



Evolutionary Modeling of Systems of Ordinary Differential Equations with Genetic Programming

HONGQING CAO, LISHAN KANG, AND
YUPING CHEN

chq@rjgc.whu.edu.cn; kang@whu.edu.cn

State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, P. R. China

State Key Laboratory of Parallel and Distributed Processing, P. R. China

JINGXIAN YU

jxyu@263.net

Institute of Electrochemistry, Department of Chemistry, Wuhan University, Wuhan 430072, P. R. China

Received May 18, 1999; Revised March 8, 2000

Abstract. This paper describes an approach to the evolutionary modeling problem of ordinary differential equations including systems of ordinary differential equations and higher-order differential equations. Hybrid evolutionary modeling algorithms are presented to implement the automatic modeling of one- and multi-dimensional dynamic systems respectively. The main idea of the method is to embed a genetic algorithm in genetic programming where the latter is employed to discover and optimize the structure of a model, while the former is employed to optimize its parameters. A number of practical examples are used to demonstrate the effectiveness of the approach. Experimental results show that the algorithm has some advantages over most available modeling methods.

Keywords: evolutionary modeling, genetic programming, genetic algorithm, system of ordinary differential equations, higher-order ordinary differential equation

1. Introduction

Many complex systems and nonlinear phenomena that change over time exist in engineering, economic management, natural sciences, social sciences and many other fields. Examples include price fluctuations, the change of temperature during a chemical process, weather changes, population growth, and so on. Researchers have long sought reasonable mathematical models for those systems based on the observed data so as to provide a basis for system analysis, design and prediction. In the past, the traditional modeling methods have been based on classical analysis and statistical analysis. With the development of system sciences and computer techniques, various modeling methods came up, such as system identification, system dynamics, fuzzy mathematics, the gray system method [1] and so on. When using these methods, modelers are required to have rich knowledge, experience and ability in many fields. The development of the assumptions of a model is a crucial step of modeling. To make reasonable assumptions, people need not only specific domain details but also abundant imagination, sensitive observation and judgement, which depends on human intelligence largely. The calculation and analysis of a model also involves complicated computation processes which require the

modelers to have sufficient professional knowledge and wide applied mathematics knowledge. All of these issues contribute to difficulties in modeling some practical systems by using traditional methods. In addition, these methods can only deal with a very limited range of problems, mainly linear systems and some special nonlinear systems.

In recent years, the study of evolutionary algorithms (EAs) has gained the interest of many researchers in a wide range of fields. EAs are adaptive methods for solving computational problems in many fields, which mimic the process of biological evolution and the mechanisms of natural selection and genetic variation. They use suitable coding to represent possible solutions to a problem, and guide the search by using genetic operators and the principle of “survival of the fittest.” Due to their merits of self-adaptation, self-organization, self-learning, intrinsic parallelism and generality, EAs have succeeded in solving a large number of problems in machine learning, pattern recognition, economic prediction, optimization control, parallel processing and many other domains [2, 3].

In view of the drawbacks in modeling the complex systems by the use of most available methods, we consider using EAs to approach the modeling problem of complex systems. That is, in the case that limited information is known to a system, to partly replace human intelligence with computational intelligence in some steps of traditional modeling, including the development of assumptions, the construction and the calculation of a model, to complete the whole modeling task. This is the idea of so-called evolutionary modeling (EM).

EM uses genetic programming (GP) as the main approach, which is a new branch of EA introduced in the 1990s [4–6]. GP is an extension of genetic algorithm (GA) [7, 8] in which the genetic population consists of computer programs of varying sizes and shapes. In standard GP, computer programs can be represented as parse trees, where a branch node represents an element from a function set, which usually contains some arithmetic operations, logic operations and elementary functions of at least one argument, and a leaf node represents an element from a terminal set, which usually contains variables, constants and functions of no arguments. These symbolic programs are subsequently evaluated by running them on a set of “fitness cases.” Fitter programs are selected for recombination to create the next generation by using genetic operators, such as crossover and mutation. This step is iterated for some number of generations until the termination criterion of the run has been satisfied.

At present our research mainly focuses on the EM of ordinary differential equations (ODE) including systems of ODE (SODE) and higher-order ODE (HODE), to build the dynamic ODE models for multi-dimensional systems and one-dimensional systems respectively. The reason for using an ODE model is that this kind of model can describe the dynamic properties of a system which changes with time quite well and can predict the future states of the system very conveniently. Recently, some researchers in the field of evolutionary computation have studied the ODE models of some complex systems from different aspects [9–11].

This paper is organized as follows. We describe the EM problem of SODE in Section 2. In this section, we first give the mathematical description of the modeling problem of SODE. Afterwards we propose a hybrid evolutionary modeling

algorithm (HEMA) to approach the EM problem of SODE. The structure of the HEMA and its detailed description follows. Two practical examples are used to test the effectiveness of the HEMA for SODE and their experimental results are also given. We study the EM problem of HODE in Section 3. Similarly, in this section, we first define the modeling problem of HODE in mathematics. Then we present the structure of the HEMA for HODE and explain those details which are different from the HEMA for SODE. We apply the algorithm to three typical time series to examine the effectiveness of the HEMA for HODE. In Section 4, we improve the HEMA for HODE and present a real-time HEMA for HODE to perform the real-time prediction task of time series and two practical examples are used to test its effectiveness. Finally, conclusions and future research are summarized in Section 5.

2. Evolutionary modeling of system of ODE

2.1. Problem statement

We first define the norm of a matrix A where $A \in \mathbb{R}^{m \times n}$ as

$$\|A\| = \sqrt{\sum_{i=1}^m \sum_{j=1}^n (a_{ij})^2} \tag{1}$$

Suppose a dynamic system can be described by n correlated functions, $x_1(t), x_2(t), \dots, x_n(t)$ and a series of observed data collected at the time $t_i = t_0 + i^* \Delta t (i = 0, 1, 2, \dots, m - 1)$ can be written as the following form

$$X = \begin{pmatrix} x_1(t_0), & x_2(t_0), & \dots, & x_n(t_0) \\ x_1(t_1), & x_2(t_1), & \dots, & x_n(t_1) \\ \vdots & \vdots & \dots & \vdots \\ x_1(t_{m-1}), & x_2(t_{m-1}), & \dots, & x_n(t_{m-1}) \end{pmatrix} \tag{2}$$

where t_0 denotes the starting time, Δt denotes the interval between two observations, $x_j(t_i) (j = 1, 2, \dots, n)$ denotes the observed value of variable x_j at the time t_i .

If we denote $X(t) = [x_1(t), x_2(t), \dots, x_n(t)]$, $f(t, X) = [f_1(t, X), f_2(t, X), \dots, f_n(t, X)]$ where $f_j(t, X) = f_j(t, x_1(t), x_2(t), \dots, x_n(t)) (j = 1, 2, \dots, n)$ is the composite function of elementary functions involving variables $x_i (i = 1, 2, \dots, n), t$ and the function space defined by those functions can be denoted as F , the modeling problem of SODE is to find the model, a system of first-order differential equations having the general form of

$$dX^*/dt = f(t, X^*) \tag{3}$$

such that

$$\min\{\|X^* - X\|, \forall f \in F\} \quad \text{where} \tag{4}$$

$$\|X^* - X\| = \sqrt{\sum_{i=0}^{m-1} \sum_{j=1}^n [x_j^*(t_i) - x_j(t_i)]^2}$$

and the values of $x_i(i = 1, 2, \dots, n)$ at the next τ time steps

$$\begin{pmatrix} x_1(t_m), & x_2(t_m), & \cdots, & x_n(t_m) \\ x_1(t_{m+1}), & x_2(t_{m+1}), & \cdots, & x_n(t_{m+1}) \\ \vdots & \vdots & & \vdots \\ x_1(t_{m+\tau-1}), & x_2(t_{m+\tau-1}), & \cdots, & x_n(t_{m+\tau-1}) \end{pmatrix} \quad [5]$$

can be predicted based on the model.

2.2. Hybrid evolutionary modeling algorithm for system of ODE

2.2.1. Hybrid evolutionary modeling algorithm. When we apply standard GP to the modeling problem of ODE, the search of the model structure and the estimation of model parameters are performed concurrently during the EM process. Those randomly generated parameters can have great effects on the performance of a model since their values affect the rate of change of the right-hand function of an ODE directly and can cause a system to be unstable if chosen inappropriately. In this case, one major problem arises. Since the fitness value of a model depends largely upon the values of its parameters, a model with a favorable structure will have a great probability of being eliminated from the population during the evolution if their parameters are not properly selected. Consequently it is unlikely for us to obtain a highly accurate model for the system. Moreover the evolutionary process can be slow due to the large number of generations needed, as well as suffering from the “premature convergence” phenomenon. As such, we consider embedding the parameter optimization process in the evolutionary modeling process, which enables us to speed up the evolutionary process and to find better models. There are a great many traditional methods for solving nonlinear parameter optimization problems, such as direct search methods, Hooke-Jeeves methods, Nelder-Mead methods, gradient methods and variable metric methods [12], but they are only applicable to specific problem types and require restricted conditions to be placed on the model, such as being continuous, derivable or single-peaked. Because our ODE models are generated randomly and may consist of multiple equations, with both the parameters contained in each equation and the total groups of parameters varying, we have no way of optimizing their parameters by using traditional methods. In view of those issues mentioned above, we present a hybrid evolutionary modeling algorithm (HEMA) to approach the EM problem of ODE. Its main idea is to embed a genetic algorithm (GA) in genetic programming (GP) where GP is employed to discover and optimize the structure of a model, while a GA is employed to optimize its parameters. The two main processes, namely the evolutionary modeling process to optimize the structure of models based on GP and the parameter optimization process to optimize the parameters of a model based on a GA, accompanied by the data preprocessing, the simplification and normalization of models, and the use of the system to perform prediction, constitute the framework of the HEMA.

2.2.2. Algorithm structure. The structure of the HEMA for SODE can be described as follows:

PROCEDURE HEMA for SODE
<pre> begin input observed data matrix $X^{(0)}$; preprocess $X^{(0)}$ and get $X^{(1)}$; GEN := 0; initialize the SODE model population $P(\text{GEN})$; repeat $P(\text{GEN}) :=$ simplification and normalization of models in $P(\text{GEN})$; $P_{GA}(\text{GEN}) :=$ parameter optimization of models in $P(\text{GEN})$ using GA; $P_{GP}(\text{GEN}) :=$ structure optimization of models in $P_{GA}(\text{GEN})$ using GP; fitness evaluation of models in $P_{GP}(\text{GEN})$; $P(\text{GEN} + 1) :=$ select [$P_{GP}(\text{GEN})$]; GEN := GEN + 1; until termination criterion; output the best evolved model in the current generation; make system prediction; end </pre>

2.2.3. Detailed description of algorithm

2.2.3.1. Preprocessing of data. As for the original observed data, we apply low-pass filtering to eliminate noise at high frequencies by means of the multi-dimensional discrete Fourier transform.

2.2.3.2. Simplification and normalization of models. The simplification of models simplifies the tree structures of each individual in the model population by replacing subtrees which consist of arithmetic operations between constants. This operation is performed on all individuals in every generation, which affects the number of parameters to be optimized but does not change the fitness of individuals.

The normalization of models adjusts the structure of subtrees in the model whose roots are “+” (plus) or “*” (multiplication) and whose left branches or right branches are constants to ensure that the constant always lies on the right of the “+” or the “*” in the S-expression of the model. This operation is useful to distinguish the model structures correctly so that “ $a + x$ ” and “ $x + a$ ” or “ $a * x$ ” and “ $x * a$ ” will not be regarded as different structures doing the optimization process redundantly.

2.2.3.3. Structure optimization of models using GP

2.2.3.3.1. Encoding. We extend the representation of single tree in standard GP to a vector of binary trees denoted as (T_1, T_2, \dots, T_n) where n is the number of

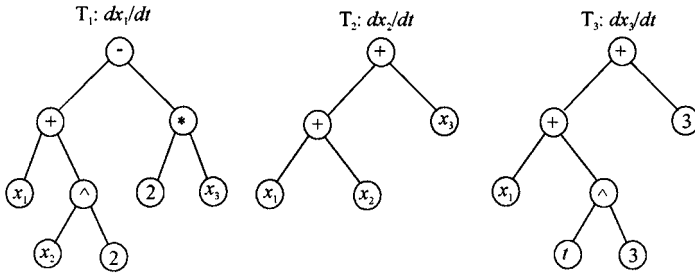


Figure 1. An example of the representation of a SODE model.

equations in the SODE. For example, a SODE model with the form of

$$\begin{cases} dx_1/dt = x_1 + x_2^2 - 2x_3 \\ dx_2/dt = x_1 + x_2 + x_3 \\ dx_3/dt = x_1 + t^3 + 3 \end{cases} \tag{6}$$

can be represented as a vector of binary trees (T_1, T_2, T_3) illustrated in Fig. 1. Besides this, the maximum depth per tree is restricted by a constant D and the complexity of a model is measured by the number of nodes contained in each tree.

2.2.3.3.2. *Fitness evaluation.* Given the observed data $X^{(0)}$ written as the form of Eq. [2], we do some preprocessing to it as described in Section 2.2.3.1 and get a new matrix $X^{(1)}$. For one individual p written as the form of

$$\begin{cases} dx_1/dt = f_1(t, x_1, \dots, x_n) \\ dx_2/dt = f_2(t, x_1, \dots, x_n) \\ \vdots \\ dx_n/dt = f_n(t, x_1, \dots, x_n) \end{cases} \tag{7}$$

in the model population, its fitness value $fitness(p)$ can be calculated as follows:

```

PROCEDURE cal_fitness
begin
  let  $X^*$  be two  $m \times n$  matrices, assign the first row of  $X^{(1)}$  to that of  $X^*$ ;
  for  $i := 2$  to  $m$  do
    begin
      integrate the system Eq. [7] for a step with some numerical
        integration methods by taking the  $(i - 1)$ st row of  $X^{(1)}$  as the
        initial conditions;
      assign the solution to the  $i$ th row of  $X^*$ ;
    end
     $\Delta X := X^{(1)} - X^*$ ;
     $fitness(p) := \|\Delta X\|$ ;
  end
```

Obviously, here the lower the fitness is, the better is the individual. Furthermore, due to the diversity of the SODE generated randomly, some of them may not be stable and will give rise to overflow during the fitness calculation. In this case, we return a large number to the fitness value as a penalty so that these unreasonable models can be eliminated from the population soon.

As the fitness evaluation takes the largest amount of time during the EM of SODE, a crucial issue is how to improve its speed and accuracy. To determine which kinds of numerical methods are most suitable to evaluate the fitness, we have done some numerical experiments with seven methods, namely the modified Euler method, the Runge-Kutta method with adaptive stepsize and fixed stepsize respectively, the Gill method with variable stepsize, the Merson method, the Hamming method and the Adams predictor-corrector method. As we have to choose the stepsize carefully during the integration, it is more convenient to use methods with fixed stepsize than to use methods with variable stepsize. Furthermore, our experimental results indicate that use of high-order methods can only increase the computational cost of processing individuals significantly but will not improve the quality of solutions noticeably. In view of these considerations, we use the modified Euler method with fixed stepsize to do the numerical integration during the evaluation of fitness. As for the determination of its stepsize, we have also experimented with different stepsizes of 0.001, 0.005, 0.01, 0.025, 0.05 and 0.1 on several data sets. We observe that the errors of the solution increase significantly when the stepsize is greater than 0.05, while too small a stepsize will be prohibitively time consuming but offer little improvement in the quality of solutions. For a tradeoff, we set the stepsize equal to 0.01.

2.2.3.3.3. Selection strategy and genetic operators. We use tournament selection with sample size of 4 in the HEMA. An elitism strategy is also adopted.

Since each individual is represented as a vector of trees, there are two levels of crossover, namely the vector-level crossover and the tree-level crossover. Consider parent a denoted as $(T_1^{(a)}, T_2^{(a)}, \dots, T_n^{(a)})$ and parent b denoted as $(T_1^{(b)}, T_2^{(b)}, \dots, T_n^{(b)})$. The vector-level crossover is performed by selecting one tree from $T_k^{(a)}$ and $T_k^{(b)}$ randomly as the tree $T_k^{(c)}$ of the offspring c denoted as $(T_1^{(c)}, T_2^{(c)}, \dots, T_n^{(c)})$ for each $k(k = 1, 2, \dots, n)$. The tree-level crossover performs the following operations on each pair of $T_k^{(a)}$ and $T_k^{(b)}$ ($k = 1, 2, \dots, n$). Randomly select a node within each tree as a crossover point, swap the subtrees rooted at the crossover points and produce two new trees, then use either of them as the tree $T_k^{(c)}$ of offspring c denoted as $(T_1^{(c)}, T_2^{(c)}, \dots, T_n^{(c)})$ on condition that its maximum depth does not exceed D .

We use the deterministic dynamic adaptation method [13] to alter the mutation rate of individuals according to fitness so that the fitter individuals change only in a small range (with a small mutation rate) but the less fit individuals in a wide range (with a large mutation rate). Given parent i denoted as $(T_1^{(i)}, T_2^{(i)}, \dots, T_n^{(i)})$, its mutation rate $p_m(i)$ is defined as

$$p_m(i) = 0.3 * (1 - f_{\min}/f_i) \quad [8]$$

where f_{\min} is the fitness value of the best individual in the current generation and f_i is the fitness value of i . The mutation of i begins by randomly selecting a $k(1 \leq k \leq n)$, and performs the following operations on the tree $T_k^{(i)}$. Randomly select a node within the tree as the mutation point, replace the subtree rooted at the mutation point with a tree randomly generated, thus produce a new tree $T_k^{*(i)}$. Then the offspring can be written in the form of $(T_1^{(i)}, T_2^{(i)}, \dots, T_k^{*(i)}, \dots, T_n^{(i)})$.

2.2.3.4. *Parameter optimization of models using GA*

2.2.3.4.1. *Encoding.* At the beginning of this process, we examine whether the model structure has been optimized in the current generation. If so, do nothing with it. Otherwise, we check all the constants contained in each tree $T_i(1 \leq i \leq n)$ first, namely count their number l_i and record their positions in the tree. Then each individual in the parameter population can be represented as a vector of n row vectors with varying dimensions of l_i :

$$(C_1, C_2, \dots, C_n)^T \quad \text{where} \quad C_i = (c_{i1}, c_{i2}, \dots, c_{il_i}) \quad [9]$$

and each element c_{ij} of vector C_i is encoded as a floating point number.

2.2.3.4.2. *Fitness evaluation.* Consider one individual in the parameter population denoted as $(C_1, C_2, \dots, C_n)^T$. Before the evaluation of its fitness, we shall return to the original model denoted as (T_1, T_2, \dots, T_n) and replace all the constants of each tree $T_i(1 \leq i \leq n)$ with the corresponding elements of vector C_i . Then the same procedure as in Section 2.2.3.3.2 is followed to calculate its fitness value.

2.2.3.4.3. *Selection strategy and genetic operators.* We adopt the same selection strategy as Section 2.2.3.3.3.

Consider parent a denoted as $(C_1^{(a)}, C_2^{(a)}, \dots, C_n^{(a)})^T$ where $C_i^{(a)} = (c_{i1}^{(a)}, c_{i2}^{(a)}, \dots, c_{il_i}^{(a)})$ and parent b denoted as $(C_1^{(b)}, C_2^{(b)}, \dots, C_n^{(b)})^T$ where $C_i^{(b)} = (c_{i1}^{(b)}, c_{i2}^{(b)}, \dots, c_{il_i}^{(b)})$. The crossover operator performs the following operations on each pair of $C_i^{(a)}$ and $C_i^{(b)}(i = 1, 2, \dots, n)$ based on the integrated arithmetical crossover. Randomly select a crossover point $k(1 \leq k \leq l_i)$ first; produce two new elements $c_{ik}^{(c)}$ and $c_{ik}^{(d)}$ from $c_{ik}^{(a)}$ and $c_{ik}^{(b)}$ as follows:

$$c_{ik}^{(c)} = \alpha c_{ik}^{(a)} + (1 - \alpha) c_{ik}^{(b)} \quad c_{ik}^{(d)} = \alpha c_{ik}^{(b)} + (1 - \alpha) c_{ik}^{(a)} \quad [10]$$

where α is a random number ranging from 0 to 1. Thus offspring c and d can be written in the form of

$$c : (C_1^{(c)}, C_2^{(c)}, \dots, C_n^{(c)})^T \quad \text{where} \quad C_i^{(c)} = (c_{i1}^{(a)}, c_{i2}^{(a)}, \dots, c_{ik}^{(c)}, \dots, c_{il_i}^{(a)})$$

$$d : (C_1^{(d)}, C_2^{(d)}, \dots, C_n^{(d)})^T \quad \text{where} \quad C_i^{(d)} = (c_{i1}^{(b)}, c_{i2}^{(b)}, \dots, c_{ik}^{(d)}, \dots, c_{il_i}^{(b)}).$$

Consider a parent denoted as $(C_1, C_2, \dots, C_n)^T$ where $C_i = (c_{i1}, c_{i2}, \dots, c_{il_i})_{(1 \leq i \leq n)}$. The mutation performs the following operations on each C_i based on multi-level mutation [14, 15]. Randomly select a mutation point $k(1 \leq k \leq l_i)$ and produce a new element c_{ik}^* from c_{ik} as follows:

$$c_{ik}^* = c_{ik} \pm 0.1 \times 2^{-j} \times (\max - \min) \tag{11}$$

where j is a random integer in the range of 0 to 15, $[\min, \max]$ is the field of definitions of c_{ik} , “+” or “-” is chosen randomly with equal probability. Then the offspring can be written as the form of $(C_1^*, C_2^*, \dots, C_n^*)^T$ where $C_i^* = (c_{i1}, c_{i2}, \dots, c_{ik}^*, \dots, c_{il_i})$.

2.2.3.5. *System prediction.* Once the best evolved model is obtained in one run, we then take the last line of $X^{(1)}$ as the initial conditions and integrate the SODE with the form of Eq. [7] for several steps with the modified Euler method to get the predicted series of the system.

2.3. Modeling experiments

2.3.1. *Parameter settings and measures.* To examine the effectiveness of the HEMA, we apply it to two practical examples with different number of variables, one being one-dimensional and the other being three-dimensional. Twenty runs are conducted independently for each example. All the experiments are performed on a Pentium II (266 Mhz) using Visual C++ compilers. The parameter settings are shown in Table 1.

In addition, to measure the modeling results of different models, we define the fitting error (FE) and the prediction error (PE) of variable x as

$$FE = \sqrt{\sum_{i=0}^{m-1} (\hat{x}_i - x_i)^2} \quad PE = \sqrt{\sum_{i=m}^{m+\tau-1} (\hat{x}_i - x_i)^2} \tag{12}$$

Table 1. Parameter settings of the modeling experiments for SODE

Structure optimization (GP)	
Function set	F = {+, -, *, /, ^, sin, cos, exp, ln} where x^n symbolizes $x^n (0 < n < 5)$
Terminal set	T = { x_1, \dots, x_n, t, c } where n is the number of equations in the SODE and c is a random constant
Control parameters	Example 1: Popsize=100 Max_Depth=4 Max_Gen=100 Example 2: Popsize=50 Max_Depth=3 Max_Gen=50
Parameter optimization (GA)	
Control parameters	Popsize = 20 $p_c = 0.6$ $p_m = 0.3$ Termination criterion: The fitness value of the best individual has remained unchanged for 3 generations

where x_i is the observed value of x at the time t_i , \hat{x}_i the fitting value and the predicted value of x for FE and PE respectively, m and τ the number of data points to be fitted and to be predicted respectively. To be specific, for Example 1, $m = 15$, $\tau = 2$; for Example 2, $m = 100$, $\tau = 10$.

2.3.2. Example 1: Population model. The experimental data are taken from [16] giving the populations of the United States from 1790 to 1950 (see the “actual data” column in Table 2 where the unit of x_i is million). We take the actual data from 1790 to 1930 as historical data to build ODE models and predict the population of 1940 and 1950.

The best models in twenty runs we have obtained by using the HEMA are

$$\text{Model I : } dx/dt = (1.384360 - (x * t)) * ((x + 1.548616) * 20.574356)$$

$$\text{Model II : } dx/dt = x * (30.912327 - (x * 0.151000))$$

$$\text{The simplified form : } dx/dt = 30.912327x - 0.151000x^2$$

$$\text{Model III : } dx/dt = ((x * 21.926580) + 111.811684) * \cos(x * t).$$

In [16], according to the assumption of the classical Logistic model

$$dx/dt = r(1 - x/x_m)x \tag{13}$$

Table 2. The modeling results of HEMA models and the classical model for Example 1

Actual data		Model I		Model II	Model III	Logistic model
Year	No. i	x_i	\hat{x}_i	\hat{x}_i	\hat{x}_i	\hat{x}_i
1790	1	3.9	3.535357	3.535357	3.535357	3.9
1800	2	5.3	5.156120	4.768738	5.634524	5.279377
1810	3	7.2	7.570995	7.238487	7.929233	7.128606
1820	4	9.6	9.915323	9.632812	10.166016	9.593163
1830	5	12.9	13.087546	12.867659	13.206610	12.852110
1840	6	17.1	17.293678	17.148132	17.261953	17.117052
1850	7	23.2	22.678036	22.613745	22.490780	22.623427
1860	8	31.4	30.352379	30.378368	30.010643	29.609574
1870	9	38.6	40.267639	40.369537	39.829693	38.279471
1880	10	50.2	49.184601	49.320061	48.743813	48.748665
1890	11	62.9	62.618629	62.753670	62.287102	60.981043
1900	12	76.0	76.934494	77.020882	76.814316	74.735514
1910	13	92.0	91.309525	91.324280	91.436218	89.549867
1920	14	106.5	107.662506	107.611816	108.040558	104.784066
1930	15	123.2	122.202255	122.157379	122.736099	119.722268
1940	16	131.7	137.306310	137.396286	137.936386	133.702390
1950	17	150.7	150.029821	150.477142	151.287262	146.22599
	FE		3.069454	3.091522	3.350870	5.703330
	PE		5.646193	5.700646	6.263979	4.901667

where r and x_m are two constants, r and x_m are determined as 0.31 and 197 respectively based on the actual data and the obtained model is

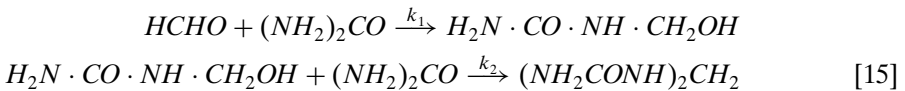
$$dx/dt = 0.31x - 0.001537x^2 \quad [14]$$

The modeling results of these models are listed in Table 2.

From Table 2, we can see that multiple superior ODE models can be found by running the HEMA. Although those models differ enormously in structures, their FEs are all smaller than that of Logistic model.

Interestingly, the simplified form of Model II is in agreement with the Logistic model. As the form of an ODE model is related to the unit of time t , the models for the same data set may look different under different time units. In the HEMA, we use the modified Euler method with the stepsize 0.01 to do the numerical integration whose time unit is 1% of the Logistic model; therefore there is a difference of 100 times between the two models while the results are identical in essence. It is worth noticing that the HEMA model II is discovered by the computer automatically based on the dynamic data of the system while the model Eq. [14] is built based on the assumption of the classical model. Their agreement indicates that the HEMA can serve as a powerful tool in the discovery of scientific laws for dynamic data.

2.3.3. Example 2: Chemical reaction model. The reaction between formaldehyde (X_1) and carbamide in the aqueous solution gives methylol urea (X_2) which continues to react with carbamide and form methylene urea (X_3). The reaction equations are



The reactions occur at 308.15K and under the excessive carbamide with the concentration ($c_{2(0)}$) of 2 mol·dm⁻³. The concentration of chlorhydric acid as a catalyst is 0.0008 mol·dm⁻³ and the initial concentration of formaldehyde ($X_{1(0)}$) is 0.1 mol·dm⁻³. As a kind of typical consecutive reaction, the concentrations of the three components in the system satisfy the following SODE

$$\begin{cases} dX_1/dt = -k'_1 X_1 \\ dX_2/dt = k'_1 X_1 - k'_2 X_2 \\ dX_3/dt = k'_2 X_2 \end{cases} \quad [16]$$

where $k'_1 = k_1 c_{2(0)}$, $k'_2 = k_2 c_{2(0)}$ with $k_1 = 0.007 \text{ dm}^3 \cdot \text{mol}^{-1} \cdot \text{min}^{-1}$, $k_2 = 0.021 \text{ dm}^3 \cdot \text{mol}^{-1} \cdot \text{min}^{-1}$. According to the exact solution of the consecutive reaction

$$\begin{cases} X_1 = X_{1(0)} e^{-k'_1 t} \\ X_2 = \frac{k'_1 X_{1(0)}}{k'_2 - k'_1} (e^{-k'_1 t} - e^{-k'_2 t}) \\ X_3 = X_{1(0)} - X_1 - X_2 \end{cases} \quad [17]$$

we calculate the concentrations of X_1 , X_2 , X_3 every other minute within 110 minutes after the reactions occur and take them as simulated data of our experiment. From them, the first 100 points are used as modeling samples and the next 10 points are used as test samples to evaluate the prediction results of the model.

The best kinetic model we have obtained in 20 runs is

$$\begin{cases} dX_1/dt = (X_1 - (X_1 + 1.40035)) * X_1 \\ dX_2/dt = (X_2 * 3.307096 - (X_1 + t)) * (-1.355543) \\ dX_3/dt = ((X_2 * 4.069420) + t) + (-0.002812) \end{cases} \quad [18]$$

which can be simplified as

$$\begin{cases} dX_1/dt = -1.400035X_1 \\ dX_2/dt = 1.355543(X_1 + t) - 4.482911X_2 \\ dX_3/dt = 4.069420X_2 + t - 0.002812 \end{cases} \quad [19]$$

Its modeling results are listed in Table 3 and its fitting and prediction curves are shown in Fig. 2.

As shown in Fig. 2, the fitting curves of the best evolutionary model and the simulated curves with respect to the concentrations of three components are nearly coincident and more surprisingly, the predicted values also coincide very well with the simulated values. This can be observed from Table 3 clearly. The difference between the simulated value and the predicted value for X_1 , X_2 or X_3 appears only in the fourth digit after the decimal point. This demonstrates the effectiveness of the HEMA in modeling the complex systems of chemical reactions by SODE. Additionally, by comparing the exact model Eq. [16] after the replacement of k'_1 , k'_2 with corresponding values with the evolutionary model Eq. [19], one notices that their structures are rather similar. As for the equation dX_1/dt , they have the same structure in essence despite the difference of 100 times between their coefficients

Table 3. Modeling results of the best evolutionary model for Example 2

Time (min)	X_1 (mol·dm ⁻³)		X_2 (mol·dm ⁻³)		X_3 (mol·dm ⁻³)	
	Simulated value	Predicted value	Simulated value	Predicted value	Simulated value	Predicted value
100	0.024660	0.024659	0.011580	0.011605	0.063760	0.063766
101	0.024317	0.024317	0.011439	0.011489	0.064244	0.064256
102	0.023979	0.023978	0.011300	0.011373	0.064721	0.064742
103	0.023645	0.023645	0.011162	0.011258	0.065193	0.065223
104	0.023317	0.023316	0.011025	0.011143	0.065659	0.065699
105	0.022993	0.022992	0.010889	0.011029	0.066119	0.066171
106	0.022673	0.022673	0.010754	0.010916	0.066573	0.066638
107	0.022358	0.022357	0.010620	0.010804	0.067022	0.067100
108	0.022047	0.022047	0.010488	0.010693	0.067466	0.067558
109	0.021740	0.021740	0.010356	0.010582	0.067903	0.068011
FE	0.000004		0.00018		0.000055	
PE	0.000002		0.000452		0.000191	

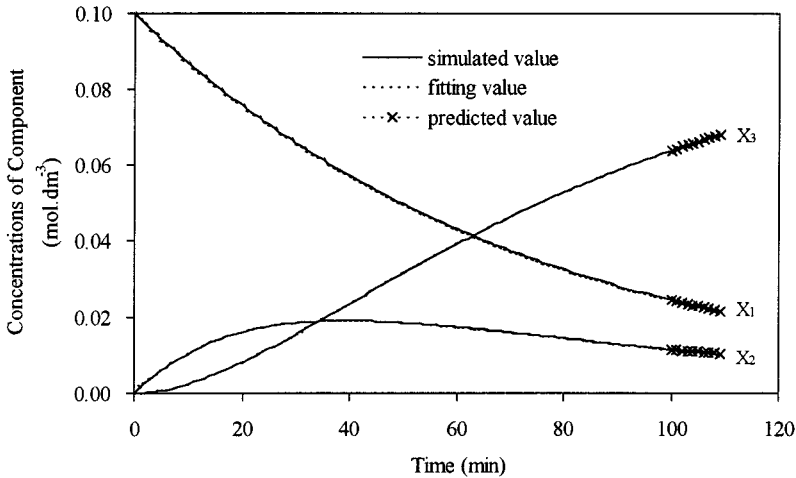


Figure 2. The fitting and prediction curves of the best evolutionary model for Example 2.

due to the fact that we use the modified Euler method with the stepsize 0.01 to do the numerical integration whose time unit is 1% of the exact model. As for the equations dX_2/dt and dX_3/dt , they contain the same variables X_1 , X_2 and the corresponding coefficients are almost identical (likewise, there is a difference of 100 times) except that the evolutionary model has some additional t items and constant item.

3. Evolutionary modeling of higher-order ODE

In traditional time-series analysis, what people study most is one-dimensional dynamic systems and the three kinds of models people have used most are Autoregressive (AR) models, Moving Average (MA) models and ARMA models [17, 18]. Some variations based on those three models are also studied, such as Autoregressive Integrated Moving Average (ARIMA) models [19], Bilinear models [20], Threshold Autoregressive (TAR) models [21] and TARMA models [22]. Most of those models are linear models which can only reflect a very limited range of dynamic processes. Considering that the ARMA Models are linear difference equations in essence, we present a new idea of modeling one-dimensional dynamic systems by HODE models instead of by the ARMA models.

3.1. Problem statement

We first define the norm of a n -dimensional column vector A as

$$\|A\| = \sqrt{\sum_{i=1}^n a_i^2} \quad [20]$$

Suppose that a series of observed data collected from a one-dimensional system $X(t)$ at the past m time steps can be written as

$$X = \begin{pmatrix} x(t_0) \\ x(t_1) \\ \vdots \\ x(t_{m-1}) \end{pmatrix} \quad [21]$$

where $t_i = t_0 + i^* \Delta t (i = 0, 1, 2, \dots, m - 1)$, t_0 denotes the starting time, Δt denotes the interval between two observations. The modeling problem of HODE for the dynamic system $X(t)$ is to find a model of n th-order ODE

$$\begin{aligned} x^{(n)}(t) &= f(t, x(t), x'(t), x''(t), \dots, x^{(n-1)}(t)) \quad \text{with the initial conditions} \\ x^{(i)}(t_0) &= x^{(i)}(0) \end{aligned} \quad [22]$$

to describe the system such that $\|X^* - X\|$ is minimized and the values of $X(t)$ at the next τ time steps

$$\{x(t_m), x(t_{m+1}), \dots, x(t_{m+\tau-1})\} \quad [23]$$

can be predicted based on the model Eq. [22] where

$$\|X^* - X\| = \sqrt{\sum_{i=0}^{m-1} [x^*(t_i) - x(t_i)]^2} \quad [24]$$

X^* is a value vector of the solution $X^*(t)$ of Eq. [22], and f is a composite function composed of some elementary functions including triangle functions, logarithm functions, exponential functions and power functions. The structure of a HODE model defined here is flexible and can take the form of a complex nonlinear ODE with variable coefficients.

We define m as the number of modeling samples, τ as the number of steps for prediction. It is usually called short-term prediction for $\tau = 1$, long-term prediction for $\tau \geq 2$. For brevity, the n th-order ODE model with the form of Eq. [22] is called ODE(n) model in following sections.

3.2. Hybrid evolutionary modeling algorithm for higher-order ODE

3.2.1. Algorithm structure. As the initial value problem of a HODE having the form of Eq. [22] can be converted into that of a SODE having the form of

$$\begin{cases} y'_1 = y_2 \\ y'_2 = y_3 \\ \vdots \\ y'_{n-1} = y_n \\ y'_n = f(t, y_1, y_2, \dots, y_n) \end{cases} \quad \text{with the initial conditions} \quad [25]$$

$$\begin{cases} y_1(t_0) = x(t_0) \\ y_2(t_0) = x'(t_0) \\ \vdots \\ y_n(t_0) = x^{(n-1)}(t_0) \end{cases}$$

by the variable substitution

$$y_1 = x, y_2 = x', y_3 = x'', \dots, y_n = x^{(n-1)}, \quad [26]$$

the EM problem of HODE is equivalent to the EM problem of SODE in essence. We only need make some modifications to the HEMA for SODE to implement the automatic modeling of HODE. The structure of the HEMA for HODE can be described as follows:

```

PROCEDURE HEMA FOR HODE

begin
  input original time series X(0);
  preprocess X(0) and get X(1);
  input the order of HODE models;           {*}
  compute the conversion matrix Y of X(1);  {*}
  GEN := 0;
  initialize the HODE model population P(GEN);  {*}
repeat
  P(GEN) := simplification and normalization of models in P(GEN);
  PGA(GEN) := parameter optimization of models in P(GEN)
  using GA;
  PGP(GEN) := structure optimization of models in PGA(GEN)
  using GP;
  fitness evaluation of models in PGP(GEN);  {*}
  P(GEN + 1) := select [PGP(GEN)];
  GEN := GEN + 1;
until termination criterion;
  output the best evolved model in the current generation;
  make system prediction;                   {*}
end
    
```

The differences between the algorithm and the HEMA for SODE in Section 2.2.2 are marked by “*.” We give their detailed explanations in the next section.

3.2.2. Detailed explanation of algorithm

3.2.2.1. Calculation of conversion matrix Y In order to calculate the approximate values of x in a time series from t_0 to t_{m-1} by means of numerical integration of SODE, thus to evaluate the fitness of the model subsequently, we first convert the HODE with the form of Eq. [22] into a SODE with the form of Eq. [25] by the variable substitution Eq. [26] and compute the conversion matrix Y of $X^{(1)}$:

$$Y = \begin{pmatrix} y_1(t_0), & y_2(t_0), & \cdots, & y_n(t_0) \\ y_1(t_1), & y_2(t_1), & \cdots, & y_n(t_1) \\ \vdots & \vdots & & \vdots \\ y_1(t_{m-1}), & y_2(t_{m-1}), & \cdots, & y_n(t_{m-1}) \end{pmatrix} \tag{27}$$

Obviously, here Y is equivalent to the preprocessed matrix $X^{(1)}$ in the HEMA for SODE. If we denote $Y_i = (y_i(t_0), y_i(t_1), \dots, y_i(t_{m-1}))^T$, then $Y_1 = X^{(1)}$, and Y_i for $i = 2, 3, \dots, n$, which are the $(i - 1)$ st-order derivatives of x in a time series from t_0 to t_{m-1} respectively, can be calculated approximately by means of numerical differentiation. For example, for $n \leq 4$, we can use the following formula of order h^2 error to calculate the values of x', x'', x''' , (i.e. y_2, y_3, y_4) from t_0 to t_{m-1} .

- forward difference approximate formula:

$$\begin{aligned} x'_i &= \frac{-x_{i+2} + 4x_{i+1} - 3x_i}{2h} + O(h^2) \\ x''_i &= \frac{-x_{i+3} + 4x_{i+2} - 5x_{i+1} + 2x_i}{h^2} + O(h^2) \\ x'''_i &= \frac{-3x_{i+4} + 14x_{i+3} - 24x_{i+2} + 18x_{i+1} - 5x_i}{2h^3} + O(h^2) \end{aligned} \tag{28}$$

(to calculate the values of x', x'', x''' at t_0)

- central difference approximate formula:

$$\begin{aligned} x'_i &= \frac{x_{i+1} - x_{i-1}}{2h} + O(h^2) \\ x''_i &= \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} + O(h^2) \\ x'''_i &= \frac{x_{i+2} - 2x_{i+1} + 2x_{i-1} - x_{i-2}}{2h^3} + O(h^2) \end{aligned} \tag{29}$$

(to calculate the values of x', x'', x''' from t_1 to t_{m-3})

- backward difference approximate formula:

$$\begin{aligned}x'_i &= \frac{3x_i - 4x_{i-1} + x_{i-2}}{2h} + O(h^2) \\x''_i &= \frac{2x_i - 5x_{i-1} + 4x_{i-2} - x_{i-3}}{h^2} + O(h^2) \\x'''_i &= \frac{5x_i - 18x_{i-1} + 24x_{i-2} - 14x_{i-3} + 3x_{i-4}}{2h^3} + O(h^2)\end{aligned}\quad [30]$$

(to calculate the values of x' , x'' , x''' at t_{m-2} , t_{m-1})

3.2.2.2. *Encoding of a higher-order ODE model.* Once a HODE is converted into a SODE, we notice that the only difference between two HODE models is the n th equation

$$y'_n = f(t, y_1, y_2, \dots, y_n) \quad [31]$$

namely the right-hand function of a HODE. When initializing the model population, the HEMA generates some individuals randomly and each individual is represented as a binary tree. For example, given a fourth-order ODE

$$x^{(4)} = 3x''' + \sin(x'') - tx' + xe^t \quad [32]$$

its corresponding equation

$$y'_4 = 3y_4 + \sin(y_3) - ty_2 + y_1e^t \quad [33]$$

can be represented as a binary tree shown in Fig. 3.

3.2.2.3. *Fitness evaluation.* Suppose that the corresponding SODE of an arbitrary individual p in the HODE model population has the general form of Eq. [25], then the fitness of p can be calculated as follows:

```

PROCEDURE cal_fitness
begin
  let  $X^*$  and  $\Delta X$  be both  $n$ -dimensional column vectors,  $Y^*$  be a  $(m + 1) \times n$  empty matrix, assign the first row of  $Y$  to that of  $Y^*$ ;
  for  $i := 2$  to  $m$  do
    begin
      integrate the system Eq. [25] for a step with some numerical
        integration methods by taking the  $(i - 1)$ st row of  $Y$  as the initial
        condition; assign the solution to the  $i$ th row of  $Y^*$ ;
    end
     $X^* := Y^*_1$ ;
     $\Delta X := X^{(1)} - X^*$ ;
     $fitness(p) := \|\Delta X\|$ ;    { $Y^*_1$  denotes the vector composed of the first
                                column of  $Y^*$ }
  end

```

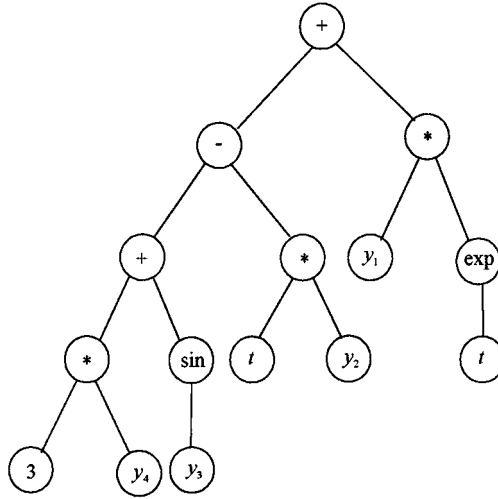


Figure 3. An example of the representation of a HODE model.

When determining which numerical methods will be feasible and suitable for evaluating the fitness, we assume that since the original data are collected at equal intervals, it is not feasible to use integration methods with variable stepsize. On the other hand, it is necessary that Y_1^* should change with different models by using the numerical method so that the evolutionary algorithm can work. Suppose that the order of a HODE is n . Considering the available integration methods, we notice that, when using the modified Euler method, Y_1^* will not change with HODE models any longer when n is greater than 2; while using the fourth-order Runge-Kutta method with fixed stepsize, Y_1^* will remain unchanged when n is greater than 4. Hence, the maximum order of the HODE models we can build by using these two methods are 2 and 4 respectively. To build ODE models with higher order, other numerical methods must be introduced. But an ODE model with too high order is usually impractical in real world problems. In our experiments, we build ODE(1) models, ODE(2) models, ODE(3) models and ODE(4) models for three examples of time series by using the fourth-order Runge-Kutta method with fixed stepsize 0.01 respectively.

3.2.2.4. System prediction. Once the best evolved model is obtained in one run, we then take the last row of Y as the initial conditions, integrate the corresponding system of ODE for several steps by using the fourth-order Runge-Kutta method with stepsize 0.01 and get the predicted series of Y^* . The first column of Y^* is just the predicted series of the dynamic system based on the model.

In addition, as each HODE model is represented as a single tree, the crossover and the mutation of individuals are performed in tree-level using standard GP operators. Moreover the dimension and the number of the parameters contained in a HODE model are much smaller than a SODE model. The GA used to optimize the parameters of a model is the same as Section 2.2.3.4.

3.3. Modeling experiments

3.3.1. Parameter settings. To examine the effectiveness of the HEMA for HODE in time-series analysis and prediction, we choose three typical practical examples of time series to do the modeling experiments. Among the three examples, the first two are most famous in traditional time-series analysis which are used by many researchers as the evidence to prove that their approaches are superior to others due to making better predictions. We build ODE(1) models, ODE(2) models, ODE(3) models and ODE(4) models for each example. Ten runs are conducted independently for HODE models with different order. All the experiments are performed on a Pentium II (266 Mhz) using Visual C++ compilers. The parameter settings are shown in Table 4.

In addition to FE and PE defined in Section 2.3.1, the following measures are applied to compare the modeling results of HODE with different order for all three examples:

- average fitting error (AFE) and average prediction error (APE) which are the mean values of FE and PE of the HODE models obtained in ten runs respectively. Here for Example 3, $m = 170$, $\tau = 6$; for Example 4, $m = 110$, $\tau = 4$; for Example 5, $m = 220$, $\tau = 6$.
- average number of nodes (AN_{nodes}) which is the mean value of the number of nodes of the HODE models obtained in ten runs.
- number of successes (N_{succ}), namely the number of runs in which the best evolved model can give reasonable predictions. If the best evolved model in one run can not make system predictions at all or its prediction error are enormously large, we declare it a failure; or else a success.

3.3.2. Example 3: Sunspot numbers. The experimental data are taken from [17] giving the annual sunspot numbers over 176 years, from 1749 to 1924. We take the observed data of the first 170 years as historical data to build ODE(1) models, ODE(2) models, ODE(3) models and ODE(4) models respectively, and predict the values of the last 6 years.

Table 5 shows the statistical results of ten runs for the HODE models with different order built for Example 3. From the fact that the rate of success is rather low for the ODE(4) model (only succeeded twice in ten runs), we infer that it is not appropriate to use fourth-order ODE models to describe this time series. In addi-

Table 4. Parameter settings of the modeling experiments for HODE

Structure optimization(GP)	
Function set	As in Table 1
Terminal set	$T = \{y_1, \dots, y_n, t, c\}$ where n is the order of a HODE and c is a random constant
Control parameters	Popsize = 50 Max_Depth = 4 Max_Gen = 50
Parameter optimization (GA)	
Control parameters	As in Table 1

Table 5. The statistical results for Example 3 (10 runs)

Model	ODE(1)	ODE(2)	ODE(3)	ODE(4)
AFE	260.657312	102.560301	26.474702	93.686328
APE	99.586979	30.452745	71.277601	32.405675
AN _{nodes}	8.1	9.1	7.8	8
Mean time (sec)	3371	3992	2770	2356
N _{succ}	10	10	9	2

tion, comparing the AFE and the APE of the other three models, we see that both kinds of error are the largest for the ODE(1) model. For the ODE(3) model, the AFE is quite small while the APE is large. For the ODE(2) model, the AFE and the APE are rather moderate. If we choose a model with emphasis on describing a system, the ODE(3) model is the best. While the main objective of modeling a system is to make predictions, rather than to fit the observed data only, we consider that the ODE(2) model is preferable to the ODE(3) model to describe and predict the time series. In fact, its order is in agreement with G. U. Yule’s AR(2) model [17]

$$x_t = 1.32x_{t-1} - 0.63x_{t-2} + a_t \quad (\sigma_a^2 = 236.63) \tag{34}$$

which is a second-order linear difference equation in essence.

We choose the best ODE(2) model in ten runs to illustrate the effectiveness of the HEMA. Table 6 presents its modeling results and Fig. 4 depicts its fitting and prediction curves.

As shown in Table 6 and Fig. 4, the model can fit the observed data quite well. As for the predicted values, the first four are good and the last two are not so good.

3.3.3. Example 4: Quantity of leopard cats in Canada. The experimental data are taken from [18] giving the quantity of leopard cats in Canada from 1831 to 1944. Many statisticians have shown great interests in the dynamic data. We now take the observed data of the first 110 years as historical data to build HODE models and predict the values of the last four years.

As the amplitude of the original data changes greatly, we first compute values of their common logarithms and take the transformed data as input to build HODE models.

The statistical results of ten runs for Example 4 are shown in Table 7.

Obviously, the ODE(3) model is most desirable to describe the system, as both its AFE and APE are much smaller than those of the other three models. We show

Table 6. The best ODE(2) model for Example 3

Evolutionary solution	$\begin{cases} dy_1/dt = y_2 \\ dy_2/dt = (\ln(y_2) - (y_1 - 57.668163)) * (y_1 * 55.339455) \\ x'' = 55.339455x(\ln x' - x + 57.668163) \end{cases}$					
Equivalent ODE						
FE	102.390884					
PE	10.011124					
Observed value	63.6	37.6	26.1	14.2	5.8	16.7
Predicted value	60.484272	40.621162	25.946901	16.748144	11.863738	10.627171

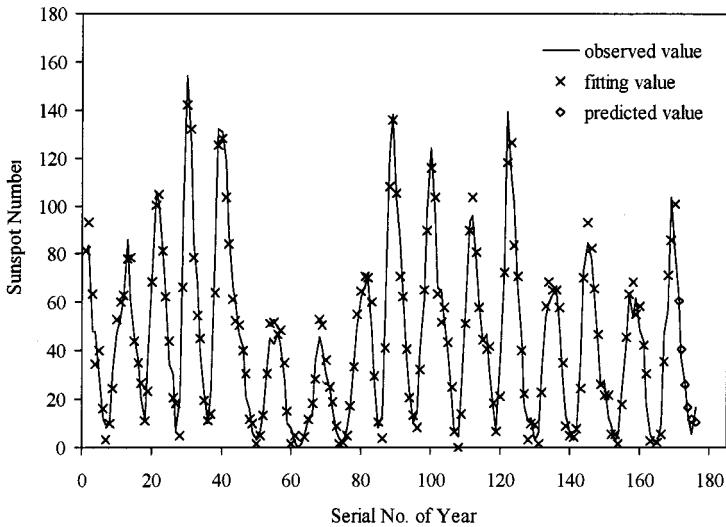


Figure 4. The fitting and prediction curves of the best ODE(2) model for Example 3.

the best ODE(3) model in ten runs in Table 8 and illustrate its curves of fitting and prediction in Fig. 5.

We can see that both the fitting values and the predicted values are pretty good, and more importantly, the model is quite simple in structure. In [18], a TAR(2,2; 8,3) model was built for the dynamic data, but its structure is rather complicated.

3.3.4. Example 5: Chemical reaction temperature. The experimental data are taken from [17] giving the centigrade temperature of a chemical reaction process recorded every other minute. In this experiment, we take the recorded data of the first 220 minutes as historical data to build HODE models and predict the values of the last 6 minutes.

The statistical results of ten runs for Example 5 are shown in Table 9.

From the results we see that, of the four models, the ODE(1) model is the worst, as it has the largest AFE and APE. The ODE(3) model has a minimal AFE but its APE is large. For the ODE(4) model, its AFE and APE seem good, but it has only a 70% rate of success. While for the ODE(2) model, both its AFE and APE are good and it has a 100% rate of success. Based on the comparisons above, we

Table 7. The statistical results for Example 4 (10 runs)

Model	ODE(1)	ODE(2)	ODE(3)	ODE(4)
AFE	3.561151	1.243644	0.124922	0.507806
APE	0.852183	0.522207	0.264809	0.574626
AN _{nodes}	9.2	8.4	8.4	8.8
Mean Time (sec)	2280	3064	2325	1397
N _{succ}	10	10	10	10

Table 8. The best ODE(3) model for Example 4

Evolutionary solution	$\begin{cases} dy_1/dt = y_2 \\ dy_2/dt = y_3 \\ dy_3/dt = y_3/(y_2 - (-0.143929)) \end{cases}$			
Equivalent ODE	$x''' = x''/(x' + 0.143929)$			
FE	0.123900			
PE	0.264711			
Observed Value	3.000000	3.201397	3.424392	3.530968
Predicted Value	2.999852	3.219896	3.482772	3.788497

think that the ODE(2) model is superior to the other models in describing this time series whose order is identical to the AR(2) model built in [17]

$$x_t = 1.806681x_{t-1} - 0.80668163x_{t-2} + a_t \quad (\sigma_a^2 = 0.02774). \quad [35]$$

The results of the best ODE(2) model in ten runs are shown in Table 10 and its fitting and prediction curves are illustrated in Fig. 6.

It is surprising to see that the fitting values of the model can coincide with the observed data so well and that its predicted values are also good. Moreover, the model is a complex nonlinear differential equation which contains a cosine function in the expression. In fact, by running the HEMA, the computer can search many such complex models whose structures are usually unimaginable to human minds. This shows that computational intelligence can be competitive with human intelligence and even surpass it in some sense.

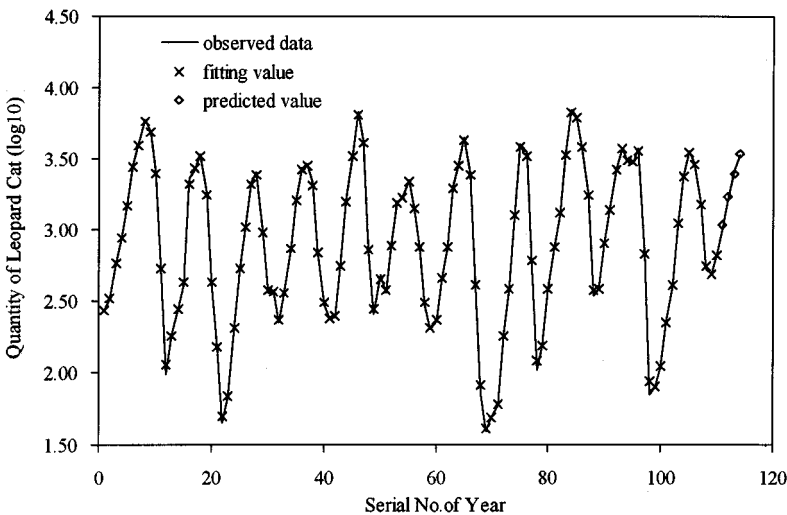


Figure 5. The fitting and prediction curves of the best ODE(3) model for Example 4.

Table 9. The statistical results for Example 5 (10 runs)

Model	ODE(1)	ODE(2)	ODE(3)	ODE(4)
AFE	3.272348	0.938791	0.083298	0.387948
APE	3.773765	1.679618	3.648096	2.104190
AN _{nodes}	7	7	8.4	6.6
Mean time (sec)	4303	5299	4432	3488
N _{succ}	10	10	10	7

4. Real-time evolutionary modeling of higher-order ODE

4.1. Real-time hybrid evolutionary modeling algorithm for higher-order ODE

When we apply the HEMA for HODE to some practical problems, we find this algorithm has one main drawback. The model is built once and for all, which can not reflect the real-time change of observed data. However in practical applications, it is usually required to adjust the structure of the model and its parameters with the renewing of the observed data so as to make real-time models and predictions. Hence we present a real-time HEMA in this section to approach this task.

Similarly we define m as the number of modeling samples, τ as the number of steps for prediction. The real-time EM of HODE is done as follows: i) take the m successive observed data points as modeling samples to build n th-order HODE models with the form of Eq. [22] using the HEMA in Section 3.2.1; ii) use the best evolved HODE model in step i) to predict the values of the system at the next τ time steps; iii) advance the samples for τ points as one step, and repeat step i) and step ii) until the number of advancing steps k specified beforehand is reached.

For brevity, we call the set of k n th-order ODE models built by running the real-time HEMA under the parameters m, τ as an ODE($m, \tau, k; n$) model family. The prediction with τ is called τ -step prediction.

4.2. Modeling experiments

4.2.1. Parameter settings. To examine the effectiveness of the real-time HEMA in making real-time predictions for time series, we apply it to two practical examples of time series and build the n th-order τ -step prediction ODE model families with $n = 1, 2, 3, 4; \tau = 1, 2, 3$ for each example respectively. Ten runs are con-

Table 10. The best ODE(2) model for Example 5

Evolutionary solution	$\begin{cases} dy_1/dt = y_2 \\ dy_2/dt = y_2/\cos(y_2*1.219242) \end{cases}$					
Equivalent ODE	$x'' = x'/\cos(1.219242x')$					
FE	0.943623					
PE	1.566485					
Observed value	20.2	19.7	19.3	19.1	19.0	18.8
Predicted value	20.250847	19.711464	19.179573	18.653486	18.121416	17.589474

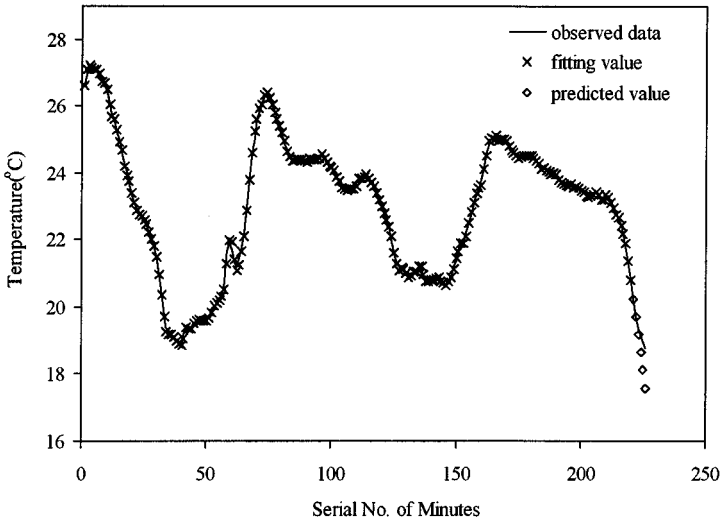


Figure 6. The fitting and prediction curves of the best ODE(2) model for Example 5.

ducted independently for each pair of (n, τ) . All the experiments are performed on a Pentium II (266 Mhz) using Visual C++ compilers. The parameter settings are the same as Table 4.

In addition, to compare the predicted results of different ODE model families, we define the prediction error (PE) as

$$PE = \sqrt{\sum_{i=m}^{m+\tau \times k-1} (\hat{x}_i - x_i)^2} \tag{36}$$

where x_i is the observed value, \hat{x}_i the predicted value. The average prediction error (APE) and minimal prediction error (MPE) are the mean value and the minimal value of PE in ten runs respectively. In one run, if there is at least one point i whose relative prediction error

$$\left| \frac{\hat{x}_i - x_i}{x_i} \right| \tag{37}$$

is greater than 1, we declare this run a failure; otherwise a success. We only take into account the successful runs when calculating the APE.

4.2.2. Example 6: Stock price. The experimental data are taken from [17] giving the daily stock price of IBM Company from May, 17, 1961 to November 2, 1962. We build the ODE($m, \tau, k; n$) model families for $m = 30; n = 1, 2, 3, 4; (\tau, k) = (1,60), (2,30), (3,20)$ by running the real-time HEMA.

Table 11 shows the statistical results of different ODE($m, \tau, k; n$) model families in ten runs for Example 6.

Table 11. The statistical results of different $ODE(m, \tau, k; n)$ model families for Example 6 (10 runs)

(τ, k)	Model family n	Failure	APE	MPE
(1,60)	1	0	43.764966	37.783276
	2	0	49.228340	43.944744
	3	1	76.678917	76.130070
	4	9	92.317932	92.317932
(2,30)	1	2	54.971361	51.628712
	2	2	94.856047	73.774323
	3	3	171.615132	169.149216
	4	10	—	—
(3,20)	1	0	66.412415	61.969063
	2	3	116.902307	94.866074
	3	4	400.622905	380.894219
	4	10	—	—

By comparison with the predicted results, one can see that whatever $\tau = 1, 2, 3$, the first-order $ODE(m, \tau, k; n)$ model family can always achieve the highest ratio of success and the minimal prediction error including APE and MPE. So we can infer that the optimal order of the ODE model for this example is 1. In addition, one can observe that as for the ODE models with the same order, their prediction errors increase obviously with the growth of τ as well as the ratios of failure. This is an inevitable result under the increasing prediction requirements. Additionally, considering the high ratios of failure (9/10, 10/10, 10/10 for $\tau = 1, 2, 3$ respectively) for the fourth-order ODE model family, we can conclude that the fourth-order ODE model is not suitable to describe the time series.

We depict the best-of-run results for $n = 1, \tau = 1, 2, 3$ in Fig. 7. We can see that for those best evolved ODE model families, the predicted values can coincide with

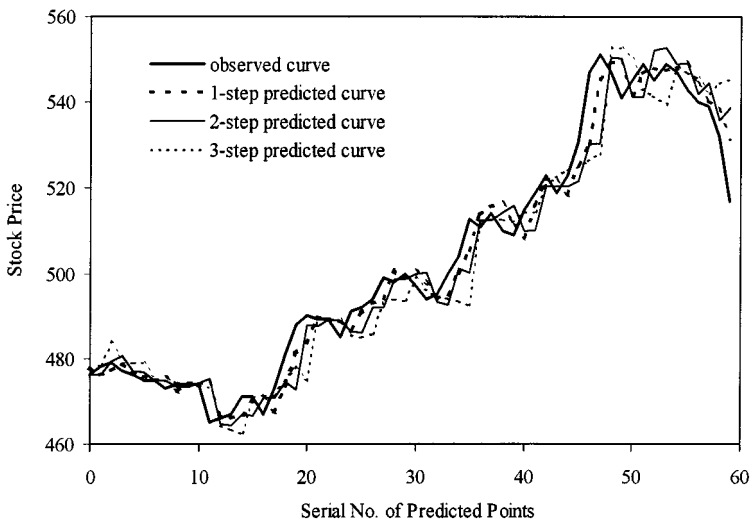


Figure 7. The best-of-run results when $n = 1, \tau = 1, 2, 3$ for Example 6.

the observed values very well. This shows that the real-time HEMA is effective in approaching the real-time modeling and predicting tasks of time series.

4.2.3. Example 7: Chemical reaction temperature. The experimental data are the same as Section 3.3.4. We build the $ODE(m, \tau, k; n)$ model families for $m = 20$; $n = 1, 2, 3, 4$; $(\tau, k) = (1,72), (2,36), (3,24)$ by running the real-time HEMA.

Table 12 shows the statistical results of different $ODE(m, \tau, k; n)$ model families in ten runs for Example 7.

From the results shown in Table 12, one can see i) for $\tau = 1$, the ODE model family with the order 3 $ODE(20,1,72;3)$ can get the smallest APE and MPE, and no failure occurs. Thus the third-order ODE model family is the optimal for 1-step prediction.; ii) for $\tau = 2$, it seems that the MPE is smaller when $n = 1, 2$ than when $n = 3$. But as a matter of fact, in view of their high ratios of failure (3/10, 7/10 for $n = 1,2$ respectively), we think that these two kinds of ODE model family are too unstable to make prediction. For the sake of reliability, we consider the third-order ODE model family $ODE(20,2,36;3)$ as the optimal model family for 2-step prediction.; iii) for $\tau = 3$, it is easily concluded that the second-order ODE model family $ODE(20,3,24;2)$ performs best in 3-step prediction in terms of its smallest APE and MPE as well as 100% ratio of success.

The results discussed in i)–iii) as above show that even for the same time series, the optimal order of ODE model family can vary with the predicted requirements which is reflected by the number of steps for prediction, τ . But as a common result of those two examples, the ODE models with too high an order (order 4, for example) are usually not applicable in practical problems.

The best-of-run results for $(\tau, n) = (1,3), (2,3), (3,2)$ are illustrated in Fig. 8. We can see that although a few predicted points deviate the original curve a little, most of them can surround it closely.

Table 12. The statistical results of different $ODE(m, \tau, k; n)$ model families for Example 7 (10 runs)

Model family		Failure	APE	MPE
(τ, k)	n			
(1,72)	1	0	3.958492	2.514435
	2	1	4.397613	2.478638
	3	0	2.236532	2.222598
	4	1	11.108332	6.294726
(2,36)	1	3	4.492476	3.933218
	2	7	7.779892	3.267442
	3	0	6.234893	4.238805
	4	10	—	—
(3,24)	1	1	9.289669	5.410786
	2	0	5.143572	3.542628
	3	0	10.532960	9.815809
	4	10	—	—

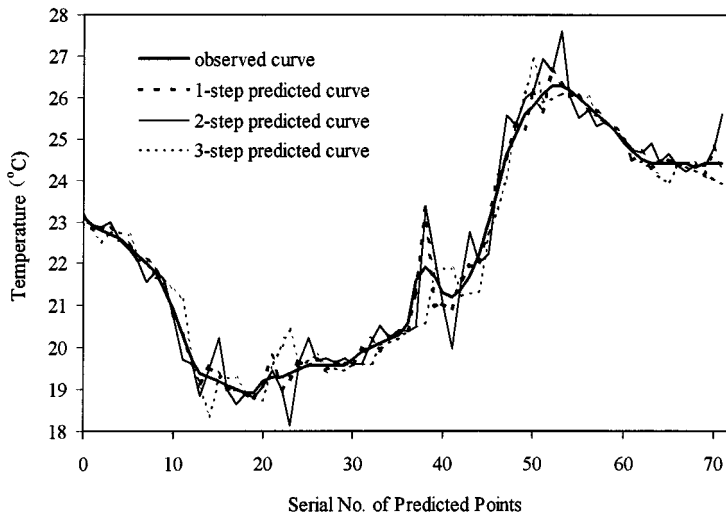


Figure 8. The best-of-run results when $(\tau, n) = (1, 3), (2, 3), (3, 2)$ for Example 7.

5. Conclusions and future research

Compared with most available modeling methods, EM has the following advantages:

- 1) It requires little system information and mainly relies on computational intelligence to discover models automatically.
- 2) The optimization of the structure of a model and the optimization of its parameters can be performed concurrently and automatically by computers. The modelers need not predetermine the model structure and the number of the parameters.
- 3) The ODE model discovered by the computer from the dynamic data of time series can not only fit and predict the annual data quite well, but also their structures may be various and beyond human understanding.

As our research in EM of ODE is currently at a preliminary stage, some problems exist when using the HEMA to practical applications. They mainly include:

- 1) The structure of a model depends largely on some control parameters predetermined by a human, such as the function set, the terminal set, the maximum tree depth and so on. If those parameters are chosen inappropriately, it is very likely that there is a large discrepancy between the model built by running the HEMA and the actual model of the system.
- 2) In our experiments, the criterion of choosing a better model is how well the model can fit the historical data. But as those observed data can only reflect the state of a system during a period, it is very likely that the best evolved model can fit the historical data quite well but may make bad predictions. Additionally, the parsimony of a model is not taken into account in our fitness function.

More complete criteria for model evaluation need to be introduced in later experiments.

- 3) As multi-step integration of SODE has to be done during the fitness calculation, this makes the time cost of EM proportional to the amount of input data. This greatly limits the size and the complexity of the problems which can be solved by the HEMA. Some techniques need to be explored to improve the efficiency of the HEMA, such as the parallel implementation of the algorithm, more effective data preprocessing and so on.
- 4) The GA used to optimize the parameters of a model is another important factor which gives rise to great computational cost. How to improve its performance is an important topic for future research.
- 5) The evolutionary models are various and their structure are usually rather complicated. Explaining these models and their parameters is a difficult task.

All these issues will be addressed in our future research.

Acknowledgments

This work was supported by State Key Laboratory of Parallel and Distributed Processing, National Natural Science Foundation of China (No. 69635030) and National 863 High Technology Project of China. The authors would like to thank the anonymous referees for their helpful comments on the paper and to especially thank Dr. Lee Spector for his efforts in improving the paper's English.

Appendix: A list of abbreviations

EM	evolutionary modeling
ODE	ordinary differential equation
SODE	system of ordinary differential equations
HODE	higher-order ordinary differential equation
HEMA	hybrid evolutionary modeling algorithm
GA	genetic algorithm
GP	genetic programming
EA	evolutionary algorithm
AR models	autoregressive models
MA models	moving average models
ARMA models	autoregressive moving average models
ARIMA models	autoregressive integrated moving average models
TAR models	threshold autoregressive models
TARMA models	threshold autoregressive moving average models

References

1. J. X. He, *System Modelling and Mathematical Models*, Xiamen: Fujian Science & Technology Press, 1995, pp. 106–173 (in Chinese).

2. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley: Reading, MA, 1989.
3. T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Trans. Evol. Comput.* vol. 1, no. 1, pp. 5–16, 1997.
4. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press: Cambridge, MA, 1992.
5. J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press: Cambridge, MA, 1994.
6. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*, Morgan Kaufmann: San Francisco, 1997.
7. J. H. Holland, *Adaptation in Natural and Artificial System*, University of Michigan Press: Ann Arbor, MI, 1975.
8. M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press: Cambridge, MA, 1996.
9. T. P. Meyer and N. H. Packard, "Local forecasting of high-dimensional chaotic dynamics," in M. Casdagli and S. Eubank (eds.) *Nonlinear Modeling and Forecasting*. Addison-Wesley: Reading, MA: 1992.
10. D. Searson, M. Willis, and G. Montague, "Chemical process controller design using genetic programming," in E. David, Hitoshi Iba, and R. L. Riolo (eds.), *Genetic Programming 1998: Proc. Third Annual Conf.*, July 22–25, 1998, University of Wisconsin, Madison, Wisconsin, Morgan Kaufmann: San Francisco, 1998, pp. 359–364.
11. R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, John Wiley & Sons: New York, 1998.
12. J. C. Nash and M. Walker-Smith, *Nonlinear Parameter Estimation*, Marcel Dekker: New York, Basel, 1987.
13. R. Hinterding, Z. Michalewicz, and A. Eiben, "Adaptation in evolutionary computation: a survey," in T. Bäck, Z. Michalewicz, and X. Yao (eds.), *Proc. 4th Int. Conf. Evol. Comput.* IEEE Press: Piscataway, NJ, pp. 65–69, 1997.
14. H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Comput.* vol. 17, pp. 619–632, 1991.
15. H. Mühlenbein and D. Schlierkamp-rose, "Predictive models for the breeder genetic algorithm," *Evol. Comput.* vol. 1, no. 1, pp. 25–49, 1993.
16. Q. Y. Jiang, *Mathematical Models*, ed. II, Beijing: Education Press, 1993, pp. 110–221 (in Chinese).
17. R. C. Gan, *The Statistical Analysis of Dynamic Data*, Beijing: Beijing University of Science and Technology Press, 1991 (in Chinese).
18. J. T. Xiang, J. G. Du, and J. E. Shi, *Dynamic Data Processing: Time Series Analysis*, Beijing: Meteorology Press, 1988 (in Chinese).
19. T. Ozaki, "On the order determination of ARIMA models," *Applicat. Stat.* vol. 26, no. 3, pp. 290–301, 1977.
20. C. W. Granger and A. P. Andersen, *An Introduction to Bilinear Time Series Models*, Vandenhoeck and Ruprecht: Göttingen, 1978.
21. H. Tong. "On a threshold model," *Pattern Recog. and Signal Process.* NATO ASI Ser. E: Appli. Sci. vol. 29, 1978.
22. W. Y. Wang, J. G. Du, and J. T. Xiang, "Threshold autoregressive moving average models: TARMA," *Comput. Math.* no. 4, 1984 (in Chinese).