PHYSICAL REVIEW E VOLUME 55, NUMBER 3 MARCH 1997

Forecasting chaotic time series with genetic algorithms

George G. Szpiro*
The Israeli Center for Academic Studies, Kiriat Ono 55000, Israel
(Received 25 January 1996; revised manuscript received 16 October 1996)

This paper proposes the use of genetic algorithms—search procedures, modeled on the Darwinian theories of natural selection and survival of the fittest—to find equations that describe the behavior of a time series. The method permits global forecasts of such series. Very little data are sufficient to utilize the method and, as a byproduct, these algorithms sometimes indicate the functional form of the dynamic that underlies the data. The algorithms are tested with clean as well as with noisy chaotic data, and with the sunspot series. [S1063-651X(97)12403-5]

PACS number(s): 02.50.−r, 02.60.Gf, 96.60.Qc

I. INTRODUCTION

Whether the data of a time series, $\{x_1,x_2,x_3,\ldots,x_T\}$ were created by a deterministic or by a random process is often decided indirectly, by estimating the fractal dimension of the time series in successively higher embedding dimensions using, for example, the Grassberger-Procaccia [1,2] algorithm. If the dimension estimates converge as the embedding dimension increases it may be concluded that a deterministic dynamic underlies the time series. (Significant drawbacks of this method are, however, that massive amounts of data are needed and that it depends on the visual inspection of graphs.) Another tool for the assessment of the deterministic origin of a time series consists, for example, in the application of noise-reduction methods [3], which attempt to ascertain the deterministic origin of the data by measuring the level of noise in the series [4]. Once it is established with a sufficient level of confidence that the series has been generated by a deterministic process, prediction of the time series' future behavior, based on its past values, can, in principle, be attempted. (Actually, successful prediction may be the most stringent proof of the data's deterministic origin.) The scalar values of a series can be forecast either by selecting a model based on the observed data [5], or by expressing them as a function of the coordinates of points that are spatially close in embedding space, or by relating them to the time series' values in the immediate past:

$$x_t=f(x_{t-1},x_{t-2},\ldots,x_{t-L}),\quad L+1\leq t\leq T,\qquad(1)$$

where L, the number of previous values that may appear in the equation, is a parameter that needs to be specified at the outset [6].

This paper deals with the latter method: we attempt to find a formula, such as Eq. (1), that links present and future values of the series to their previous entries. Ideally, the formula is identical to the true data-generating process; at the least, it should mimic its behavior such that forecasts can be made. The technique is global, in the sense that a single formula is sought that allows forecasts of future entries in *any* series generated by the process—starting at *any* point in time. Local approximation schemes, on the other hand, deal only with data that lie in a close neighborhood in embedding space, and require separate computations for forecasts in different regions. Obviously, a conventional search for the actual equation of the data generating process is hopeless, since the dynamics underlying most data, especially chaotic series, are far too complicated. Classical regression analysis is also all but useless. During the past decade various nonlinear techniques have been developed to accomplish the task of forecasting. Chaos theory suggested the method of ''nearest neighbors'' [7,8] and of radial basis function predictors [9,10], artificial intelligence led to the use of neural nets [11,12,13,14], and recently wavelets [15,16] have been employed to predict chaotic time series. The nearest neighbor technique (which, again, requires massive amounts of data) embeds the time series in a space of sufficiently high dimension, and then seeks vectors in the historical series that are similar to the one that is to be predicted. Neural nets also use a historical series, and the backpropagation of errors, to fine tune a set of parameters, and thereby build a forecasting function that links past values of the time series with future values. Predictions based on radial basis functions use global interpolation techniques and have proven useful in practice when only sparse data was available. Finally, wavelet analysis—like the Fourier transform—decomposes a time series into its basic components, thus allowing insight into its fine structure, and then uses a weighted sum of these wavelets for forecasting purposes.

This paper proposes genetic algorithms, a technique borrowed from evolutionary biology, to accomplish the task. It usually requires only a few dozen data points to give dependable results, and frequently offers the additional benefit of indicating the underlying process' functional form. As we use them here, genetic algorithms actually attempt to search for a formula that determines the dynamic, i.e., for an equation *in symbolic form,* which fits the data globally. But they do this in a far more sophisticated way than by trying to exhaustively enumerate all possible equations. Another approach to forecasting complex dynamics with genetic algorithms has been reported [17], and efforts have been made to combine the nearest neighbor technique with genetic algorithms [18], wavelets with neural nets [19], or to merge the three methods (nearest neighbors, neural networks, and ge-

*Permanent address: P.O. Box 6278, Jerusalem 91060, Israel. Electronic address: george@netvision.net.il netic algorithms) for prediction purposes [20]. There have been some attempts to utilize such methods for investment purposes [21,22].

Genetic algorithms were pioneered by Holland [23], and have been used in such diverse areas as engineering [24], chemistry [25], optimization [26], acoustics [27], pattern recognition [28], and economics [29], to mention just a few. A formal analysis of genetic algorithms can be made, using the tools of mechanical statistics [30]. The closely related method of evolutionary programming has been employed to elicit some basic physical laws from real-life data, for example, Kepler's third law and Ohm's law [31].

II. THE GENETIC ALGORITHM

The basic idea in this paper is to break up mathematical equations into their building blocks, and to use these blocks in the same way that nature uses genetic material and chromosomes to evolve ever fitter individuals. The Darwinian processes of natural selection and survival of the fittest tend to make suitable blocks of the equations combine with other useful blocks, while the useless parts eventually disappear.

Let us assume a scalar time series $\{x_1,x_2,x_3,\ldots,x_T\}$. We want to determine the dependence of the values x_t $(L+1\leqslant t\leqslant T)$ on their previous values $x_{t-\lambda}$ $(1\leqslant\lambda\leqslant L$, where L is the maximum lag that we allow). The algorithm starts out with a population of N initial ''equation strings.'' These are sequences of randomly chosen symbols that conform to a simple grammar of mathematical equations: two arguments are combined by an arithmetic operator, and the resulting expression is enclosed in parentheses. The arguments are either real numbers, or values from the time series, or are themselves self-contained expressions enclosed in a pair of parentheses. The resulting expressions, super-expressions, and compounded expressions form the building blocks of the equation strings on which the Darwinian processes operate. (In the remainder of the paper we will sometimes call such equation-strings ''agents.'') Genetic algorithms consist of a few, quite simple routines, which are described, step by step, in what follows:

(a) Initialization. The agents of the populations are initially endowed with simple equation strings of the form

$$S_j=((A\otimes B)\otimes(C\otimes D)),\quad 1\leqslant j\leqslant N \tag{2}$$

where A, B, C, and D are either real numbers (i.e., parameters of the equations that we are looking for), or variables $x_{t-\lambda}$ $(1\leqslant\lambda\leqslant L)$. In the following routines these variables will be identified with values in the time-series: for each t $(L+1\leqslant t\leqslant T)$, $x_{t-\lambda}$ is a previous, or ''lagged'', value of x_t. $\otimes$ stands for one of the four arithmetic operators, addition, subtraction, multiplication, or division. At this stage the operators and arguments are assigned at random: numerical values are chosen from a finite set of real numbers, uniformly distributed in $[-Z,+Z]$, and the integer λ is uniformly distributed in $1\leqslant\lambda\leqslant L$. (One of the algorithm's parameters that must be specified at the outset is the probability for the argument to be a number, or a variable representing a previous value of the time series.)

In order to avoid division by zero, or by very small numbers, we use a ''protected'' division which is defined as the division by Max (0.001, $|B|$), while preserving the sign, i.e.,

$$A\div B\equiv\frac{A}{\mathrm{Max}(0.001,|B|)\operatorname{sgn}(B)}. \tag{3}$$

Other mathematical constructs are conceivable, for instance, the log and exp functions (with one argument), ''if-then-else'' statements (with three arguments), or Boolean operators. In this paper we limit ourselves to the four arithmetic operators.

(b) Computing the fitness. We now define a criterion that measures how well the equation strings perform on the training set, $\{x_{L+1},x_{L+2},\ldots,x_T\}$. For each agent j the equation string is used to compute estimates of all x_t in the training set, as a functions of the previous values of the time series,

$$x_t^j=f_j(x_{t-1},x_{t-2},\ldots,x_{t-L}),\quad L+1\leqslant t\leqslant T,\quad 1\leqslant j\leqslant N \tag{4}$$

[x_t^j denotes agent j's estimate, and $f_j(\,)$ represents the equation string of agent j]. The squared errors are summed for each agent,

$$\Delta_j^2=\sum_{t=L+1}^{T}(x_t^j-x_t)^2. \tag{5}$$

The lower the sum of squared errors, the fitter is the agent. The percentage of the training set's total variance that is explained by the equation string, denoted by the symbol R_j^2,

$$R_j^2=1-\frac{\Delta_j^2}{\sum_{t=L+1}^{T}(x_t-\overline{x})^2} \tag{6}$$

lends itself as an initial fitness measure. ($\overline{x}$ represents the mean of all x_t in the training set.) However, in order to discourage the genetic algorithm from overfitting, by creating ever longer strings through the combination of more and more parts of equations, we perform a modification to the value of R^2 in the following manner (we drop the subscript j henceforth),

$$r^2=1-(1-R^2)\,\frac{T-L-1}{T-L-k}, \tag{7}$$

where $T-L$ is the length of the training set, and k is the number of variables $x_{t-\lambda}$ that appear in the string. (r^2 will be called the *modified* R^2.) This new fitness measure, which can be negative if R_j^2 is close to zero, confers an advantage on short strings. r^2 is inspired by, but not the same as, the adjusted R^2, as is known from multiple linear regression: since multiple occurrences of the same variable are counted separately, compositions like $x_{t-2}x_{t-3}/x_{t-4}$ increase k by 3, and x_{t-5}/x_{t-5} or $x_{t-4}-x_{t-4}$ increase k by 2, even though in the latter cases there are no new variables. We use r^2 as the fitness measure in the algorithm, but report the more familiar R^2 value, which we will call the ''explained variance.''

(c) Ranking the agents. The agents are ranked in descending order of their fitness.

(d) Choice of mates. In this routine the members of the population are organized into pairs. The fittest agent is the first to choose a mate. It makes its choice from among the remaining agents, the probability of any one of them becom-

ing a partner being proportional to its fitness. (We do not allow agents to mate with copies of themselves.) Then the next fittest chooses, etc., until $N/2$ agents have formed pairs. The remaining agents disappear. Thus, a total of $N/4$ pairs are formed.

(e) Reproduction and crossover. This routine is the heart of the algorithm. Each of the $N/4$ pairs has four offspring. (Hence, the population size stays constant.) We choose the following method of passing ‘‘genetic information’’ to the next generation. The first two offspring are identical to their parents. The equation strings of the two other offspring are formed as recombinations of their parents' equation strings: randomly chosen, self-contained parts of the parents' equation strings (either a randomly chosen datum or real number, or a randomly chosen opening parenthesis, and all the contents of the equation string until the corresponding closing parenthesis) are interchanged. For example, let us assume the parents' equation strings are as follows:

$$\begin{aligned} &\text{Parent 1:} \quad (A*B)/C, \\ &\text{Parent 2:} \quad (D-(E/F)), \end{aligned} \tag{8}$$

where A, B, C, D, E, and F are, again, either equal to $x_{t-\lambda}$ $(1 \leq \lambda \leq L)$, or to real numbers. The equation strings of offspring 1 and offspring 2 are identical to their parents. Then—by crossover of the building blocks B and (E/F)—the two other offspring's equation strings could be of the form

$$\begin{aligned} &\text{Offspring 3:} \quad (A*(E/F))/C, \\ &\text{Offspring 4:} \quad (D-B). \end{aligned} \tag{9}$$

(The operators are left untouched in this routine.) To summarize, with the first two offspring, fitness is preserved, while provisions for improvement are made with the second pair of offspring.

(f) Mutation. A small percentage of the equation strings' most basic elements, single operators and arguments, are mutated at random. An element is randomly selected and, depending on whether it us a number, a variable, or an operator, a new number, variable, or operator is chosen at random as in routine (a) above. The top ranked equation strings are exempted from mutation, however, so that their information is not lost inadvertently. Mutation ensures that the agents do not converge prematurely to a stable, but relatively low, level of fitness [32]. The number of mutations in each generation, and the percentage of strings that are exempted, are parameters that have to be specified at the outset.

Parts (b)–(f) of the algorithm are rerun for a certain number of generations, or until some stopping criterion is satisfied, for example, when fitness no longer increases. Then editing operations are performed on the top-ranked equation string, which cancel, concatenate, and simplify the variables and numbers that appear in the equation string, and break it down into a concise formula.

For our numerical experiments, the algorithm is implemented with the following parameter values: a population size of 400, a maximal string length, including parentheses, of 600 (if equation strings become longer than that, the string is reinitialized, when they are shorter, the remaining slots are filled with null symbols), and training sets with length 100. Whenever an operator needs to be chosen at random—at initialization or at mutation—the probabilities of getting either a number or a lagged datum, are 25% and 75%, respectively. The maximal lag, L, is 10, and numbers are uniformly distributed in $[-Z,+Z]$, with $Z=10$ and a precision of one decimal. In each generation 240 mutations occur. The top-ranked 10% of the population are exempted from mutation. (Multiple mutations can occur in the same equation string. On average, two-thirds of the 360 lower-ranked agents undergo one mutation in each generation.) In general, we let the algorithm continue for 200 generations, but for high-dimensional and for noisy series up to 1200 generations were run.

Other evolutionary schemes are, of course, also possible. For example, the procedure for the choice of a mate, or the method of passing ‘‘genetic information’’ from one generation to the next may be varied, and obviously the numerical values of the parameters can be changed as well. Or, for example, one could envisage, say, forming $N/6$ pairs in each generation, with 6 offspring each. It is interesting to note, though, that the algorithm is quite robust to such changes. The most efficient algorithm is found through experimentation.

III. REFINEMENTS AND IMPROVEMENTS

The genetic algorithm proposed in this paper searches for equations that mimic the dynamic that underlies a time series. However, the equation string that is the top scorer after many generations is not necessarily a well-adapted agent, even if the explained variance is good: over-fitting may produce misleadingly high R^2. To ascertain that the genetic algorithm bred an equation string that is fit—in a meaningful sense of the word—the top-ranked equation must always be applied to a test set and its predictive power must be rechecked with these new data.

It should be noted that the genetic algorithm does not necessarily find the simplest version of an expression. For instance, the number 1.0 could be stated as x_{t-5}/x_{t-5}, zero as $x_{t-4}-x_{t-4}$, $2x_{t-3}$ as $x_{t-3}(x_{t-5}+x_{t-5})/x_{t-5}$, etc. Also, equations need not be expressed in terms of the most recent lags, they could be recursively defined in terms of previous lags.

After the genetic algorithm has evolved an equation string, the result can sometimes be enhanced by applying the method to the series of residuals. The latter is defined as the series of values ε_t that remain after the results of the fittest equation string have been deducted from the original series:

$$\varepsilon_t = x_t - f^*(x_{t-1}, x_{t-2}, \ldots, x_{t-L}), \quad L+1 \leq t \leq T, \tag{10}$$

where $f^*(\,)$ is the top-scoring equation string at the end of the algorithm's run. If some parts of the data-generating formula dominate the other parts, an improvement of the result may be brought about by running the algorithm with the new time series $\{\varepsilon_{L+1}, \varepsilon_{L+2}, \ldots, \varepsilon_T\}$ [33].

In the numerical experiments described below, we first breed formulas that give forecasts of x_t as functions of the immediately preceding $x_{t-\lambda}$ ($L+1 \leqslant t \leqslant T$ and $1 \leqslant \lambda \leqslant L$). To express values in the time series that are further ahead in the future, the formulas could be iterated,

$$x_{t+n} = f^* \circ f^* \circ \cdots \circ f^*(x_{t-1}, x_{t-2}, \ldots, x_{t-L}), \quad n \geqslant 1. \tag{11}$$

But one cannot expect very good results: in the examples that follow in the next section we use chaotic processes, whose sensitive dependence on initial conditions is well known. Hence, even a slight error in the forecasts of x_t will lead to rapidly increasing errors in the forecasts of x_{t+n} if the latter are computed according to Eq. (11). However, by simply modifying one parameter, genetic algorithms can be directed to specifically breed n-period forecasts: if the integer λ is required to be uniformly distributed in $[n,K]$ instead of $[1,L]$ in the initialization and mutation stages of the algorithm, the strings that give x_t^j [Eq. (4)] will be expressed specifically in terms of $x_{t-\lambda}$, with $n \leqslant \lambda \leqslant K$. Thus the new equation string allows the forecast of x_t in period t-n. The fitness for n-period forecasts is generally lower than for 1-period forecasts, but it is usually better than when Eq. (11) is applied recursively n times.

Finally let it be noted that the genetic algorithm described here can be modified to handle vector time series, $\{\langle x_1, y_1, \ldots, z_1\rangle, \langle x_2, y_2, \ldots, z_2\rangle, \ldots, \langle x_T, y_T, \ldots, z_T\rangle\}$. By letting the algorithm choose the lagged variables in the initialization and the mutation routines not only from among the $x_{t-\lambda}$, but also from $y_{t-\lambda}, \ldots, z_{t-\lambda}$, strings will be bred that express x_t as a function of $x_{t-\lambda}, y_{t-\lambda}, \ldots, z_{t-\lambda}$ ($1 \leqslant \lambda \leqslant L$).

IV. NUMERICAL EVIDENCE

We apply the genetic algorithm to clean and to noisy data, and to a series of sunspot data. In the numerical experiments we use chaotic time series whose future values are notoriously difficult to predict from historical data. In Sec. V we compare our results to those obtained in other studies.

(a) Rössler attractor (without noise). We simulate the Rössler attractor [34] as

$$\begin{aligned} x(t+\delta) &= x(t) - [y(t) + z(t)]\delta, \\ y(t+\delta) &= y(t) + [x(t) + ay(t)]\delta, \\ z(t+\delta) &= z(t) + [b + x(t)z(t) - cz(t)]\delta, \end{aligned} \tag{12}$$

with $a=0.2$, $b=0.2$, $c=5.7$, initial values $x_0=-1$, $y_0=0$, $z_0=0$, and $\delta=0.02$. The results of this equation system for integer values of t (i.e., for every 50th step) form the time series x_t, y_t, and z_t. We first apply the genetic algorithm to the time series of the x variable. After 200 generations of a typical run, the top-ranked equation string was [35]

$$\begin{aligned} A_0 = (((((((((0.4+(\mathrm{A6}+\mathrm{A8}))*(((((\mathrm{A3} \\ -\mathrm{A2})*(\mathrm{A3}*4.5))/4.4)+\mathrm{A3})/4.5))/8.4)-\mathrm{A3}) \\ +\mathrm{A1})*(\mathrm{A5}/4.5))/4.4)-\mathrm{A3}) \end{aligned}$$

with an R^2 of 0.878. In edited form, this equation is

$$\begin{aligned} x_t = {} & -x_{t-3} + 0.0505051 x_{t-5}(x_{t-1} - x_{t-3}) \\ & + 0.00136647 x_{t-3} x_{t-5}(0.977775 - x_{t-2} + x_{t-3}) \\ & \times (0.4 + x_{t-6} + x_{t-8}). \end{aligned} \tag{13}$$

In order to test the string's predictive power, a new time series was created with different data. Figure 1 shows how the equation performs on a set with new data. The R^2 of these forecasts is 0.912.

For the time series of the z variable we got the following equation after 200 generations, for example,

$$z_t = \frac{1.2 z_{t-1}^2}{(104.304 + z_{t-1})(0.4 z_{t-1}^3 z_{t-2}^2 + z_{t-2} z_{t-3} z_{t-10})[1.8 + 49.104 z_{t-3} + 27.28 z_{t-3}(z_{t-1} + z_{t-4})]}, \tag{14}$$

with explained variance of 91.6%. This particular example shows, however, how a genetic algorithm can get somewhat thrown off the mark. A test with a new data set results in an R^2 of only 0.568 for fifty data points. The reason is the following: since more than 90% of the values of the z series are contained in the interval [0, 1.0], while less than 10% of the data lie in [1.0, 15.0], the genetic algorithm is fooled. By getting the outliers approximately right, and setting all other values close to zero, it performs quite well in terms of R^2. (The results are depicted in Fig. 2 and, in more detail, in Fig. 3.) Hence, depending on what the aim of the exercise is, the fitness measure may need to be adjusted. If the goal is to predict the occurrence of outliers, then it is correct to use the genetic algorithm in the above manner. If, on the other hand, the aim is to get as many data points as close to their true values as possible, then it would be worthwhile to specify a cap on the maximum value for the error for each datum. This reduces the weight of the outliers, putting the emphasis on the mainstream data.

(b) Mackey-Glass delay differential equation (without noise). The dimension of the time series produced by the Mackey-Glass process [36]

$$\frac{dx_t}{dt} = \frac{a x_{t-\tau}}{1 + x_{t-\tau}^c} - b x_t \tag{15}$$

(with $a=0.2$, $b=0.1$, and $c=10$) depends on the delay parameter τ. We will use $\tau=30$, which corresponds to a dimension of about 3.5, and $\tau=100$, which corresponds to a dimension of about 7.5. (The series was created by subdividing every time unit into ten subunits, letting transients die out,

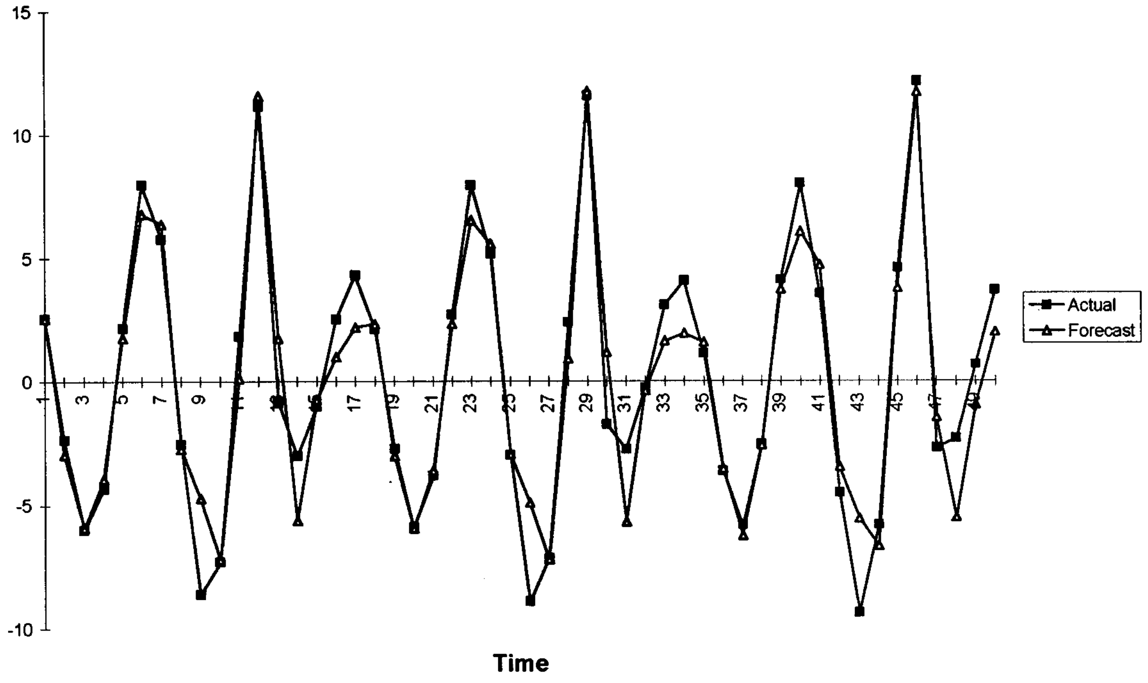


FIG. 1. Rössler attractor (x values).

and then observing every tenth subunit.) For $\tau=30$ we obtain—after not more than a couple of generations—equation strings of the form

$$A_0=((\mathrm{A6}*\mathrm{A1})/\mathrm{A6})$$

or

$$A_0=((\mathrm{A1}*\mathrm{A3})/\mathrm{A4}),$$

with R^2 values of 0.990 and 0.997, respectively. Since the series is very highly autocorrelated, excellent one-period forecasts are not surprising [37], and it will be more instructive to inspect longer-term predictions. We breed 40-period forecasts, as explained at the end of Sec. III, by setting the first lag equal to 40. One such run gave the equation

$$A_0=(\mathrm{A40}-((\mathrm{A40}*\mathrm{A40})-0.8)),$$

with 0.631 explained variance, another resulted in

$$A_0=(0.8-((\mathrm{A40}*\mathrm{A44})-(\mathrm{A46}+(\mathrm{A45}-\mathrm{A49})))),$$

with R^2 of 0.642. Using these equations and new data sets, 40-period forecasts typically produced R^2 values of between 0.400 and 0.650 in both cases. (Some test-series did have much lower R^2 values, however.)

Let us now turn to a series of higher dimension by setting the parameter τ in Eq. (12) equal to 100. One-period predictions are again trivial and will not be listed here. We investigate 10-period forecasts. With high-dimensional data the algorithm must be rerun with longer training periods, and we now use training periods of length 500 to breed equation strings and produce forecasts. Because of the high autocorrelation of the Mackey-Glass time series, one could justify ''naive'' forecasts of the form $x_{t+10}=x_t$ and, in effect, the equation string A0=A10 usually results in explained variances of up to 0.500. However, genetic algorithms perform significantly better. Three examples are

$$A_0=(\mathrm{A10}+((\mathrm{A17}-\mathrm{A15})-(\mathrm{A13}-\mathrm{A10}))),$$
$$A_0=(\mathrm{A17}+((\mathrm{A12}-6.1)*(\mathrm{A12}-\mathrm{A10}))),$$
$$A_0=((\mathrm{A15}*(\mathrm{A19}+(\mathrm{A10}-(6.4*(\mathrm{A13}-\mathrm{A10}/((\mathrm{A10}*\mathrm{A19})+\mathrm{A10}),$$

with R^2 of 0.705, 0.770, and 0.776, respectively. Applying these equation strings to different realizations of the Mackey-Glass time series with $\tau=100$, we typically get R^2 values of between 0.450 and 0.800.

(c) Belousov-Zhabotinskii simulation (with noise). Now let us use a simulation of the behavior of the Belousov-Zhabotinskii reaction in chemistry, with noise added [38],

$$x_t=\begin{cases}-(0.125-x_{t-1})^{1/3}+0.50607357\exp(-x_{t-1})+\varepsilon & \text{for } x_{t-1}\leq 0.125\\ (x_{t-1}-0.125)^{1/3}+0.50607357\exp(-x_{t-1})+\varepsilon & \text{for } 0.125<x_{t-1}\leq 0.3\\ 0.121205692\left[10x_{t-1}\exp\left(-\frac{10}{3}x_{t-1}\right)\right]^{19}+\varepsilon & \text{for } x_{t-1}>0.3.\end{cases} \tag{16}$$

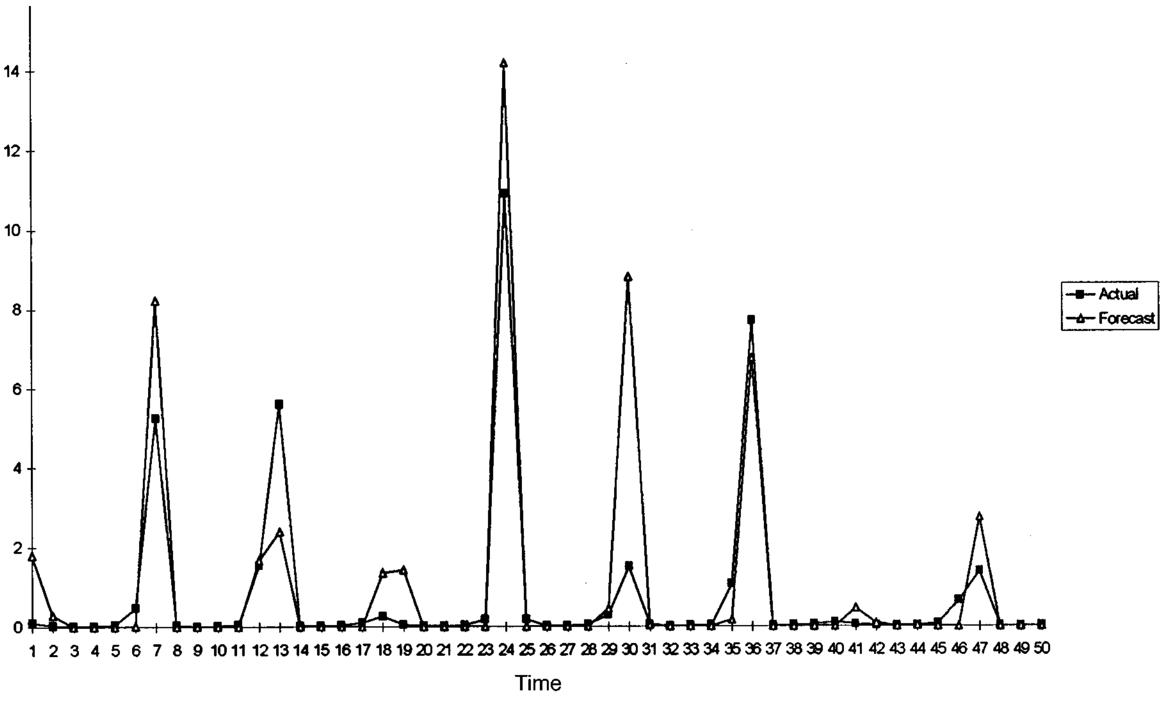


FIG. 2. Rössler attractor (z values).

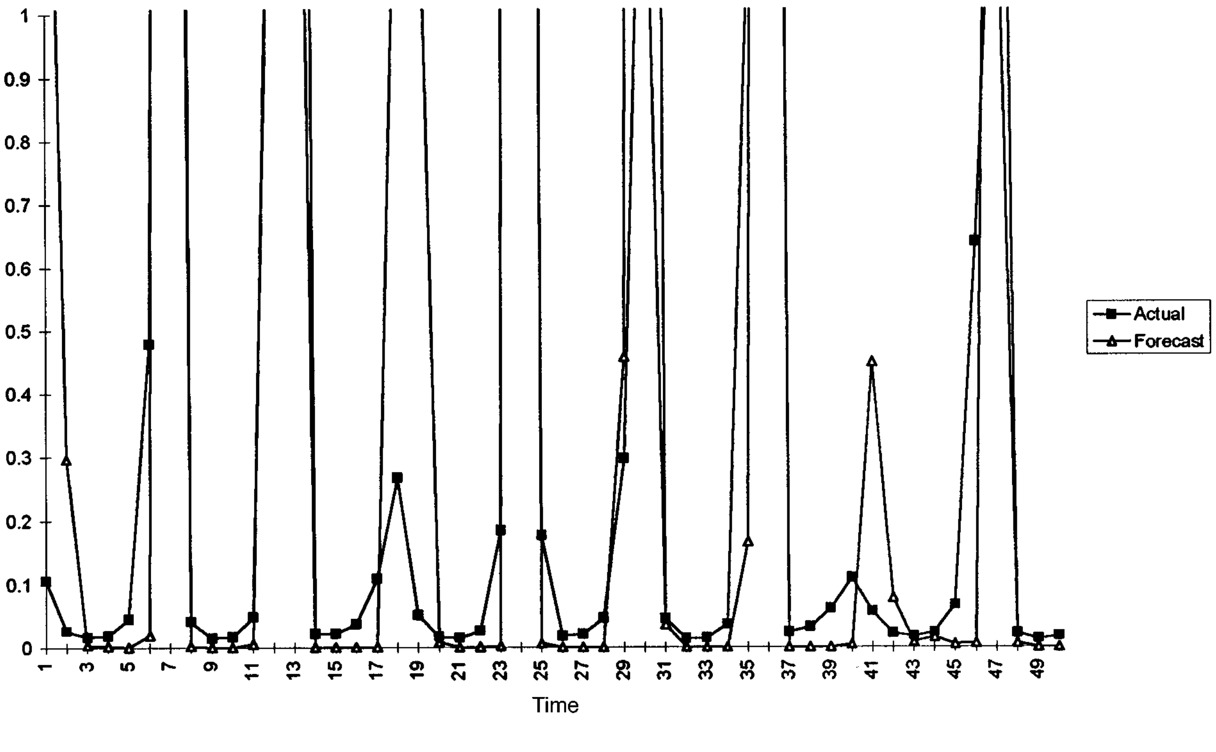


FIG. 3. Detail of Fig. 2.

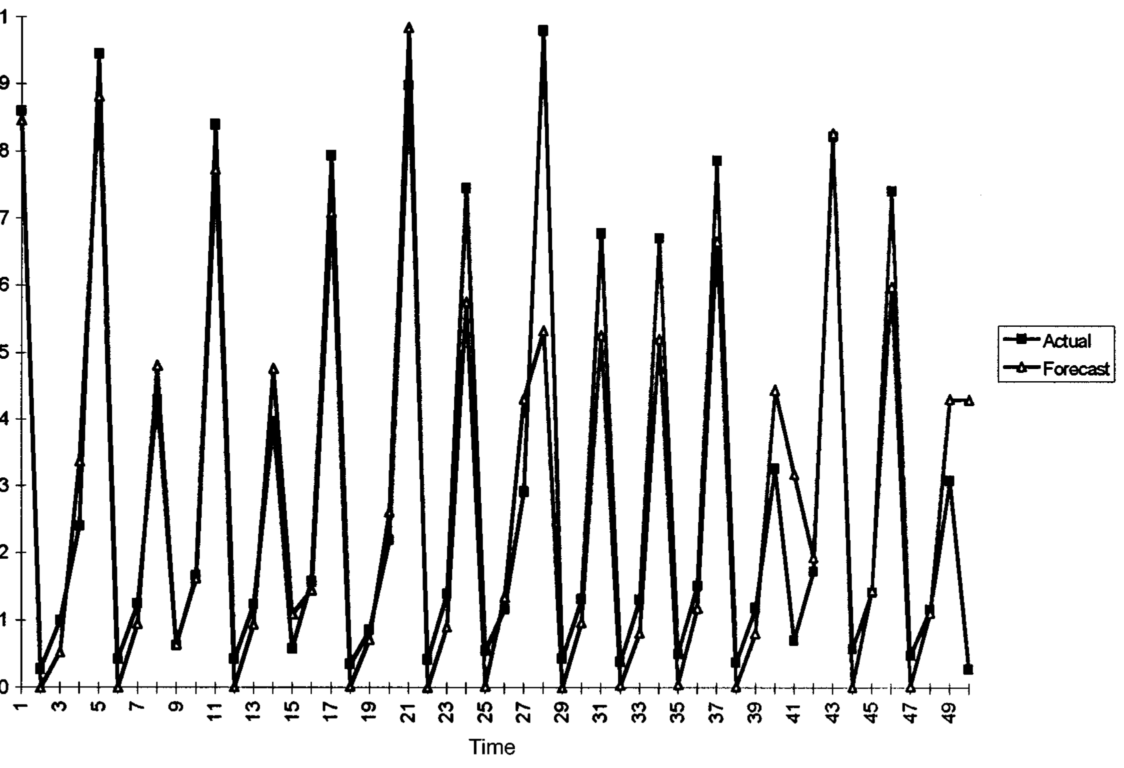


FIG. 4. *B-Z* series with noise.

The noise term ε is uniformly distributed in [0, 0.05]. As before, our training period consists of 100 data points. After 200 generations of one typical run the edited version of the top-ranked equation string had the form

$$x_t = \frac{3.1x_{t-1} + 46.98x_{t-1}^2}{2.3 + 87.74x_{t-1}^4 x_{t-2}(3.1 + x_{t-2} + x_{t-4} + x_{t-5} + 2847.71630(x_{t-1}^4 + x_{t-1}^3 x_{t-4}) + 2284.14727(x_{t-1}^5 + x_{t-1}^4 x_{t-4}))}. \quad (17)$$

The explained variance for the training set is 0.923. In order to test the predictive power of this equation, we apply it to a new data set. Starting with a completely new series of the form (16) we use Eq. (17) to estimate fifty one-period-forward predictions. The fit between the predicted values and the series is very good: in the example presented in Fig. 4, R^2 equals 0.933. Furthermore, it is especially interesting to note that the genetically evolved equation predicted the peaks correctly, in every instance, whether they occurred after four periods ($t=5$, 21, and 28), or after three periods. Two-dimensional phase portraits of the simulated Belousov-Zhabotinskii reaction, according to Eq. (16), and of the reconstruction bred by the algorithm [Eq. (17)] are depicted in Fig. 5. The similarity between the two is apparent.

(d) Hénon attractor (with noise). As another example, let us investigate the Hénon attractor [39], with noise added. The time series that we examine have either dynamic noise δ,

$$x_t = 1 - 1.4x_{t-1}^2 + 0.3x_{t-2} + \delta, \quad (18)$$

or measurement noise μ,

$$y_t = 1 - 1.4y_{t-1}^2 + 0.3y_{t-2},$$

$$x_t = y_t + \mu. \quad (19)$$

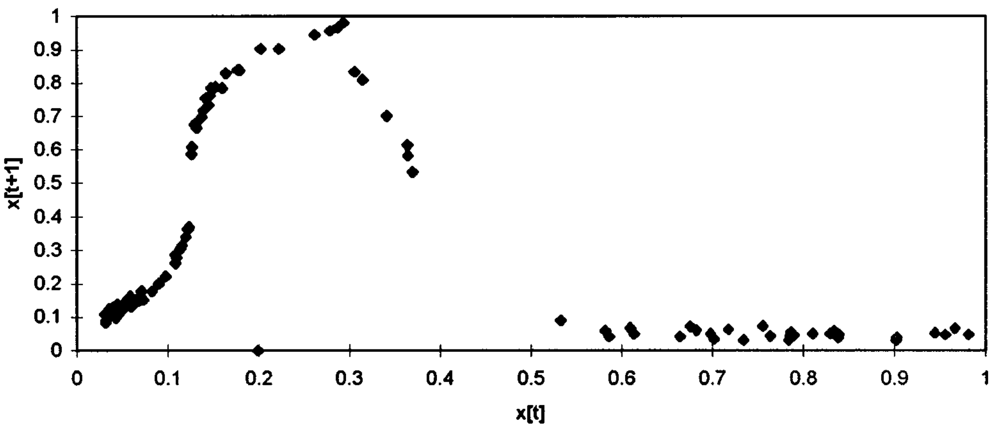
Convergence to well-adapted (fit) equation strings was significantly slower for the noisy time series, and in the following examples, the runs included up to 1200 generations. Data requirements were not increased, however, and the training period was held constant at 100 data points. For a Hénon attractor, with dynamic noise δ distributed as a normal, with mean 0, and standard deviation 0.05 (truncated at 0.01, so that the time series does not always explode), we received, for example,

$$A_0 = ((1.5 - (\mathrm{A1} * \mathrm{A1}))/2.0),$$

$$A_0 = ((0.2 - ((\mathrm{A1} * 0.4) * \mathrm{A1})) * 4.0),$$

$$A_0 = (0.8 - (\mathrm{A1} * \mathrm{A1})).$$

The explained variances for the training sets were 0.442, 0.620, and 0.839, respectively. The structure of these equation strings shows distinct similarity to the structure of the clean dynamic that generated the time-series—in fact they represent the logistic equation, which can be considered as a



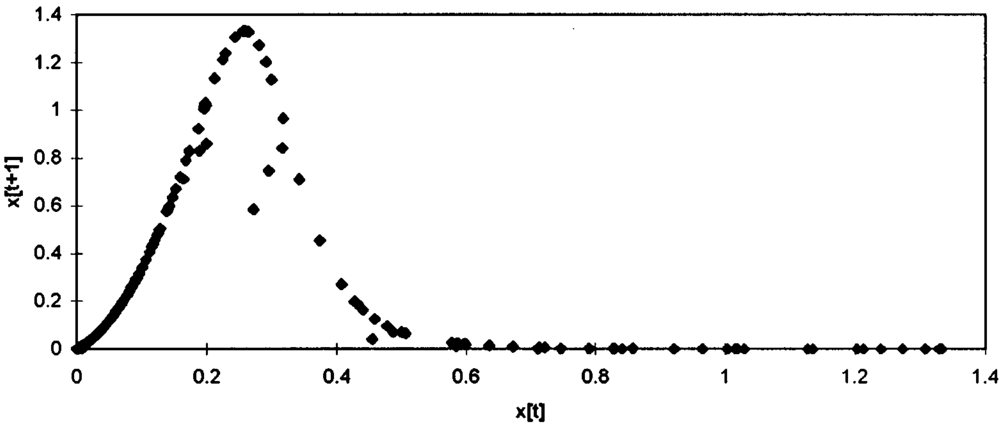


FIG. 5. *B-Z* series with noise, phase portraits. Top: simulation [Eq. (16)]; bottom: forecasts [Eq. (17)].

one-dimensional version of the Hénon process—and the R^2 values for new data sets were the same as for the training sets. (The series that exploded were excluded [40].) For δ uniformly distributed in [0, 0.1] we received similar results.

Turning to measurement noise, we investigated time series, whose noise term μ is normally distributed with standard deviation 0.20. As can be seen from Fig. 6 (top), the structure of the dynamic is hardly recognizable with that amount of noise. Nevertheless, after 500 generations one run of the genetic algorithm produced the equation string

$$A_0=((((1.4-\mathrm{A}3)+((\mathrm{A}7-(\mathrm{A}3-\mathrm{A}8))$$
$$-((\mathrm{A}1+\mathrm{A}1)*\mathrm{A}1)))+\mathrm{A}2)/4.5),$$

with $R^2=0.443$. In tests with new data sets, the explained variance of this equation string was typically between 0.250 and 0.450, which is surprisingly good, considering the amount of noise present. The edited version of this string is

$$x_t=0.311-0.444x_{t-1}^2+0.222x_{t-2}$$
$$+0.222(-2x_{t-3}+x_{t-7}+x_{t-8}), \qquad (20)$$

and some similarity to the "true" dynamic is apparent. Figure 6 (bottom) plots forecasts against actual values, and the correlation between them is evident—even if it is much less than perfect.

(e) Sunspots. As a final example we apply the genetic algorithm to a time series whose underlying dynamic is as yet unknown, the series of sunspots from 1700 until 1992, as registered in the Sunspot Index Data Center [41]. We divide the series into a training set, which runs from 1700 to 1899, to which the algorithm is applied, and a test set (1900 to 1992), which represents the test series. After only about a dozen generations the algorithm settles down to an equation string like A5*A1/A5 or similar, with R^2 of 0.621. This, again, is not surprising, because of the high degree of autocorrelation in the series. As mentioned in Sec. III, we can refine the results. Since we know that $x_t=f(x_{t-1})+\varepsilon_t$, we determine the best linear fit, by minimizing the sum of squared residuals, $\Sigma\varepsilon_t^2$, and receive

$$x_t=7.7762+0.8205x_{t-1}+\varepsilon_t, \qquad (21)$$

with R^2 of 0.673. Now we take the series of residuals, ε_t, and let the algorithm breed equations in terms of $x_{t-\lambda}$, with lags $1\leqslant\lambda\leqslant20$. (Hence the training set actually consists of the years 1720 to 1900, the first 20 observations being used as lags for the beginning of the series.) Five separate runs resulted in the following equation strings after about 200 generations:

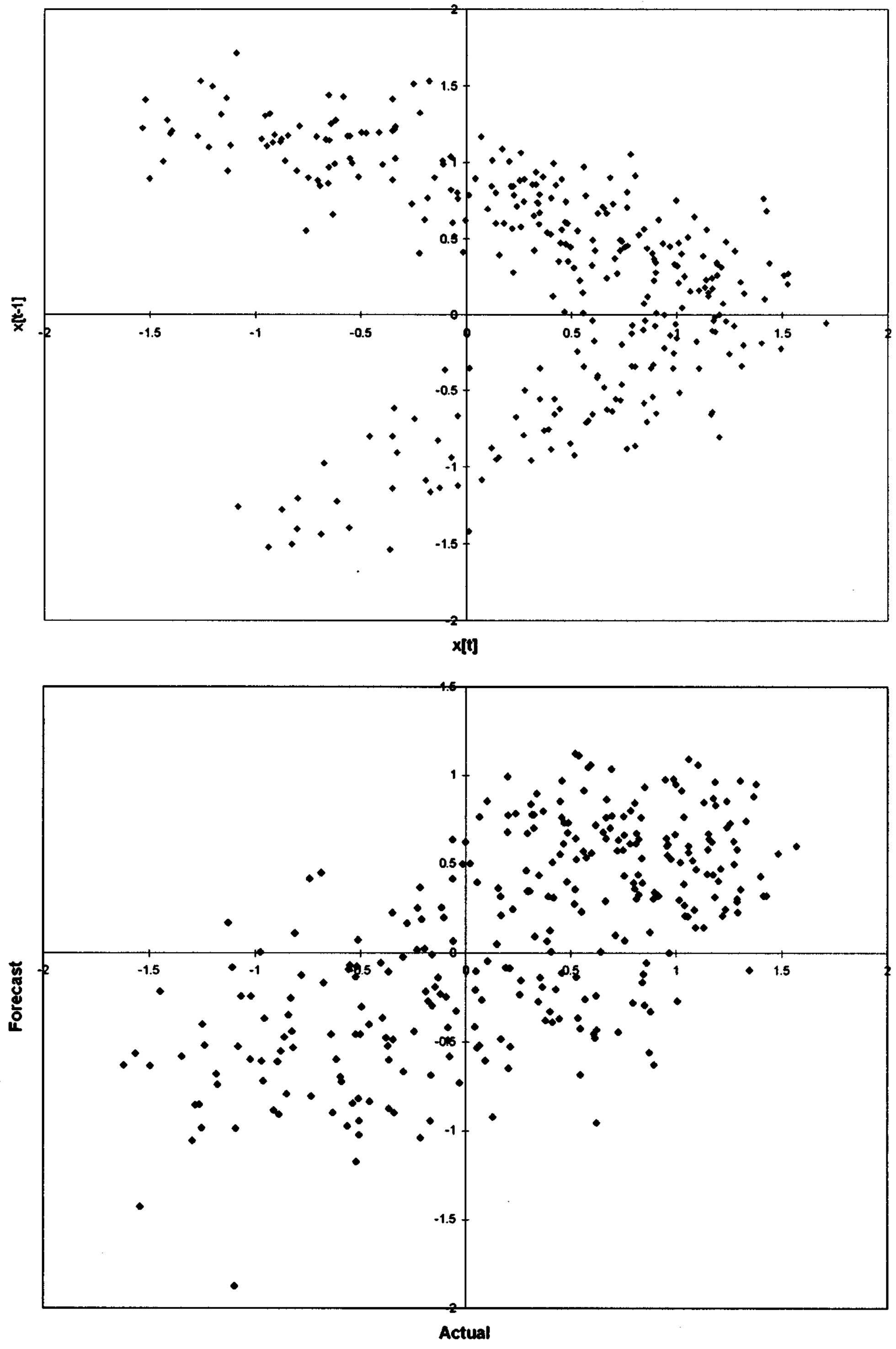


FIG. 6. Hénon attractor with measurement noise. Top: phase portrait; bottom: actual values vs forecasts.

$$A_0 = ((\text{A9}-\text{A3})/((\text{A9}/(4.0*\text{A1}))+4.0)),$$

$$A_0 = (((\text{A9}-\text{A3})*\text{A1})/(\text{A13}+((\text{A9}+\text{A2})+(9.4+\text{A2})))),$$

$$A_0 = ((\text{A9}-\text{A3})/(4.8+(\text{A9}/(9.5*\text{A1})))),$$

$$A_0 = ((\text{A9}-\text{A3})/(3.9+(\text{A11}/\text{A1}))),$$

$$A_0 = ((\text{A9}-\text{A3})/((\text{A13}/\text{A1})+3.9)),$$

with R^2 of 0.468, 0.538, 0.440, 0.462, and 0.457, respectively. These R^2 values refer to the variance that remained after the algorithm in the first stage already helped explain 67.3% of the total variance. Hence, the combination of both stages explains between 0.814 and 0.849 of the training series' variance, e.g.,

$$0.673 + 0.538(1-0.673) = 0.849. \tag{22}$$

It is particularly interesting to note that the edited versions of

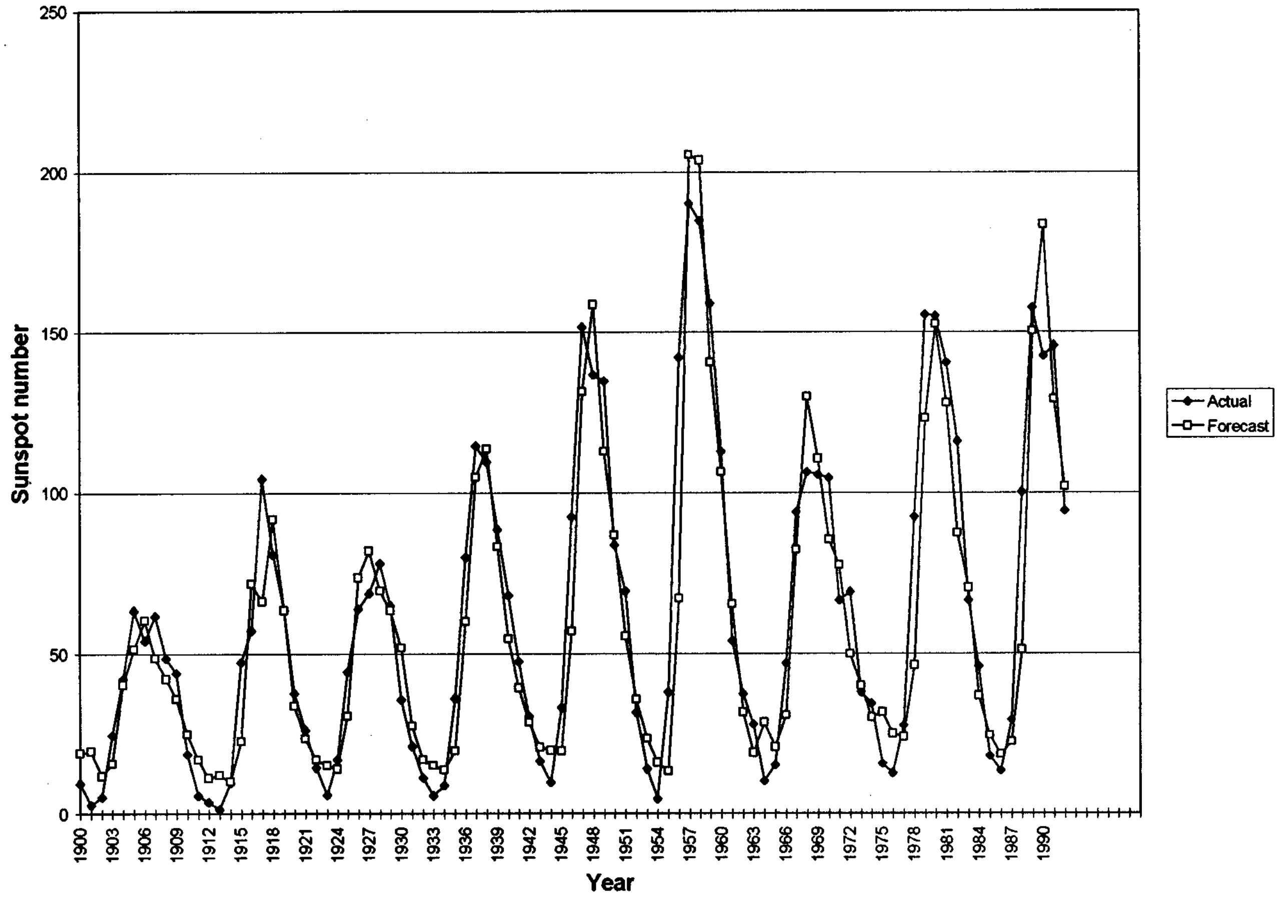


FIG. 7. Sunspots.

theses strings are very similar, with the expression $x_{t-1}(x_{t-9}-x_{t-3})$ appearing in every single case:

$$\varepsilon_t=\frac{x_{t-1}(x_{t-9}-x_{t-3})}{4x_{t-1}+\frac{1}{4}x_{t-9}},\quad \varepsilon_t=\frac{x_{t-1}(x_{t-9}-x_{t-3})}{9.4+2x_{t-2}+x_{t-9}+x_{t-13}}$$

$$\varepsilon_t=\frac{x_{t-1}(x_{t-9}-x_{t-3})}{4.8x_{t-1}+\dfrac{1}{9.5}x_{t-9}},\quad \varepsilon_t=\frac{x_{t-1}(x_{t-9}-x_{t-3})}{3.9x_{t-1}+x_{t-11}},$$

$$\varepsilon_t=\frac{x_{t-1}(x_{t-9}-x_{t-3})}{3.9x_{t-1}+x_{t-13}}. \tag{23}$$

Applying the combined results of both stages,

$$x_t=7.7762+x_{t-1}\left(0.8205+\frac{(x_{t-9}-x_{t-3})}{Q}\right), \tag{24}$$

to the series, 1900 to 1992, where Q designates any one of the divisors on the right-hand sides of Eqs. (23), we obtain R^2 values of between 0.851 and 0.872. Figure 7 plots the series and the forecasts for one of Eq. (23) [42].

V. DISCUSSION

In order to assess the ability to forecast time series, the Mackey-Glass differential equation, with $\tau=30$, and the series of sunspots during the past three centuries are often employed as benchmarks to test for the quality of different methods. The normalized, root-mean-square error E, which relates to our R^2 as

$$E=\sqrt{1-R^2}, \tag{25}$$

is frequently used as a measure for the goodness of fit. Farmer and Sidorovich [7] applied the nearest-neighbor technique, to forecast 40 periods in the future. Using 100 data points they reported $E\sim1.0$, which corresponds to an R^2 value of approximately zero. When the training set was increased to 5000 observations, the nearest-neighbor technique allowed 40-period forecasts with R^2 values of 0.990. Lapedes and Farber's [11] neural networks, using 500 observations for training purposes, produced one-period forecasts with goodness of fit of 0.999. The wavelet method reported in Cao *et al.* [43] also gave one-step-ahead predictions, using 300 observations as a training set. In their study the relative errors were reported, and even though it is difficult to quantify the results (given in their Fig. 2), it seems that while the vast majority of forecast errors is smaller than 0.025, a sizable amount is found between 0.025 and 0.05, and a significant number is much larger, some being as high as 0.200. Finally, Meyer and Packard's [17] genetic algorithm allows 400-period forecasts, but only for those observations of the time series that are located in a very limited part of a seven-dimensional embedding space.

Let us recall that, using training sets with a length of only 100 data points, our genetic algorithm was able to produce one-period forecasts with a goodness of fit of at least 0.990, and forty-period forecasts with R^2 values of between 0.400 and 0.650. About 72% of the absolute errors were smaller than 0.025, another 22% were between 0.025 and 0.05, and even the largest errors were less than 0.130.

Turning to sunspots for the years 1921 to 1955, a recalculation of Subba Rao's [44] results gave an R^2 value of 0.926, while Weigend *et al.* [45] predict this part of the series with an accuracy of 0.914. This compares with an explained variance of 0.895 of the genetic algorithm. For the years 1956 to 1979, Weigend *et al.* received R^2 of 0.650, while our genetic algorithm gave 0.826.

In conclusion, it appears that forecasts produced by our genetic algorithm are at least comparable, if not superior, to some of the other techniques. Moreover, the method has the advantages of giving global estimates, and sometimes indicating the structure of the underlying dynamic. Finally, it may be possible to use genetic algorithms as an alternative method to distinguish chaotic time series from random data.

VI. CONCLUDING REMARKS

It may seem nothing less than surprising that an algorithm that performs unmotivated operations on an arbitrary choice of short equations, with a meager choice of parameter values, should produce formulas that recreate, or at least mimic, the dynamics of a process that underlies a data set. While the lack of a goal, and of an appropriate justification for the path by which this goal is reached, may seem a theoretical weakness, it actually points to the essence of evolution. Starting in preambrian times cells, plants, animals, apes and, finally, humankind evolved from a very meager soup of initial building blocks in such an unmotivated manner. With successive generations nature evolved individuals that were better and better adapted to their environment, without any motivating force. No goal, objective, purpose, or sense of direction was required. Genetic algorithms emulate nature's process, and their significance lies precisely in the fact that a black-box method like the one described in this paper produces successful predictions.

The fact that the formulas bred by the genetic algorithm may be very intricate—their complication sometimes exceeding that of the original system—does not spoil the argument. Evolution has to make do with the semifinished, intermediate products that are available at any given point in time, without regard to elegance of design or engineering. For example, it is by no means certain that, say, the human eye is the simplest mechanism that allows vision. Nature's and genetic algorithms' sole aim—if it can be called that—is to evolve systems that are well adapted to the current environment.

ACKNOWLEDGMENT

The author would like to thank Sergiu Hart, Naftali Tishby, and Benny Shanon for helpful comments.

[1] P. Grassberger and I. Procaccia, Phys. Rev. Lett. **50**, 448 (1983).
[2] A. Ben Mizrachi, I. Procaccia, and P. Grassberger, Phys. Rev. A **29**, 975 (1984).
[3] P. Grassberger, R. Hegger, H. Kantz, C. Schaffrath, and T. Schreiber, Chaos **3**, 127 (1993).
[4] G. G. Szpiro, Physica D **65**, 289 (1993).
[5] K. Judd and A. Mees, Physica D **82**, 426 (1995).
[6] We shall sometimes call previous values *lagged* values, and the time span between x_t and $x_{t-\lambda}$ the lag.
[7] J. D. Farmer and J. J. Sidorowich, Phys. Rev. Lett. **59**, 845 (1987).
[8] R. Doerner, B. Hübinger, and W. Martienssen, Phys. Rev. E **50**, 12 (1994).
[9] M. Casdagli, Physica D **35**, 335 (1989).
[10] J. Moody and C. J. Darken, Neural Comput. **1**, 281 (1989).
[11] A. Lapedes and R. Farber, Los Alamos National Laboratory Technical Report No. LA-UR-97-2662, 1987 (unpublished).
[12] A. M. Albano, A. Passamante, T. Hediger, and M. E. Farrell, Physica D **58**, 1 (1992).
[13] *Nonlinear Modeling and Forecasting*, edited by M. Casdagli and S. Eubanks, SFI Studies in the Sciences of Complexity (Addison-Wesley, New York, 1992).
[14] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction*, SFI Studies in the Sciences of Complexity (Addison-Wesley, New York, 1994).
[15] U. Parlitz and G. Mayer-Kress, Phys. Rev. E **51**, 2709 (1995).
[16] J. B. Ramsey and Z. Zhang, in *Proceedings of the First International Conference on High Frequency Data in Finance* (Olsen and Associates, Zurich, Switzerland, 1995).
[17] T. Meyer and N. H. Packard, in *Nonlinear Modeling and Forecasting* (Ref. [13]), pp. 249–264.
[18] Jie Lin, Y. Bartal, and R. E. Uhrig, Nucl. Technol. **111**, 46 (1995).
[19] Cao Liangyue, Hong Yiguang, Fang Haiping, and He Guowei, Physica D **85**, 225 (1995).
[20] X. Guan, R. J. Mural, and E. C. Uberbacher, in *Proceedings of the Tenth Conference on Artificial Intelligence for Applications* (IEEE Computer Society Press, Morgantown, 1994).
[21] J. Bauer, *Genetic Algorithms and Investment Strategies* (Wiley, New York, 1994).
[22] E. M. Azoff, *Neural Networks Time Series Forecasting of Financial Markets* (Wiley, New York, 1994).
[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. (University of Michigan Press, Ann Arbor, 1992).
[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Reading, MA, 1989).
[25] T. Brodmeier and E. Pretsch, J. Comput. Chem. **15**, 588 (1994).
[26] Y. Zeiri, Phys. Rev. E **51**, 2769 (1995).
[27] P. Gerstoft, J. Acoust. Soc. Am. **97**, 2181 (1995).
[28] J. Piper, Patt. Recog. Lett. **16**, 857 (1995).
[29] R. G. Palmer, W. B. Arthur, J. H. Holland, B. LeBaron, and P. Taylor, Physica D **75**, 264 (1994).
[30] A. Prügel-Bennett and J. L. Shapiro, Phys. Rev. Lett. **72**, 1305 (1994).
[31] J. R. Koza, *Genetic Programming* (MIT Press, Cambridge, 1992).

[32] In the language of search and optimization algorithms, mutation ensures that the population of agents does not get stuck on a local maximum in the ''landscape'' of the search-space.

[33] In a manner of speech, ''heavy'' components of the equation may hide the ''light'' ones during the first run of the algorithm. Once the effect of the heavy components has been isolated, the genetic algorithm may—in the second stage—discover remaining parts of the formula.

[34] O. E. Rössler, Phys. Lett. A **57**, 397 (1976).

[35] For simplicity we write AZ for x_{t-z} when reproducing equation strings.

[36] M. C. Mackey and L. Glass, Science **197**, 287 (1977).

[37] When referring to one-period forecasts we simply mean the next entry in the time series, and similarly for n-period predictions. The *period* does not necessarily refer to the characteristic time of the chaotic process.

[38] K. Matsumoto and I. Tsuda, in *The Theory of Dynamical Systems and its Application to Nonlinear Problems*, edited by Hiroshi Kawakami (World Scientific, Singapore, 1984).

[39] M. Hénon, Commun. Math. Phys. **50**, 69 (1976).

[40] The ability to predict such explosions would be a desirable feature of the algorithm. However, a training period could include at most one explosion which may not suffice to train the genetic algorithm for this type of occurrence. Further investigation is needed in this matter.

[41] See *Neural Networks Time Series Forecasting of Financial Markets* (Ref. [22]), pp. 104–105.

[42] The 11-year cycle that is generally believed to underlie the sunspot series, is recreated by Eq. (23), even when random initial data are used to produce different realizations of the series. The formulas bred by the algorithm could therefore provide an indication that the 11-cycle may actually be produced by a combination of 3- and 9-cycles.

[43] L. Cao, Y. Hong, H. Fang, and G. He, Physica D **85**, 225 (1995).

[44] T. Subba Rao, in *Nonlinear Modeling and Forecasting* (Ref. [13]), pp. 199–228, in particular Table 2.

[45] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, in *Nonlinear Modeling and Forecasting* (Ref. [13]), pp. 395–432.