *Quantum Mechanics*, 3 rd ed., University of Birmingham, UK, IOP Publishing Ltd.

Geman, S. et al. (1992), "Neural Networks and the Bias / Variance Dilemma," *Neural Computation*, vol. 4, pp. 1-58.

Luke, S. (1996), and L. Spector, "Evolving Teamwork and Coordination with Genetic Programming," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, MIT Press, pp. 150-156.

Sansone, G., Sansome, G. and Diamond, A. H. (1991), *Orthogonal Functions*, Dover Publications.

Casdagli, M. C. (1989), "Nonlinear prediction of chaotic time series," *Physics D*. 35, pp. 335-356.

Lee, G. Y. (1999), "Genetic Recursive Regression for modeling and forecasting real-world chaotic time series," in *Advances in Genetic Programming,* vol. 3. MIT Press, Chapter 17.

Kreider, J. F. (1993), *results.asc*, ASHRAE Competition ftp site, ftp.cs.colorado.edu/pub/energy-shootout, 1993.

Smith, L. A. (1993), "Does a meeting in Santa Fe imply chaos ?," in *Time Series Prediction – Forecasting the Future and Understanding the Past*, A. S. Weigend, and N. A. Gershenfeld, Eds., SFI Studies in the Science of Complexity, vol. XV. Addison-Wesley Publishing Co, pp. 323-343.

Kailath, T. (1980), *Linear Systems*, Englewood Cliffs, NJ: Prentice Hall.

Dechert, W. D. (1996), *Chaos Theory in Economics : Methods, Models and Evidence*, International Library of Critical Writings in Economics, No. 66. Edward Elgar Pub.

Nayeh, A. H. (1993), and A. H. Nayfeh, *Introduction to Perturbation Techniques*, John Wiley & Sons.