A Quadratic Empirical Model Formulation for Dynamical Systems using a Genetic Algorithm

S. E. HAUPT
Applied Research Laboratory
P.O. Box 30
The Pennsylvania State University
State College, PA 16804, U.S.A.
haupts2@asme.org

(Received March 2005; revised and accepted October 2005)

Abstract—A new procedure to formulate nonlinear empirical models of a dynamical system is presented. This nonlinear modeling technique generalizes the Markovian techniques used to build linear empirical models, but incorporates a quadratic nonlinearity. The model fit is accomplished using a genetic algorithm.

The nonlinear empirical model is applied to two low order model test cases demonstrating different forms of nonlinearity. The two equation predator/prey model (Lotka-Volterra equations) is modeled in the regime of a stable limit cycle. The nonlinear empirical model is able to capture the general shape of the limit cycle, but does not display the long time stability. The second example is the three dimensional Lorenz system forced in the chaotic regime. The general shape and location in phase space of the chaotic attractor is reproduced by the nonlinear empirical model.

The results presented here demonstrate that nonlinear empirical models may be able to reproduce some of the nonlinear behaviors of dynamical systems. © 2006 Elsevier Ltd. All rights reserved.

Keywords—Empirical models, Inverse models, Lorenz system, Predator/prey model, Genetic algorithms, Dynamical systems.

1. INTRODUCTION

Numerical modeling of time dependent problems such as those found in fluid dynamics has traditionally involved using some type of time stepping scheme, either explicit or implicit, combined with known dynamics discretized from a partial differential equation. Sometimes, however, the details of the dynamics are not sufficiently known or we wish to develop a model that reproduces dynamic behavior without involving the details of the full physical equations. In such cases, we can substitute an empirical model based on observed data. These stochastic empirical models are often Markovian and are built from either measured or simulated data. Given a sufficient amount of data to develop the model, the discretized dynamics of the traditional dynamical model can
be replaced by a matrix of computed values that serve as a propagator matrix. Such models have become popular for empirical modeling of geophysical fluids in recent years [1-9]. Some advantages of using empirical inverse models include that they save valuable computer time when it is not necessary to know the details of the time evolution, they can be used to force a forward dynamical model, they can be used to directly produce an equilibrium state in response to an imposed forcing, and they are useful as a diagnostic tool to identify the most rapidly growing modes of the system.

Linear empirical models are reasonably straightforward to produce from data using standard least squares inversion techniques. For highly nonlinear fluid dynamics problems, however, linear models are no longer adequate [2]. In these cases, a nonlinear term is often necessary to capture the dynamics. Since in fluid dynamics nonlinearity often enters as the quadratic advective term, the nonlinearity of the empirical model is also likely to be of quadratic form. Achatz and Brandstator [10] added nonlinear terms based on the dynamics of the problem to form a hybrid between the dynamics (nonlinear terms) and an empirical model (linear terms). Here, we seek to add the nonlinear terms fully empirically. Unfortunately, nonlinear models are more difficult to devise due to the introduction of higher order tensors to the problems. We resolve this issue through redefining the problem in terms of optimization and directly searching for the propagator matrix given the data. We show that solution is possible using an artificial intelligence (AI) technique such as a genetic algorithm (GA).

Section 2 introduces the linear empirical modeling technique, and then presents our generalization to the nonlinear form. The GA solution methodology is described. Two examples of fully nonlinear dynamical systems are presented in the following sections: the predator/prey model in Section 3 and the Lorenz system behavior in the chaotic regime in Section 4. Section 5 summarizes the findings and speculates on potential uses.

2. MODEL FORMULATION

2.1. Linear Empirical Models

A linear, time-varying model can be written in the form,

$$\dot{s}_i = B_{ij} s_j + \xi_i,$$

(1)

where $s_i$ is the $N$-dimensional state and can represent states such as velocities at various locations or the spectral coefficients of the velocity,

$\dot{s}_i$ is the time rate of change of the state,

$B_{ij}$ is a linear $N \times N$ tensor that relates the above two,

$\xi_i$ is a vector of white noise.

The deterministic dynamics are contained in the second order tensor, $B_{ij}$. Nonlinearities are parameterized by corrections to $B_{ij}$ as well as within the noise, $\xi_i$. This simple linear form is easily fit using standard analytical techniques to minimize the least square error between the model and time series data. These techniques involve minimizing the square of the error between the model and time series data. The error is minimized in a least square sense by requiring that

$$E = \left\langle \left\{ (\dot{s}_i - B_{ij} s_j)^2 \right\} \right\rangle$$

(2)

is minimized.

The parentheses represent a spatial averaging while the angle brackets represent an ensemble time average. The solution to this problem involves finding where the derivative of $E$ vanishes. In doing this, one computes the covariance and lagged covariance matrices as

$$\Lambda_{ij} = \langle s_i(t) s_j(t) \rangle$$

is the covariance tensor, averaged over time;

$$\Lambda_{r,ij} = \langle s_i(t + \tau) s_j(t) \rangle$$

is the lagged covariance tensor;

$\tau$ is the chosen lag time.
Upon solving the least squares problem, we find

\[ B_{ij} = \ln \left( \frac{\Lambda_{i,j}}{\Lambda_{ij}} \right) / \tau \]  

or in terms of a propagator equation, we can write

\[ s_j(t + \tau) = G_{ij}(\tau)s_j(t), \]  

where

\[ G_{ij}(\tau) = \exp(B_{ij} \tau). \]  

Note that the model depends on the value of the lag, \( \tau \), the amount of data used to build it, and the resolution of the data. For more detailed discussion of the basis of this model and suggestions for how to formulate it, see the work of [2].

Such linear empirical models often compete well with linearized dynamical models in reproducing the statistics of the modeled field [1-9]. The El Nino/Southern Oscillation phenomenon has been well reproduced by this technique [2,11-13]. Kondrashov et al. [14] extended the technique to a nonlinear form using multiple polynomial regression enabling capture of higher order moments of ENSO. In addition, an empirical model of the climate was shown to reproduce a response to an imposed forcing better than a linearized dynamical model [6]. The modes of the matrix give an indication of optimal growth [15]. Also, much can be diagnosed about the flow dynamics [6].

2.2. Nonlinear Empirical Models

When the dynamics are highly nonlinear, stochastic linear dynamics cannot be expected to reproduce all of the system's behavior [2]. Thus, it is convenient to formulate stochastic empirical models using nonlinear dynamics. We choose to concentrate on quadratic nonlinearity as follows.

1. It is the most reasonable to calculate. As the order of the nonlinearity increases, so does the order of the tensor that must be fit for the nonlinear term. The higher the order of the problem, the more data is necessary to obtain a good fit.
2. A quadratic deterministic form is sufficient to produce the entire range of coupled dynamical behavior, such as limit cycles and chaotic motion.
3. The forward dynamical models used in geophysical fluid dynamics use a quadratic term to specify the nonlinear advection. Menke [16] describes generalization of a model to higher order nonlinear terms.

Here, we formulate the quadratic empirical model as

\[ \dot{s}_i = C_{ijk} s_j s_k + B_{ij} s_j + \xi_i. \]  

The nonlinear interactions now occur explicitly through the nonlinear third order tensor operator, \( C_{ijk} \). \( B_{ij} \) is again an \( N \times N \) tensor that serves as the linear propagator, \( C_{ijk} \) is an \( N \times N \times N \) third-order tensor that gives the coefficients of the quadratic interactions, and \( \xi_i \) is the constant noise vector. We wish to compute the tensors \( B_{ij} \) and \( C_{ijk} \) so that the least square error of (6).

\[ E = \left( \langle \dot{s}_i - B_{ij} s_j - C_{ijk} s_j s_k \rangle \right)^2 \]  

is minimized. (7)

The angle brackets denote a time average. Since (6) is a quadratic generalization of (1), standard methods can be used for determining \( B_{ij} \) and \( C_{ijk} \) (see, for instance reference [16]). Minimizing \( E \) with respect to \( B_{ij} \) and \( C_{ijk} \) gives the system of equations,

\[ T_{mn}^{(3)} + C_{ijk} T_{jkmn}^{(4)} = 0, \]  

\[ B_{ij} = \left[ \langle \dot{s}_i s_m \rangle - C_{ilp} \langle s_l s_p \rangle \right] \left( \langle s_m s_j \rangle \right)^{-1}, \]  

(8)
where
\[
T_{mn}^{(3)} = (s_t s_m s_n) - (s_i s_j) (s_j s_k)^{-1} (s_k s_m s_n),
\]
\[
T_{kmn}^{(4)} = (s_j s_k s_l) (s_l s_p)^{-1} (s_p s_m s_n) - (s_j s_k s_m s_n).
\]

Although this is a closed form solution, to compute the third order tensor \( C_{ijk} \) requires inverting the fourth-order tensor \( T_{jkmn}^{(4)} \). Such an inversion is not trivial. Therefore, we choose to instead compute \( C_{ijk} \) in equation (8) by doing a best fit with a genetic algorithm.

2.3. Genetic Algorithms

The genetic algorithm is an optimization technique fashioned after the biological concepts of genetics and evolution [18–23]. There are many flavors of GA. For this problem, it is convenient to use the continuous parameter GA, which is posed in terms of real numbers. The flow chart in Figure 1 provides an overview of the technique. The variables being minimized become the basic building blocks, or genes, which are concatenated to form a one-dimensional array called a chromosome. Initially, a population of chromosomes is randomly generated. The “goodness,” or cost, of each chromosome is evaluated by the objective function, also known as the cost function. The “fittest,” or lowest cost chromosomes survive while the less fit, higher cost ones die off. 1 This process mimics natural selection in the natural world. The lowest cost survivors are paired to mate. The mating process combines information from the two parents to produce two offspring. Some of the population experiences mutations. The process iterates and each new member of the population is evaluated by the cost function.

There are a variety of methods to pair the chromosomes for mating. Some popular methods are reviewed by Haupt and Haupt [22,23]. The method used here pairs the chromosomes according to numerical rank. The mating method is a combination of an extrapolation method with a crossover method and was designed to simulate the advantages of the binary genetic algorithm mating scheme [22,23]. It begins by randomly selecting a parameter in the first pair of parents to be the crossover point.

\[
\alpha = \text{roundup} \{ \text{random} \times N_{\text{par}} \}
\]
Beginning with the parent chromosomes,

\[
\text{parent}_1 = [p_{m1}p_{m2} \cdots p_{ma} \cdots p_{mN_{par}}],
\]
\[
\text{parent}_2 = [p_{d1}p_{d2} \cdots p_{da} \cdots p_{dN_{par}}],
\] (11)

where the \( m \) and \( d \) subscripts discriminate between the *mom* and the *dad* parent. The selected parameters are combined to form new parameters that will appear in the children,

\[
P_{\text{new}1} = p_{ma} - \beta [p_{ma} - p_{da}],
\]
\[
P_{\text{new}2} = p_{da} - \beta [p_{ma} - p_{da}],
\] (12)

where \( \beta \) is a random value between 0 and 1. Crossover is completed by swapping the left side of the *mom* with the right side of *dad* and visa versa to form,

\[
\text{offspring}_1 = [p_{m1}p_{m2} \cdots p_{\text{new}1} \cdots p_{dN_{par}}],
\]
\[
\text{offspring}_2 = [p_{d1}p_{d2} \cdots p_{\text{new}2} \cdots p_{mN_{par}}].
\] (13)

If the first parameter of the chromosomes is selected, then only the parameters to the right of the selected parameter are swapped and similarly if the last parameter of the chromosome is selected. This method allows offspring parameters outside the bounds set by the parent if \( \beta \) is greater than one. In this way, information from the two parent chromosomes is combined in a way that mimics the crossover process during meiosis.

If care is not taken, the genetic algorithm could converge too quickly to a local minimum. To avoid premature convergence, mutations, or random changes in some of the parameters, are introduced. A gene selected for mutation is merely substituted with a new random number in the specified range. The operations of mating and mutation allow the GA to continue to explore the solution space while combining information on the best solutions to gravitate towards the optimal solution. The benefits of using a GA include that they

- do not require a good first guess,
- simultaneously search from a wide sampling of the objective function surface,
- deal with a large number of parameters,
- are well suited for parallel computers,
- optimize parameters with extremely complex objective function surfaces,
- work well for nonlinear problems.

Such advantages outweigh GAs' lack of rigorous convergence proofs and speed.

For the nonlinear empirical model, the parameters to optimize are the elements of \( C_{ijk} \). Symmetries can be invoked to reduce the dimensionality of the problem. Each chromosome is a "guess" at the correct solution to \( C_{ijk} \). The elements of \( C_{ijk} \) become the genes that are concatenated into chromosomes. The elements of \( B_{ij} \) are computed from (8). The cost function for this problem is (7). The values of \( s_i \) and its time derivative, \( \dot{s}_i \), are based on a time series of data for the problem of interest, then the time summation in (7) done over the entire time series. Therefore, the goal of the optimization is to correctly fit the \( C_{ijk} \) tensor, then compute the accompanying \( B_{ij} \) so that the expected error of the predictor equation is statistically minimized.

### 3. EXAMPLE 1—PREDATOR/PREY MODEL

We begin with time series data from the predator-prey model (also known as the Lotka-Volterra equations). For a simple application, see [24].

\[
\frac{dx}{dt} = ax - bxy,
\]
\[
\frac{dy}{dt} = -cy + dxy.
\] (14)
Figure 2. Time series of predator/prey variation with time (equation (14)).

Figure 3. State space plot of predator/prey variation with time (equation (14)).

Figure 4. Least squares linear fit to (14).

Here, $x$ is the number of prey and $y$ the number of predators. The prey growth rate is $a$ while the predator death rate is $c$. Parameters $b$ and $d$ characterize the interactions. Equations (14) are integrated using a fourth-order Runge Kutta with a time step of 0.01 and parameters $a = 1.2$. 
Figure 5. Time series of predator-prey interactions as computed by the nonlinear empirical model.

Figure 6. The predator/prey relation in state space as computed by the nonlinear model with parameters fit by the GA.

Figure 7. Evolution of the minimum cost.

$b = 0.6$, $c = 0.8$, and $d = 0.3$. The time series showing the interaction between the two variables appears in Figure 2. This time series serves as the data for computing the inverse models. The
phase space plot of these data appears as Figure 3 where we see the limit cycle between the predators and the prey.

The least squares fit to the linear empirical model of equation (1) appears in Figure 4. We note that the agreement is quite poor, as one would expect given that the system (14) is highly nonlinear in this range of parameters. With no nonlinear interaction available, the number of prey would grow while the number of predators remains stationary.

The time series data is then modeled using the nonlinear empirical model of equation (6). The nonlinear model fit was done according to the discussion of Section 2, using a GA to fit $C_{ijk}$. The GA used an initial population size of 200, a working population size of 100, a crossover rate of 0.5, and a mutation rate of 0.2. A time series of the solution as computed by the GA appears in Figure 5. Note that although the time series does not exactly reproduce the data (in fact, we see nonphysical negative population members), the oscillations with a phase shift of roughly a quarter period is reproduced. The wavelength is not exact and the amplitudes grow in time, indicating an instability. This instability is likely inherent in the way that the model is matched. However, the reproduction of such a difficult nonlinear system is quite an improvement over the linear empirical model of Figure 4.

The state space plot of a time integration of the nonlinear empirical model appears in Figure 6. The limit cycle is not exactly reproduced. The nonlinear empirical model instead appears to be unstable and slowly growing. However, in comparison with the linear least squares model, which resulted in merely a single expanding curve (not shown), the nonlinear empirical model was much better at capturing the cyclical nature of the oscillations.

Finally, Figure 7 shows the convergence of the GA for a typical run of fitting the nonlinear model (6) to the data. Note that due to their random nature, the results of the GA are never exactly the same. In particular the convergence plots will differ each time. However, the results are found to be repeatable for this problem when the algorithm is initialized with differing random seeds.

4. EXAMPLE 2—LORENZ EQUATIONS

A second example of a nonlinear empirical model examines whether it is possible to capture the chaotic behavior of the three equation Lorenz system [25], which can be written as

$$\begin{align*}
\dot{x} & = -\sigma x + \sigma y, \\
\dot{y} & = \rho x - y - xz, \\
\dot{z} & = -bz + xy,
\end{align*}$$

where $x, y, z$ are the lowest order coefficients of a truncated series of atmospheric flow and we use parameters, $\sigma = 10, b = 8/3, \rho = 28$. These parameters produce a chaotic regime that results in a strange attractor, often referred to as the “butterfly” attractor. Equations (15) are integrated using a fourth-order Runge-Kutta method to produce the data trajectory shown in Figure 8, depicted in three-dimensional phase space.

A nonlinear empirical model of these data is created using the techniques presented above. The parameters of the GA are an initial population of 500, working population of 100, crossover rate of 0.5, and mutation rate of 0.3 for a total of 200 generations. Taking into account symmetries for this problem results in 18 unique parameters in the $C_{ijk}$ tensor to find. For this highly nonlinear regime, it is difficult to find a solution. Not every attempt converged to a small residual of the cost function. It required multiple attempts to produce the time trajectory shown in Figure 9. Although the match is not perfect, the general shape of the strange attractor is replicated.

For comparison, the solution due to a linear empirical model fit of (1) is shown in Figure 10, similar to that shown previously in [2]. The linear model is not able to capture the shape of the attractor, but instead shows a decaying spiral behavior. Although the parameters have been
chosen to model the chaotic regime, the linear model can only capture a homoclinic orbit toward a stable fixed point.

5. DISCUSSION

This work has demonstrated that nonlinear empirical models show promise for capturing the essential dynamics of nonlinear systems. Although these models can be posed in terms of closed form solutions, they are not easily solved given the necessity of inverting a fourth-order tensor. Here, a genetic algorithm was successful at completing this inversion and computing the best fit coefficients.

The example problems demonstrate that for nonlinear systems that cannot be modeled adequately by the linear form of the empirical model, the essence of both limit cycles and chaotic attractors can be captured through application of a nonlinear empirical model when coupled with
the artificial intelligence methodology of genetic algorithms. The nonlinear model (6) when combined with a GA reproduces the shape of the attractor to the Lorenz equations reasonably well compared to the essential decay of the linearized method. For a chaotic system, we do not expect to reproduce the exact time behavior and are rather pleased to see the form of the attractor near to that of the actual data used to create the model.

These nonlinear empirical models are not perfect. However, they show great ability at not only reproducing the behavior of complex models, but also predicting future behavior better than more simplified models and reproducing a response to forcing better than linearized forward dynamical models. Such models are only beginning to be explored. The fact that even linear empirical models have proven useful for forecasting some systems brings hope that nonlinear empirical models may prove to be even more so for systems where nonlinearity is an essential feature of the intrinsic dynamics. Future work will concentrate on demonstrations with larger systems and with systems where there may be reasons to believe that the actual dynamics are not well modeled by known model equations.

REFERENCES