6.5 # NONLINEAR EMPIRICAL MODELING

Sue Ellen Haupt
Utah State University, Logan, UT

1. INTRODUCTION

Empirical models have gained popularity in recent years as an alternative to the more traditional dynamical models (for example see Hasselman 1988, Penland 1989, Branstator and Haupt 1998). Linear empirical models are easy to produce from data using standard least squares inversion techniques. Numerical modeling of time dependent flows has traditionally involved using some type of time stepping combined with known dynamics discretized from a partial differential equation. However, sometimes 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 equations of motion. In such cases, we can substitute an empirical model based on observed data. These stochastic empirical models are built from measured or simulated data and are based on a Markhov model. Given a sufficient amount of data to develop the model, the discretized dynamics of the traditional model can be replaced by a matrix of computed values that serve as a propagator matrix.

However, for highly nonlinear fluid dynamics problems, linear models are no longer adequate. 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 Branstator (1999) 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 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 using an artificial intelligence technique such as a genetic algorithm (GA).

* Corresponding author address: Sue Ellen Haupt, Computational Mechanics Division, Applied Research Laboratory, Penn State University, Post Office Box 30, State College, PA 16804; email: suehaupt@engineering.usu.edu

2. MODEL FORMULATION

Linear Empirical Models

First, we introduce the linear, time-varying model, which can be written in the form:

$$\frac{d\vec{s}}{dt} = B\vec{s} + \vec{x} \qquad (1)$$

where: $\vec{s}$ is the N-dimensional state and can represent states such as velocities at various locations or the spectral coefficients of the velocity

$\frac{d\vec{s}}{dt}$ is the time rate of change of the state

B is a linear $N \times N$ matrix that relates the above two

$\vec{x}$ is a vector of white noise (denoting the error in the fit)

The dynamics are contained in the matrix, B. Nonlinearities are parameterized by corrections to B as well as within the noise portion of ?. 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 the time averaged data tendencies. Specifically, we wish to minimize the noise vector (equivalently, the error). We assume that $\vec{s}$ behaves as a Markov process. We can minimize the error in a least square sense by requiring that:

$$E = \left\langle \left\{ \left(\frac{d\vec{s}}{dt} - B\vec{s} \right)^2 \right\} \right\rangle \text{ is minimized} \qquad (2)$$

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 vanishes. In doing this, one computes the covariance and lagged covariance matrices as:

$\Lambda = \left\langle s(t)^T s(t) \right\rangle$ is the covariance matrix, averaged over time. The superscript (T) indicates a matrix transpose.

$\Lambda_t = \left\langle s(t+t)^T s(t) \right\rangle$ is the lagged covariance matrix.

t is the chosen lag time.

Upon solving the least squares problem, we find:

$$B = \ln\left(\frac{\Lambda_t}{\Lambda}\right)/t \qquad (3)$$

or in terms of an propagator equation, we can write

$$s(t+t) = G(t)s(t) \qquad (4)$$

where:

$$G(t) = \exp(Bt) \qquad (5)$$

Note that the model depends on the value of the lag, t , the amount of data used to build it, and the resolution of the data.

Such linear empirical models often compete well with linearized dynamical models in reproducing the statistics of the modeled field. In addition, much can be learned about the flow dynamics.

Nonlinear Empirical Models

When the dynamics are highly nonlinear, stochastic empirical models can also be formulated using nonlinear dynamics. For instance, a quadratic empirical model could have the form:

$$\vec{s}_t = C\vec{s}^T\vec{s} + B\vec{s} + \vec{x} \qquad (6)$$

The nonlinear interactions now occur explicitly through the nonlinear third order tensor operator, *C*. Thus *B* is an *N x N* matrix that serves as the linear propagator, *C* is an *N x N x N* third order tensor that gives the coefficients of the quadratic interactions, and x is the constant noise vector.

We wish to compute the matrices *B* and *C* so that the least square error of (12),

$$E = \left\langle \left(\vec{s}_t - B\vec{s} - C\vec{s}^T\vec{s} \right)^2 \right\rangle \text{ is minimized} \qquad (7)$$

The angle brackets denote a time average. Minimizing *E* with respect to *B* and *C* gives the system of equations:

$$\begin{aligned} &T_3 + CT_4 = 0 \\ &B = \langle ss_t \rangle - C\langle sss \rangle \langle ss \rangle^{-1} \end{aligned} \qquad (8)$$

where:

$$\begin{aligned} &T_3 = \langle sss_t \rangle - \langle ss_t \rangle \langle ss \rangle^{-1} \langle sss \rangle \\ &T_4 = \langle sss \rangle \langle ss \rangle^{-1} \langle sss \rangle - \langle ssss \rangle \end{aligned} \qquad (9)$$

Although this is a closed form solution, to compute the third order tensor *C* requires inverting the fourth order tensor T_4 (Haupt and Weiss 1998). Such an inversion is not trivial. Therefore, we choose to instead compute *C* in equation (8) by doing a best fit with a genetic algorithm.

Genetic Algorithms

The flow chart in Figure 1 provides a “big picture" overview of a continuous genetic algorithm, that is, one in which the parameters are real numbers. The parameters are the genes which are strung together in a one-dimensional array known as a chromosome. The GA begins with a population of chromosomes which are fed to the cost function for evaluation. The fittest chromosomes survive while the highest cost ones die off. This process mimics natural selection in the natural world. The lowest cost survivors mate. The mating process combines information from the two parents to produce two offspring. Some of the population experiences mutations.

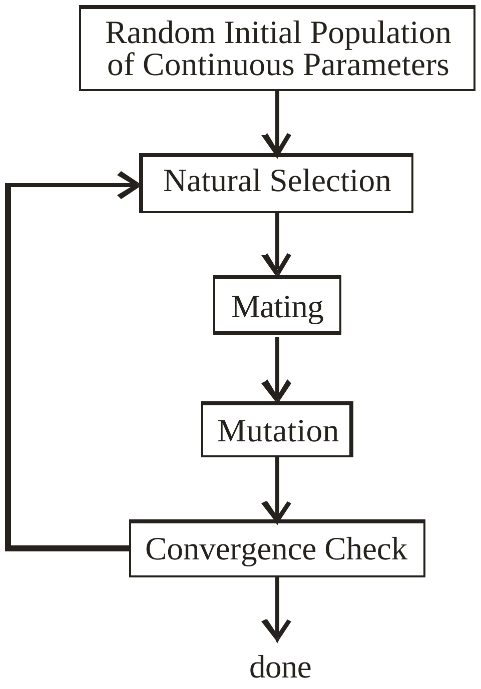

Figure 1. Flowchart of continuous parameter genetic algorithm.

As seen in the figure, the first step of a continuous parameter genetic algorithm is creating the population of chromosomes. First, the real parameters are concatenated together into a chromosome as:

$$chromosome = \lfloor p_1 p_2 \cdots p_a \cdots p_{N_{par}} \rfloor \quad (10)$$

where the p_i are the parameters and there are a total of N_{par} parameters. The parameters are simply floating point numbers. The encoding function needs only keep track of which digits represent which parameters and to make sure they are within given bounds. A population of such chromosomes is created using a random number generator so that the chromosome arrays are gathered together in a two dimensional matrix. Once the chromosomes have been created, their cost or fitness must be evaluated. This is done by the cost or objective function, which is very problem specific. The lowest cost chromosomes (N_{keep}) remain in the population while the higher cost ones are deemed less fit and die off. The reduced population is then the portion of the population available for mating.

There are a variety of methods to pair the chromosomes for mating. Some popular methods are reviewed by Haupt and Haupt (1998). Here, we choose to pair the chromosomes according to numerical rank. After the cost function evaluation, the chromosomes are sorted in order from lowest cost to highest. That is, the nth chromosome will have a probability of mating of:

$$P_n = \frac{N_{keep} - n + 1}{\sum_{k=1}^{N_{keep}} k} \quad (11)$$

Then the cumulative probabilities are used for selecting which chromosomes mate.

Once two parents are chosen, some method must be devised to produce offspring which are some combination of these parents. Many different approaches have been tried for crossing over in continuous parameter genetic algorithms (Adwuya 1996, Haupt and Haupt 1998).

The method used here is a combination of an extrapolation method with a crossover method. We wanted to find a way to closely mimic the advantages of the binary genetic algorithm mating scheme. It begins by randomly selecting a parameter in the first pair of parents to be the crossover point.

$$a = roundup\{random \times N_{par}\} \quad (12)$$

We'll let

$$\begin{aligned} parent_1 &= \lfloor p_{m1} p_{m2} \cdots p_{ma} \cdots p_{mN_{par}} \rfloor \\ parent_2 &= \lfloor p_{d1} p_{d2} \cdots p_{da} \cdots p_{dN_{par}} \rfloor \end{aligned} \quad (13)$$

where the *m* and *d* subscripts discriminate between the *mom* and the *dad* parent. Then the selected parameters are combined to form new parameters that will appear in the children:

$$\begin{aligned} p_{new1} &= p_{ma} - b[p_{ma} - p_{da}] \\ p_{new2} &= p_{da} - b[p_{ma} - p_{da}] \end{aligned} \quad (14)$$

where b is also a random value between 0 and 1. The final step is to complete the crossover with the rest of the chromosome as before:

$$\begin{aligned} offspring_1 &= \lfloor p_{m1} p_{m2} \cdots p_{new1} \cdots p_{dN_{par}} \rfloor \\ offspring_2 &= \lfloor p_{d1} p_{d2} \cdots p_{new2} \cdots p_{mN_{par}} \rfloor \end{aligned} \quad (15)$$

If the first parameter of the chromosomes is selected, then only the parameters to right of the selected parameter are swapped. If the last parameter of the chromosomes is selected, then only the parameters to the left of the selected parameter are swapped. This method does not allow offspring parameters outside the bounds set by the parent unless b is greater than one. In this way, information from the two parent chromosomes is combined a way that mimics the crossover process during meiosis.

If care is not taken, the genetic algorithm converges too quickly into one region of the cost surface. If this area is in the region of the global minimum, that is good. However, some functions have many local minima and the algorithm could get stuck in a local well. If we do nothing to solve this tendency to converge quickly, we could end up in a local rather than a global minimum. To avoid this problem of overly fast convergence, we force the routine to explore other areas of the cost surface by randomly introducing changes, or mutations, in some of the parameters. A mutated parameter is replaced by a new random parameter.

3. EXAMPLE 1 – PREDATOR/PREY MODEL

We begin with time series data from the predator-prey model (also known as the Lotka-Volterra equations), namely:

$$\frac{dx}{dt} = ax - bxy \qquad\qquad (16)$$
$$\frac{dy}{dt} = -cy + dxy$$

where 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 (16) were integrated using a fourth order Runge Kutta with a time step of 0.01 and parameters a=1.2, b=0.6, c= 0.8, and d=0.3. The time series showing the interaction between the two appears as Figure 2. This time series serves as the data for computing the inverse models below. 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 model appears in Figure 4. We note that the agreement is quite poor, as one would expect given that the system (16) is highly nonlinear. With no nonlinear interaction available, the number of prey grows while the number of predators remains stationary.

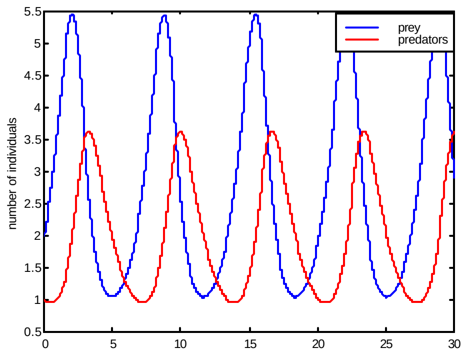

Figure 2. Time series of predator/prey variation with time (eq. 16).

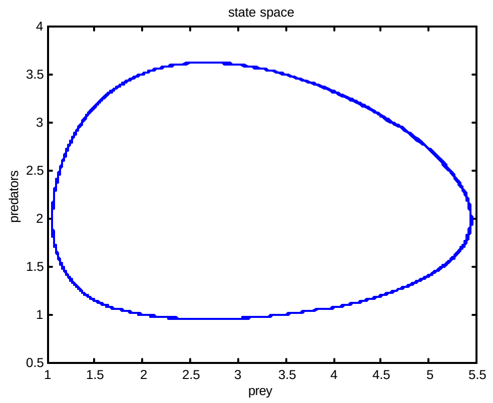

Figure 3. State space plot of predator/prey variation with time (eq. 16).

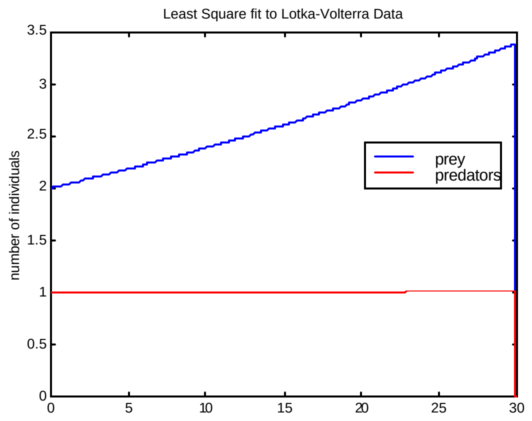

Figure 4. Least Squares linear fit to (16).

To obtain a more appropriate nonlinear fit, we now choose to model the data with the nonlinear model of eq. (6). The nonlinear fit was done according to the discussion of section 2 and used a GA to fit the parameters. The GA used an initial population size of 200, a working population size of 100, 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, 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 amazing given the comparison to traditional linear models.

The state space plot of a time integration of the nonlinear empirical model appears in Figure 6. Once again, the limit cycle is not actually reproduced. The nonlinear model instead appears unstable and slowly grows. However, the comparison with the linear least squares model resulted in merely a single slowly growing curve (not shown). The nonlinear empirical model was able to capture the cyclical nature of the oscillations much better than that.

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, it is amazing how the results are so reliable.

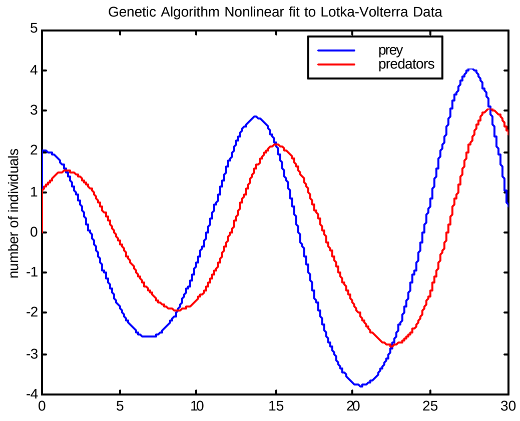

Figure 5. Time series of predator-prey interactions as computed by the nonlinear empirical model.

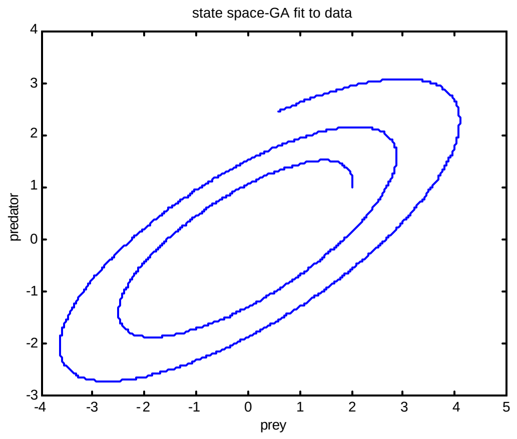

Figure 6. The predator-prey relation in state space as computed by the nonlinear model with parameters fit by the GA.

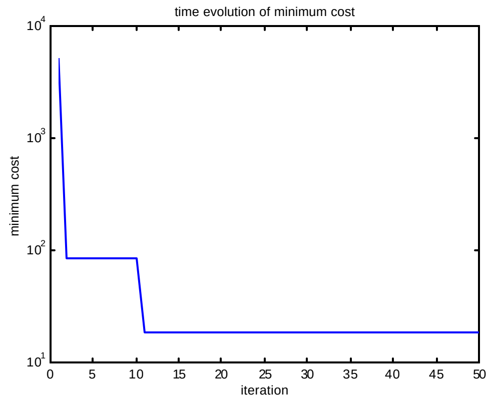

Figure 7. Evolution of the minimum cost.

5. EXAMPLE 2 – LORENZ EQUATIONS

A second example of a nonlinear empirical model examines whether it is possible to capture the chaotic behavior of the Lorenz system (Lorenz 1963) which can be written:

$$\begin{aligned}\dot{x} &= -sx + sy \\ \dot{y} &= rx - y - xz \\ \dot{z} &= -bz + xy\end{aligned} \tag{17}$$

where x, y, z are the lowest order coefficients of a truncated series of atmospheric flow and we use parameters: $s = 10, b = 8/3, r = 28$ These parameters produce a chaotic regime that results in a strange attractor. The equations (17) were integrated using a fourth order Runge-Kutta method to produce the data in Figure 8.

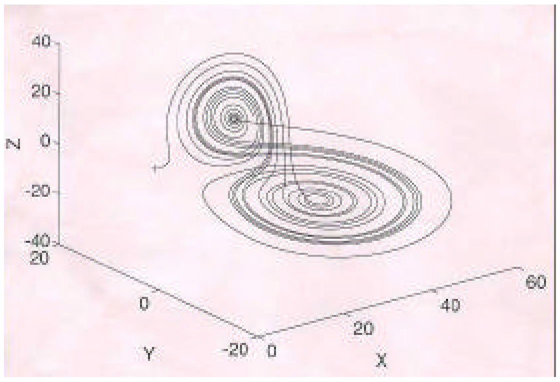

Figure 8. A Lorenz attractor computed by integrating equations (17) in time for 2000 steps.

We wish to create an empirical model of these data using the techniques presented above. The parameters of the GA are an initial population of 500, working population of 100, mutation rate of 0.3 for a total of 200 generations. Taking into account symmetries for this problem results in 18 unique parameters to find. For this highly nonlinear regime, we were able to find a solution, which when propagated via equation (6) produces the time evolution depicted in Figure 9. Although the match is not perfect, we have replicated the general shape of the strange attractor and the size of the domain is approximately correct.

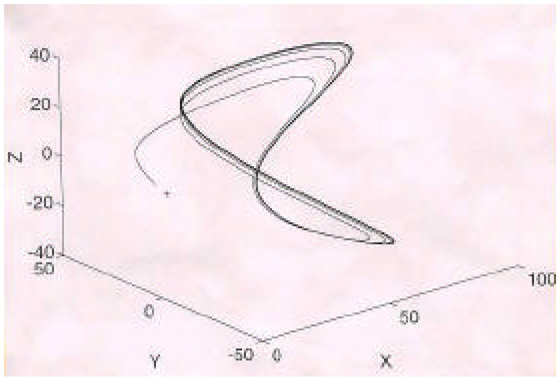

Figure 9. Nonlinear model of Lorenz attractor (equation 12) as computed with a GA.

For comparison, the solution is compared to a linear model fit, which is merely the linear part of (12). For this portion, there is a simple closed form solution that is easily computed. The linear match is shown in Figure 10. We see that the linear model is not able to capture the shape of the attractor, but instead shows a decaying spiral behavior.

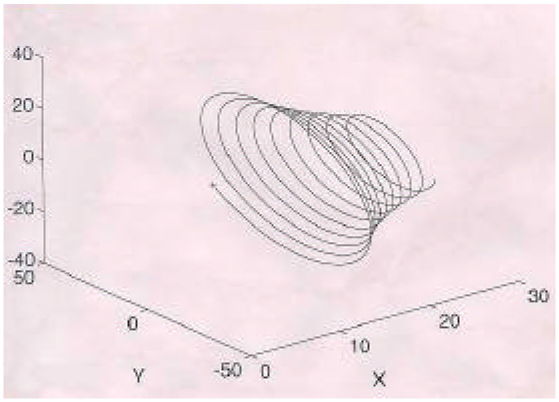

Figure 10. Linear model of the Lorenz equations.

6. CONCLUSIONS

We have shown 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, we have used a genetic algorithm to complete this inversion.

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 reproduced the shape of the attractor to the Lorenz equations reasonably well compared to the essential decary 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.

Empirical models are not perfect. However, they show great ability at not only reproducing the behavior of complex models, but also serve to predict future behavior better than more simplified models and reproduce a response to forcing better than linearizxed forward dynamical models. Such models are only beginning to be explored.

ACKNOWLEDGMENTS

Elements of this work were inspired through discussions with colleagues and students, including Grant Branstator, Jeffrey Weiss, and Christina Perez.

REFERENCES

Achatz, U., and G. Branstator, 1999: A Two-Layer Model with Empirical Linear Corrections and Reduced Order for Studies of Internal Climate Variability. *J. Atm. Sci*, **56**, , pp. 3140–3160.

Adewuya, A.A., 1996. New Methods in Genetic Search with Real-Valued Chromosomes, Master's Thesis, Cambridge, Massachusetts Institute of Technology.

Branstator, G. and S.E. Haupt, 1998: An empirical model of forced barotropic atmospheric anomalies, *J. Climate*, **11**, 2645-2667.

Hasselman, K., 1988: PIPs and POPs: The reduction of complex dynamical systems using principal interaction and oscillation patterns. *J. Geophys. Res.*, **93**, 11015-11021.

Haupt, R.L. and S.E. Haupt, 1998: *Practical Genetic Algorithms*, New York: John Wiley & Sons, 177 pp.

Haupt, S.E. and J. Weiss, 1998: A Quadratic Inverse Model of the Lorenz Equations, Abstract Volume of the 1998 Fall Meeting of the American Geophysical Union Abstract Volume, San Francisco, CA.

Lorenz, E.N., 1963: Deterministic Nonperiodic Flow, *J. Atmos. Sciences*, **20**, 130-141.

Penland, C., 1989: Random forcing and forecasting using principal oscillation pattern analysis. *Mon. Weather Rev.*, **117**, 2165-2185.