Reconstructing noisy dynamical systems by triangulations

Stuart Allie*
Institute for Nonlinear Science, University of California at San Diego, San Diego, California

Alistair Mees†
Institute for Nonlinear Science, University of California at San Diego, San Diego, California

Kevin Judd
Centre for Applied Dynamics and Optimization, University of Western Australia, Nedlands, 6907, Australia

David Watson
Centre for Applied Dynamics and Optimization, University of Western Australia, Nedlands, 6907, Australia

(Received 22 July 1996)

We show how to construct triangulations of noisy data from dynamical systems. We model the dynamics using piecewise linear or C^1 models defined over the triangulation. The number and positions of vertices of the triangulation are selected by the minimum description length criterion. We test the method on two artificial data sets and on experimental data from a chaotic electronic circuit. The models reproduce the qualitative aspects of the data as well as quantitative aspects such as correlation dimension and periodic points. [S1063-651X(97)00201-8]

PACS number(s): 05.45.+b

I. INTRODUCTION

This paper deals with an approach to modeling dynamical systems from data, particularly scalar-valued time series. In recent years there has been a great deal of work in the field of modeling from time series with particular emphasis on low-dimensional nonlinear systems that exhibit chaotic behavior. This paper relates closely to the work described in [1] and [2]. Mees [1] deals with the triangulation of noise-free data, while Mees and Judd [2] use the concept of *minimum description length* (MDL) to select the optimal model from a given class (specifically, radial basis models). The present work uses piecewise linear or piecewise C^1 models based on the triangulation of the state space.

In comparison to other model classes (see, for example, [3–5]) we note that our model class is very flexible in its ability to fit the data, is continuous, local, and does not possess any artificial symmetry (cf. radial basis models.) We also note that the size of the model itself will be determined from the data, reducing the number of arbitrarily chosen parameters.

Rather than simply constructing the triangulation of all the data points we attempt to find a ‘‘minimal’’ triangulation that best models the data. The criterion that is used to select the minimal triangulation is that of description length, as described by Rissanen [6]. The essential idea adopted here is that we choose the model with the shortest code length (in some optimal encoding scheme) that enables us to reproduce completely the data set. Following a description of the method, we discuss some example applications and conclude with a summary, caveats, and comments on future research.

II. OUTLINE OF METHOD

Consider a dynamical system with state vectors $z \in M$, where M is an n-dimensional manifold. Assume that the dynamics are described by the equation

$$z_{t+1} = g(z_t) + \xi_t,$$

where ξ_t is the dynamical noise present. We observe the system through a measurement process $\phi : M \to \mathbb{R}$ such that

$$y_t = \phi(z_t) + \eta_t,$$

where η_t is the observational noise. We then embed [7,8] the data set $\{y_t\}$ in d dimensions to give states $\{x_t\}$. An example of such an embedding is a time-delay embedding with fixed ‘‘lag’’ τ given by

$$x_t = (y_t, \ldots, y_{t-(d-1)\tau}) \in \mathbb{R}^d;$$

however, the embedding need not be as regular as this, and indeed we may choose particular embedding strategies depending on the information we wish to extract from the model. We describe the dynamics of the embedded system by

$$y_{t+1} = f(x_t) + \epsilon_t \tag{1}$$

and it is precisely this function $f : \mathbb{R}^d \to \mathbb{R}$ that we desire to model. In general, the dynamics should be described by $y_{t+1} = f(x_t, \epsilon_t)$; however, for small noise, Eq. (1) is usually a good approximation. ϵ_t is a random variate about which we can, in general, say very little.

*Permanent and mailing address: CADO, University of Western Australia, Nedlands, 6907, Australia.

†Permanent address: CADO, University of Western Australia, Nedlands, 6907, Australia.

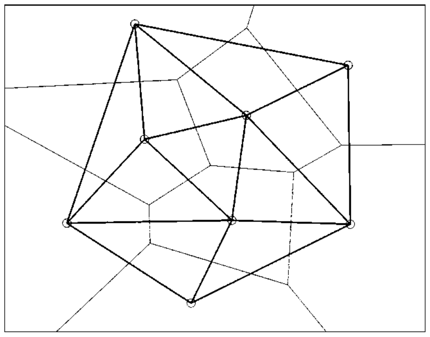

FIG. 1. Delaunay triangulation (heavy lines) and its dual, the Dirchlet tesselation (light lines), for eight points (circles) in the plane.

A. Interpolation on triangulations

The map f defines a hypersurface in $\mathbb{R}^{d+1}$. Our model consists of describing this hypersurface in some way. To do this, we make use of piecewise interpolation on triangulations in d dimensions.

A triangulation of a set of points $V \in \mathbb{R}^d$ is a collection of d-dimensional simplices with disjoint interiors and vertices chosen from V. There are many triangulations for a given set of points. One of the more useful is the Delaunay triangulation [9], which we describe by considering its dual, the Dirichlet tesselation. (The Dirichlet tesselation is also called the Voronoi tesselation, natural-neighbor tiling, or Thiessen tiling in the literature.) Given a set of vertices $V = \{v_i\}$, we associate with each v_i the region consisting of all points that are closer to v_i than to any other point in V. That is, the tile belonging to v_i is

$$T(v_i) = \{x \in \mathbb{R}^d : |x - v_i| < |x - v_j| \forall j \neq i\}.$$

If two tiles share a face, then we say that the corresponding vertices are neighbors. The collection of such tiles forms the Dirichlet tesselation. The dual of this structure, in which we join neighbors by line segments, is the Delaunay triangulation of V. The Delaunay triangulation has the property that its simplices are most nearly equiangular. The Delaunay triangulation is discussed in detail in [10] and efficient algorithms for computing the Delaunay triangulation are described in [11]. Figure 1 shows the Delaunay triangulation and its dual for eight points in the plane.

We now describe how to use triangulations to approximate a surface.

Definition. $\lambda_i(x)$, $i \in N(x)$, are *local coordinates* for x with respect to $\{v_j\}_{j=1}^{n_v}$ if

$$x = \sum_{i \in N(x)} \lambda_i(x) v_i$$

and

$$\sum_{i \in N(x)} \lambda_i(x) = 1,$$

where $N(x)$ is a set of indices to the set $\{v_j\}$.

An approximation to $f(x)$ is then given by

$$\hat{f}(x) = \sum_{i \in N(x)} \lambda_i(x) f(v_i).$$

Some results regarding this type of approximation were proved in [1]. They are reproduced here for convenience.

(i) If $f \in C^1$ then $\hat{f}(x) = f(x) + o(\Delta)$, where

$$\Delta = \text{diameter}[\{x\} \cup \{v_i : i \in N(x)\}].$$

(ii) If f is affine then $\hat{f} = f$.

(iii) If we choose $N(x)$ to index the simplex containing x then our approximation is piecewise linear and continuous in x.

(iv) If we choose $N(x)$ so that the $\lambda_i(x)$ are the so-called *subtile weights* [10] then our approximation is C^1 in x for $x \in \text{co}\{v_j\} \setminus \cup \{v_j\}$, where $\text{co}(x)$ denotes the convex hull of x. These weights (also called natural-neighbor coordinates) are defined as follows. If we were to add the point x to the tesselation, it would create a tile $T(x)$ that is the union of sections of tiles from the existing tesselation. The subtile weight corresponding to a vertex v_i is the normalized (Lebesgue) volume of the section of $T(v_i)$, which would belong to x.

Item (i) above justifies our use of the Delaunay triangulation. The Delaunay triangulation has the property that it minimizes, on average, the diameter of the simplices formed [9,12,13,1] and hence minimizes the average error in approximation that is due to the choice of triangulation.

We would expect this model class to perform at least as well as other local models and to compare well with global models such as radial basis models. A comparison of MDL-selected radial basis models and the models described here is a subject for future research. For data from nonlinear dynamical systems where the dynamics is not overwhelmed by noise, we would expect our models to perform significantly better than statistical models such as the threshold autoregressive models of Tong [14]. Of course, for any given data set, the choice of model is best determined by direct comparison. It is hoped that MDL and flexible model classes (which incorporate other models as special cases) make the selection somewhat less arbitrary, but we are still far from being able to determine automatically the best model for a given case.

B. Fitting to noisy data

The approximation above assumes that we know the values of $f(v_i)$ exactly. In general, of course, this is not true. If our v_i are data points then we have noisy data values as an approximation to $f(v_i)$. If the v_i are not data points, then we have no *a priori* information about the $f(v_i)$. From now on, we will label the approximation to $f(v_i)$ as $h_i = \hat{f}(v_i)$ and call the h_i ''heights.'' The approximation is now written

$$\hat{f}(x) = \sum_{i \in N(x)} \lambda_i(x) h_i,$$

where $N(x)$ is now a set of indices to the set $\{v_j\}$. We might choose the set $\{v_i\}$ to be a subset of our data points $\{x_t\}$, but it is not required.

Assuming we have chosen the v_i in some way, we wish to choose the h_i so as to give the best approximation to f given

$\{x_t\}$ and $\{y_t\}$. We define the local coordinates as previously for $j\in N(x_i)$ and $\lambda_j(x_i)=0$ for $j\notin N(x_i)$. We further define a matrix Λ with elements $\Lambda_{ij}=\lambda_j(x_i)$. We can then write the approximation as a matrix equation thus

$$\hat{f}(x_i)=\sum_{j\in N(x_i)}\lambda_j(x_i)h_j$$

$$=\sum_j \lambda_j(x_i)h_j \qquad (2)$$

$$=(\Lambda h)_i, \qquad (3)$$

where $h=(h_1,\ldots,h_{n_v})^T$. Recalling that we would like y_i to be near $\hat{f}(x_i)$ and taking ϵ_t in 1, we solve for the h_i by solving the least-squares problem defined by minimizing

$$J(v,h)=(y-\Lambda h)^T(y-\Lambda h) \qquad (4)$$

over h, where $y=(y_1,\ldots,y_n)^T$ and v denotes the set of vertices. Let the minimum of $J(v,h)$ over h be $J(v)$. Specifying v determines the model fully, and we now consider how to do this.

C. Vertex selection

We now deal with the problem of how to select the vertices. We choose the v_i from $\{x_t\}$ initially, but then allow the vertices to move in an attempt to find a better model. In order to obtain a set of vertices which are nearly optimal for a given model size (number of vertices), we have used an algorithm of model expansion and contraction. We allow the model to grow by k vertices by selecting the k data points with the worst fitted data values. That is, the data points with the largest values of $|y_t-\hat{f}(x_t)|$. We then remove the k vertices which are (or are closest to) the data points which are best fitted. This process of growing and shrinking the model continues until either there is no change in the set of vertices or $J(v)$ does not decrease.

The algorithm is as follows.

(i) Construct the initial affine model using $d+1$ vertices that form a single simplex containing the entire data set.

(ii) Select the k data points with the worst-fitted data values. Bring these points into the triangulation.

(iii) Select the k vertices with the best-fitted data values. If the vertex is not a data point, use the closest data point. Remove these points from the triangulation.

(iv) Add a small random perturbation to the vertices and keep the positions of vertices that give the smallest $J(v)$.

(v) If the triangulation has changed and $J(v)$ has decreased, go to step (ii).

(vi) Calculate the description length for the current model size.

(vii) Have we found a minimum of description length as a function of number of vertices? If so, stop now.

(viii) Increase the model size by one by bringing the worst-fitted point into the triangulation and go to step (ii).

We identify a minimum in the description length (DL) as a function of model size by storing the DL for each number of vertices and declaring a local minimum if we have not found a smaller value after a certain number of increases in model size.

Cubanski and Cyganski [15] describe an algorithm that appears to be similar to our own. In particular they perform an optimal selection of vertices by a similar process of expansion and contraction. Their approach differs in that they use a fixed (and arbitrary) number of vertices, and the introduction of new vertices is done by testing every possible addition and selecting the one that decreases $J(v)$ the most. As this involves the alteration of the triangulation for each test, this is a computationally intensive approach to the problem and would be infeasible for large data sets. Their emphasis is on their ''systolic'' (expansion-contraction) method and classifier problems. The present paper is more concerned with model selection by the MDL; the problem class and algorithms used to fit the model are not major concerns.

III. DESCRIPTION LENGTH OF TRIANGULATION MODELS

Suppose we wish to compress the data in order to transmit it to someone else. To do so, we must encode sufficient information for the receiver to be able to reconstruct the data completely. We can do this by using what Rissanen calls a ''two-part code'' [6]. First, if we have a probability distribution P on the data it may be used to define an optimal encoding of a realization of a time series $w=\{(y_t,x_t)\}$. It is a well-known result of coding theory [16] that one can encode this information in a code of length bounded below by $-\log_2 P(w)$ bits (and this bound is approachable to better than one bit). In fact, one can do better than this in the present case because the model was constructed from the data. The data impose restrictions on the parameters estimated and hence on the allowable error vectors. The correction for this is nontrivial and is discussed in [17]. We will use the natural logarithm here so our code lengths will be in ''nats.'' The model defines $\epsilon_t=y_t-f(x_t)$ and we have taken P to be Gaussian. For the second part of the code, we must encode information about our model. In our case, this means that we must transmit the values of the vertex positions $v_i\in\mathbb{R}^d$ and their heights $h_i\in\mathbb{R}$.

Thus the total description length for our system is

$$L(w,v,h)=L(w|(v,h))+L(v)+L(h),$$

where $L(X)$ denotes the code length of x. Since the parameter values are, in principle, known to arbitrary precision, we must truncate them in order to transmit them in a finite length code. It is shown in [2] that the description length of a parameter a truncated to relative accuracy δ_a is

$$L(\overline{a})=L^*(\lceil 1/\delta_a\rceil)+L^*(\lceil\ln[2\max\{\overline{a},1/\overline{a}\}]\rceil),$$

where $\overline{a}$ is the truncated value of a and the function L^* is given by

$$L^*(p)=\ln c+\ln p+\ln\ln p+\ln\ln\ln p+\cdots,$$

with the series terminating as soon as one of the terms is reduced to 0 or 1. The number c is the cost to encode small integers and the shortest code length corresponds to $c \approx 2.865$ [6].

The key to the minimum description length method is that we may optimize over the truncation of these model parameters; that is, we optimize over δ_a as described below. The first term in the total description length will in general decrease as the model size increases, due to better fitting, while the other two terms will tend to increase with model size. Hence the model selected by the minimum description length represents a compromise between the competing desires of good fitting and model parsimony. It is worth noting that as the level of noise in the data decreases, we would expect the model selected by the MDL principle to grow. This is indeed observed in experiments.

We have restricted the use of the MDL to the selection of model size for a given model class. In principle, the MDL enables us to choose the best model from a range of model classes. There are, however, practical difficulties to be overcome. Within a given model class, we are consistent in our choice of encoding and comparing the description lengths is sensible. If we mix model classes, then it is difficult to compare the code lengths of the two classes; the problem is akin to comparing algorithm lengths in different programming languages. While the principles are well understood, the practical application of them has yet to be achieved. It is hoped to be able, in the near future, to include into the MDL framework a wide range of model classes and the ability to choose between them.

A. ''Empirical'' MDL calculation

We will denote the precisions of the heights by α_i and of the vertex positions by β_i. In order to calculate α_i we follow the procedure described in [2] whereby the precision α_i is treated as a continuous parameter rather than a discrete one corresponding to truncation of a floating-point representation of the heights. As our model may be written as a matrix equation

$$y = \Lambda h,$$

the analysis is identical to that in [2], and we obtain the α_i by solving the equation

$$(Q\alpha)_i = 1/\alpha_i \tag{5}$$

numerically. Here

$$Q = \Lambda^T \Lambda / \sigma^2,$$

where $\sigma^2 = J(v)/n$ and n is the number of data points.

We now make an approximation and assume that the vertices are independent with regard to the choice of optimal precisions. We ignore the off-diagonal elements of the matrix Q and so the solution to Eq. (5) is given simply by

$$\alpha_i = \frac{1}{\sqrt{Q_{ii}}}.$$

This is properly cautious as it causes us to overestimate the description length. We have observed the difference between calculating the α_i exactly and this approximation to be around 1–5 %. This has a negligible impact on the total description length calculation.

For the vertex positions, treating the β_i as continuous parameters is not very useful for practical calculations. One needs to calculate the second derivative matrix $\partial^2 \Lambda / \partial v_i \partial v_j$ and for data sets of even modest size, this is a reasonably large calculation. Hence we take a simpler and more direct approach to the calculation; in effect, the calculation of the β_i is done in an ''empirical'' fashion.

We reproduce the effect of truncating the parameter (the vertex coordinates) by adding or subtracting an appropriate amount to or from the floating point value for the parameter and calculating the description length of the model with that truncated parameter value. This is done for successive truncations until a minimum is found. This procedure is applied to each coordinate of each vertex in turn. The range of truncations used correspond approximately to precisions ranging (in 1-bit steps) from 12 to 2 bits of accuracy in the parameters. This procedure explicitly assumes the independence of the β_i in the same way that we did for the α_i.

B. Approximate MDL calculations

We can make further approximations to the calculation of description length in order to save computation time. We can, for example, use one precision for all the vertex positions. Three methods come to mind: (i) a fixed number of bits (or nats), (ii) a fixed relative precision, and (iii) a fixed absolute precision for every vertex. Since we are interested only in the model size that gives the minimum description length, the actual values of the DL are unimportant. We have observed experimentally that all three of the above methods return minima at, or very close to, the same place as the more complete DL calculation. The first method also allows a more efficient coding scheme to be used, as each parameter is encoded in the same number of bits.

IV. EXAMPLES: ARTIFICIAL DATA SETS

A. Rössler system

The equations used were

$$\dot{x} = -(y+z), \quad \dot{y} = x + ay, \quad \dot{z} = bx + z(x-c), \tag{6}$$

which generate the Rössler attractor. Here a, b, and c are parameters; we used $a=0.36$, $b=0.4$, and $c=4.5$. The data were generated by numerical integration, with a fixed time step of 0.2, using MATLAB. Dynamical noise with distribution $N(0,0.01)$ was added at each time step. As the scalar time series, 750 points of the x coordinate was used. These data were embedded in three dimensions with a lag of 8 and the model was made to predict one time step ahead, that is, we chose to model the map f defined by

$$y_t = f(x_t),$$

where

$$x_t = (y_{t-1}, y_{t-9}, y_{t-17}).$$

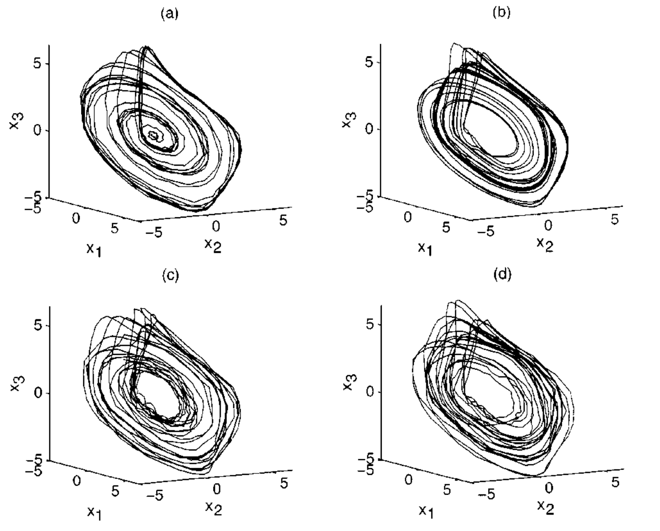

FIG. 2. Rössler system: (a) embedded data set, (b) trajectory of the model with no added noise, and (c) and (d) model trajectories with added noise. The axes $x\hat{1}$, $x\hat{2}$, and $x\hat{3}$ are the components of the embedded vector x_t. Note that the model successfully reproduces the characteristic folding.

The lag was chosen to be close to 1/4 of the major periodic cycle of the data, as it has been found that for data sets with a dominant period, this choice gives ''good'' embeddings. It has been shown elsewhere that the Rössler attractor may be successfully embedded in three dimensions and this is supported by calculations using the ''false-nearest-neighbors'' method on the data [18].

The model chosen by the MDL principle contained 27 vertices and the residuals had a standard deviation of 0.0098, in good agreeement with the known dynamical noise. To demonstrate that this model had in fact correctly extracted information about the dynamics of the system from the data, we produced sample trajectories of the model with randomly chosen initial points. The data set and three trajectories of the model are shown in Fig. 2.

The model is run with dynamical noise added at each step. The noise values used are chosen by a form of bootstrapping [19] by randomly choosing one of the residuals to be the noise added at a given step. This type of ''dynamical bootstrapping'' has been found to give much more realistic trajectories. The analysis of this type of bootstrapping is a topic for further research.

In order to verify that the model orbits are indeed similar to the data, we calculated the correlation dimension for several sample trajectories and compared this to the dimension for the original data. The method used to calculate dimension is that described in [20]. This method produces an estimate of dimension as a function of ''cutoff scale.'' Figure 3 shows the results of such dimension calculations for the original data set and sample trajectories of the model. Although adding noise to the trajectories is the ''right'' thing to do as the model as constructed includes this noise, Fig. 3 shows that this results in trajectories of greater dimension than the original time series. This is probably due to the overly simplistic noise model and bootstrapping used. For example, if the noise has more effect in some areas of the reconstructed state space than in others, this should be reflected in our bootstrapping technique.

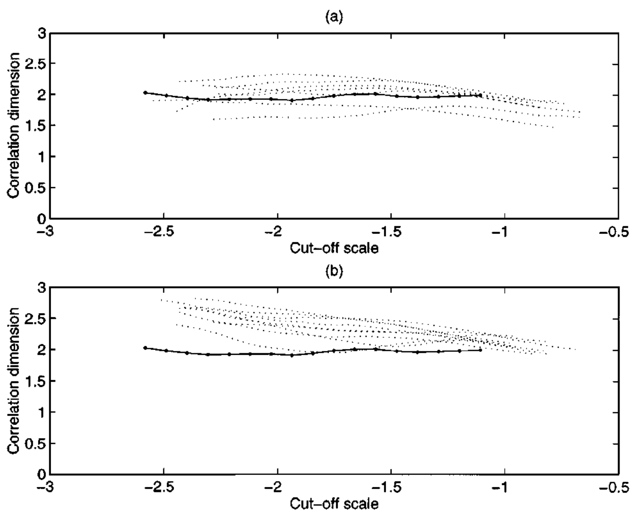

FIG. 3. Rössler system. Correlation dimension as a function of cutoff scale for the original data (solid line) and trajectories of the model (dashed lines). Model trajectories created (a) without dynamical noise and (b) with dynamical noise are shown.

B. Hénon map

For this example we use the map f: $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by

$$f(x,y) = (1 - ax^2 + y, by),$$

with parameters $a = 1.4$ and $b = 0.3$. We generated a time series of 1000 points by iterating this map, with noise of distribution $N(0,0.001)$ added at each iteration. Observational noise with distribution $N(0,0.1)$ was then added. We constructed separate models of each component of the map f.

The x-component model had 22 vertices in total; the y-component model contained only the three vertices enclosing the data, giving (as expected), a linear model. Trajectories of the model, with and without added noise, appear to be qualitatively similar to the original data; see Fig. 4. The

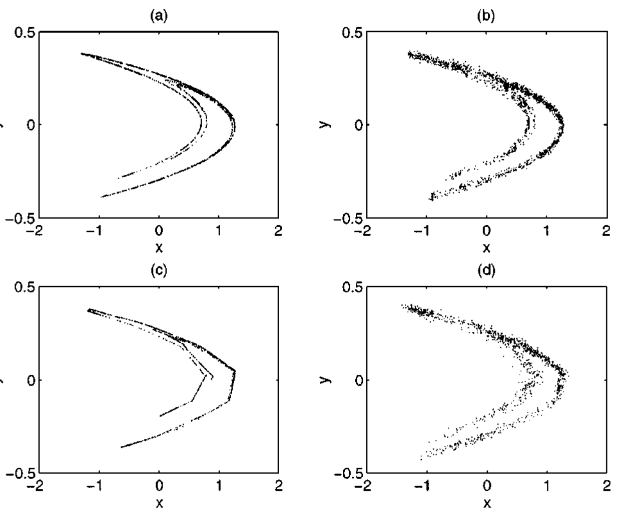

FIG. 4. Hénon map: (a) Data set before observational noise added. (b) data set with observational noise; (c) model trajectory, created with no noise; and (d) model trajectory with noise added.

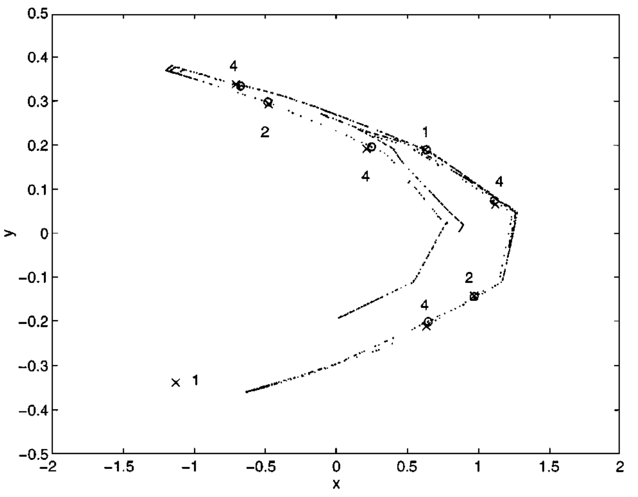

FIG. 5. Model trajectory with the fixed points up to period 4, labeled by their period. Crosses are the actual fixed points of the Hénon map and circles are the fixed points of the model.

model also possesses periodic points up to period 4 in locations similar to the original system and with (mostly) similar Jacobians at the periodic points; see Fig. 5. The periodic points were found using an algorithm that can locate all fixed points of a triangulated map in a finite number of steps. Space precludes our describing it here; it is based on earlier work (see, for example, [21]) and we will describe it elsewhere. Recurrence diagrams [22] for model trajectories indicate that no period-3 point exists, in agreement with the known map. Note that when looking for periodic points, there were some spurious points found, but they were some distance from the data. We could not expect the model to be correct in regions about which it had no information. This is why the model did not reproduce the fixed point at $(-1.13,-0.34)$; there were no data near that point when building the model.

V. EXAMPLES: EXPERIMENTAL DATA

A. Electronic circuit

To test our method on experimental data we used a scalar time series consisting of voltage measurements from a particular nonlinear electronic circuit. This particular circuit displays chaotic behavior; for details of the circuit, see [23]. The data set used to construct the model consisted of 2000 points. The data was embedded in three dimensions with a lag of 5 and the model was constructed to predict one time-step ahead. The embedding dimension was chosen by means of the false-nearest-neighbor method [18] and the lag was chosen to be close to 1/4 of the dominant period of the time series.

The MDL-selected model had 71 vertices. Figure 6 shows the data set and Fig. 7 shows a typical trajectory of the model. The model trajectory was produced using the dynamical bootstrapping method described above; however, the residuals were very small and trajectories produced without noise show the same qualitative behavior. Figure 8 shows the dimension estimates for the data and the model trajectories. We note that the model appears to have reconstructed the system quite well, in terms of both the appearance of the attractor and the dimension estimates.

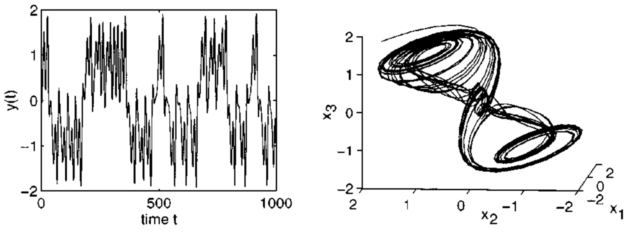

FIG. 6. Electronic circuit: 1000 points of the data as a time series and embedded in $\mathbb{R}^3$. Here we have $y(t)=y_t$ and the axes $x\hat{1}$, $x\hat{2}$, and $x\hat{3}$ are the components of the embedded vector x_t.

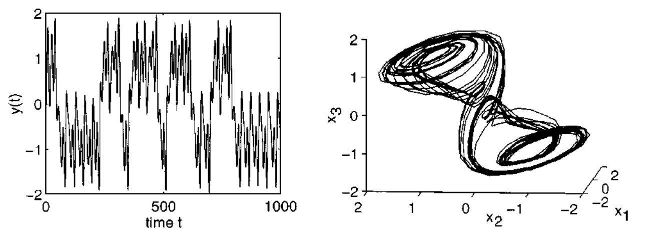

FIG. 7. Electronic circuit: 1000 point trajectory of the model as a time series and embedded in $\mathbb{R}^3$. Here we have $y(t)=y_t$ and the axes $x\hat{1}$, $x\hat{2}$, and $x\hat{3}$ are the components of the embedded vector x_t.

VI. SUMMARY AND CONCLUSIONS

This paper describes how to model dynamical systems from noisy data, using triangulations. It is intended as a demonstration of the potential benefits, problems, and features of modeling using the minimum description length criteria for selecting the model. The model class was piecewise linear interpolation on triangulations. The calculation of the description length used a complete specification of the model, something that has not been done before in this context.

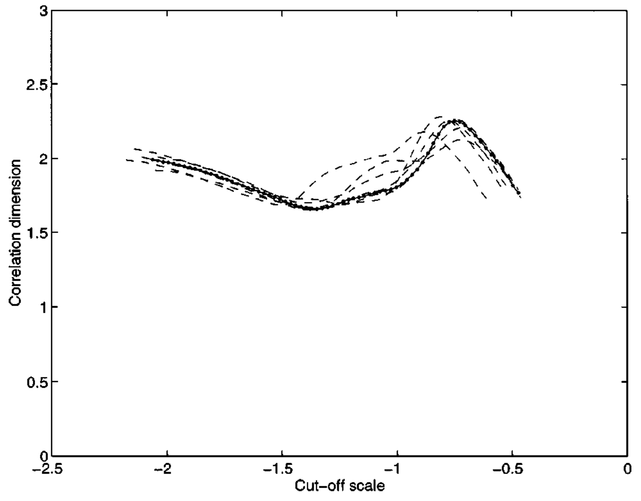

FIG. 8. Correlation dimension as a function of cutoff scale for the original data (solid line) and nine trajectories of the model (dashed lines.)

The techniques described were applied to artificial data sets and some experimental data. The models of the Rössler system and the experimental data produced trajectories that were very similar in appearance to the original data sets and had similar correlation dimensions. It appeared that the models had captured the essential dynamics of the systems. The Hénon model also produced trajectories qualitatively similar to the known system. This model reproduced the position and character of periodic points up to period 4.

There are some known sources of errors in the techniques described in this paper. The first of these is the lack of smoothness of the model class. It seems, however, that, in practice, this is not a major problem. Another error source is the difficulty of the (nonlinear, high-dimensional) optimization problem of finding the *minimum* description length. We have a local minimum to an approximation of the description length. In principle, of course, this is unsatisfactory; in practice, the success of the models will decide the validity of our approximations. The final source of error is in the assumption of normality of the residuals implicit in solving a least-squares problem. Relaxing this assumption means making more of an effort to model explicitly the noise present in the data. This will make the modeling process much more difficult to implement and is an area of ongoing research.

ACKNOWLEDGMENTS

This research was partially supported by a grant from the Australian Research Council. S.A. and A.M. wish to thank The University of Western Australia for leave and financial support. We thank Henry Abarbanel for providing us with the electronic circuit data.

[1] A. I. Mees, Int. J. Bif. Chaos **1**, 777 (1991).
[2] A. I. Mees and K. Judd, Physica D **83**, 426 (1995).
[3] D. S. Broomhead and D. Lowe, Complex Sys. **2**, 321 (1988).
[4] M. Casdagli, Physica D **35**, 335 (1989).
[5] J. D. Farmer and J. J. Sidorowich, Phys. Rev. Lett. **59**, 845 (1987).
[6] J. Rissanen, *Stochastic Complexity in Statistical Inquiry* (World Scientific, Singapore, 1989).
[7] F. Takens, in *Dynamical Systems and Turbulence*, edited by D. A. Rand and L. S. Young (Springer, Berlin, 1981), Vol. 898, pp. 365–381.
[8] T. Sauer, J. A. Yorke, and M. Casdagli, J. Stat. Phys. **65**, 579 (1992).
[9] P. J. Green and R. Sibson, Comput. J. **21**, 168 (1978).
[10] R. Sibson, Math. Proc. Cambridge Philos. Soc. **87**, 151 (1980).
[11] D. F. Watson, Comput. J. **24**, 167 (1981).
[12] D. F. Watson, *Contouring: A Guide to the Analysis and Display of Spatial Data* (Pergamon, Oxford, 1992).
[13] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tesseslations: Concepts and Applications of Voronoi Diagrams* (Wiley, Chichester, 1992).
[14] H. Tong, *Non-linear Time Series: a Dynamical Systems Approach* (Oxford University Press, Oxford, 1990).
[15] D. Cubanski and D. Cyganski, IEEE Trans. Pattern Anal. Mach. Intell. **17**, 403 (1995).
[16] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication* (University of Illinois Press, Urbana, 1949).
[17] J. Rissanen, IEEE Trans. Info. Theory **42**, 40 (1996).
[18] H. D. I. Abarbanel *et al.*, Rev. Mod. Phys. **65**, 1331 (1993).
[19] B. Efron, *The Jackknife, the Bootstrap, and other Resampling Plans* (Society for Industrial and Applied Mathematics, Philadelphia, 1982), Vol. 38.
[20] K. Judd, Physica D **56**, 216 (1992).
[21] G. van der Laan and A. J. J. Talman, *Simplicial Fixed Point Algorithms* (The Mathematical Centre, Amsterdam, 1980), Vol. 129.
[22] J.-P. Eckmann, O. S. Kamphosrt, and D. Ruelle, Europhys. Lett. **4**, 973 (1987).
[23] H. D. I. Abarbanel, L. Korzinov, A. I. Mees, and I. M. Starobinets (unpublished).