1-1994

Analysis of deterministic chaotic signals

William Robertson

ANALYSIS OF DETERMINISTIC CHAOTIC SIGNALS

by

William Robertson

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
MASTER OF SCIENCE
in
Mechanical Engineering

Approved by:

J. S. Török

Dr. Josef S. Török -- Thesis Advisor

Hany Ghoneim

Dr. Hany Ghoneim

Teresa A. Wallace

Teresa Wallace

Charles W. Haines

Dr. Charles W. Haines -- Department Chairman

Department of Mechanical Engineering
College of Engineering
Rochester Institute of Technology
August 1994

ANALYSIS OF DETERMINISTIC CHAOTIC SIGNALS

I, William Robertson, hereby grant permission to the Wallace Memorial Library of the Rochester Institute of Technology, Rochester, NY, to reproduce my thesis in whole or in part providing that no reproduction will be used for commercial use or profit.

William Robertson August 1994

William Robertson

ABSTRACT

The prediction of a single observable time series has been achieved with varying degrees of success. The quality and duration of the prediction is dependent on many factors, the two most important being the reconstruction technique and the quantity of data. The goal of this work is to reduce the computational effort required to achieve satisfactory predictions. Without new methods, which are beyond the scope of this work, this requires a reduction in the size of the data set.

This thesis expands on earlier works using the delay vector space method and the autocorrelation function for reconstruction and applies this analysis technique to a well known non-linear dynamic system. The embedding delay and the sampling rate were varied while keeping the number of points the same in order to study the effects of varying the sampling rate. The results of this experimentation show the importance of the sampling rate and duration of the sample in the reconstruction and prediction. It is shown that the sampling duration may be more important than the number of points. It is apparent from this characteristic that a time series sampled over a longer duration may contain more information in fewer points.

Table of Contents

INTRODUCTION

This thesis is based upon the concept that it is possible to reconstruct the geometric structure of nonlinear dynamical systems from the measurement of a single variable, an idea proposed as early as 1979. This reconstruction then allows the short term prediction of that variable. The underlying theorem indicates that reconstruction is possible when an infinite amount of data is available for a single state variable of the system.

While the proof of the theorem requires an infinite volume of continuous, noise free data, application has shown that a set of 10^4 to 10^5 discrete points is usually sufficient for good reconstruction and prediction capabilities. With the present level of computing power that is available, it is not difficult to reconstruct systems with two or three degrees of freedom with a data set of 10,000 points or more. It becomes very time consuming when the data set or the dimension of the system is larger, with the dimension ultimately playing a larger part in the computational effort. Compounding the problems of computational effort is the restrictions that a finite data set place on the analysis. The finite data set means that the choice of reconstruction variable values is no longer arbitrary. This primarily effects the embedding delay and sampling rate. The embedding delay effects the extraction of information from the attractor while the sampling rate determines how much information per point is contained in the data.

This work analyzed the feasibility of reducing the computational effort by reducing the data set size by the choice of more optimal reconstruction parameters. It is possible to greatly reduce the data set size for low dimensional mildly nonlinear systems by careful selection of the data set. However, in most cases it is not possible to carefully select the data set unless there is knowledge of the system already. Since it is desired for the methods to be widely applicable to unknown systems, another method is necessary. The ability of a very small data set, ~100 points, to give good reconstruction however is the event which led to the more important discovery, which is widely applicable.

It was discovered from the geometry of the small set, that if the data is sampled over a longer time period a better geometric reconstruction and prediction was possible. The sample is taken over a longer time period, with the possibility of decreasing the actual number of points. This method allows the system to visit more of the attractor's space and giving a better statistical picture of the attractor.

The delay vector space method is applied to the first state variable of the Lorenz attractor in order to demonstrate the effects of the sampling method. The permutations of the embedding delay and sampling rate are studied to evaluate their impact on the estimation of the fractal dimension of the system which is well known. This provides a strategy for reducing the computational effort when investigating signals from systems which are not well known.

ACKNOWLEDGMENTS

I would like to take the opportunity to thank the following for their patience, understanding, and help while I prepared this thesis:

- My parents. Thank you for putting me through school to this level, making sure that I always had a place to sleep, and food to eat.

Dr. Josef Török, my advisor. Thank you for the chance to work with one of the best and brightest that I have had the good fortune to know. Thank you for all of the help and especially the patience as another project distracted me.

- Ed Ziegler, a fellow student. Thank you for spending the time to teach me the rudiments of programming in the MATLAB environment, and for all his help in exploring many different ideas related to this thesis.

Mr. David Hathaway, my boss at times, and always a good friend. Thanks for always going out of your way to help every student.

CHAPTER 1

DYNAMICAL SYSTEMS

The following material is meant to be a brief review of dynamic systems. The systems are introduced in an order which shows the increasing complexity of systems which leads eventually to stochastic systems. Chaos is defined[i] as:

1. *confusion, or confused mass, of formless matter and infinite space, supposed to have existed before the ordered universe.*
2. *any mixed mass, without due form or order; confusion.*
3. *an empty, immeasurable space; an abyss.*

If these definitions were used in the description of chaotic dynamics it would seem foolish to have pursued this work. There is a much more complex relationship which needs to be understood. Chaos is a subset of a much larger area of study called nonlinear dynamics.

It has been discovered[ii] by many investigators that very simple deterministic systems with a few elements can generate apparently random behavior. The randomness in their behavior is a fundamental property. It was once thought if enough data could be collected and processed that any system was predictable.

---

[i] Webster's New Twentieth Century Dictionary - Unabridged, 2nd Edition (1960)
[ii] Crutchfield, J.P., et al, *Chaos*, Scientific American, December 1986, 46

The randomness associated with these deterministic systems is called chaos. The bulk of this work is to develop analysis methods for nonlinear studies which may in the future be extended to chaotic dynamics.

Periodic systems

Periodic systems are characterized by simple repeated motions at regular intervals. This regular interval is defined as the period, *T*. From the definition of the period follows the relationship:

$$x(t) = x(t+T) \tag{1}$$

It then becomes very simple to predict any future state of *x* by using equation (1). The periodic system is the simplest of dynamic systems.

Quasi-periodic systems

Quasi-periodic systems may appear much more complex than periodic systems. Although more complex, the quasi-periodic system may be represented as the linear combination of periodic modes. The mathematical combination is the finite summation of periodic modes:

$$x(t) = \sum_{m=1}^{n} y_m(t) \tag{2}$$

A system becomes increasingly complex as the number of excited modes, *n*, needed to represent the system increases. Once all individual

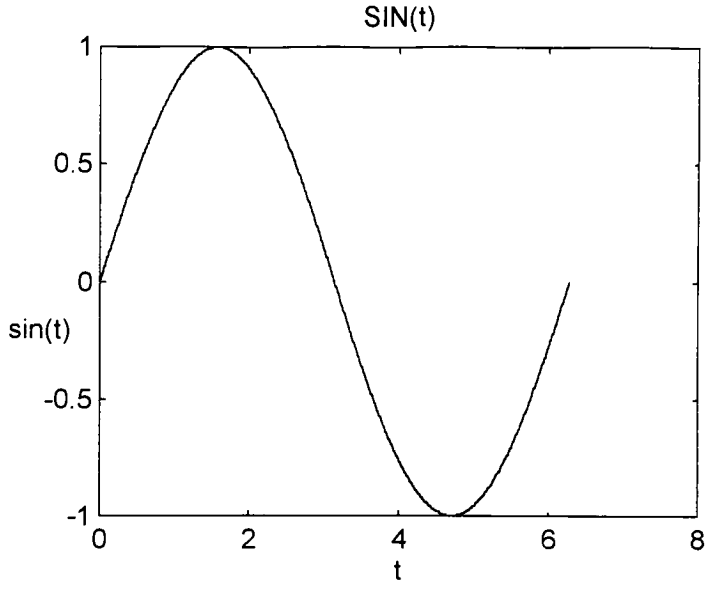


Figure 1. Sin(t) with period T=2π

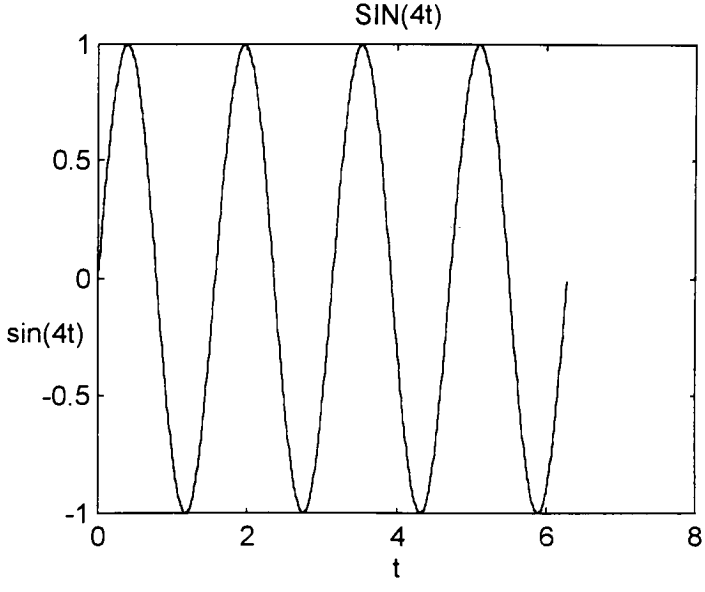


Figure 2. Sin(4t)

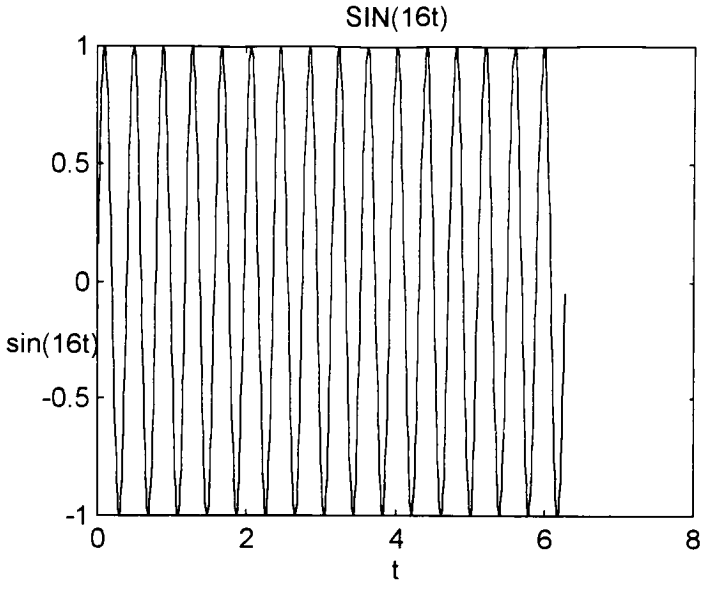


Figure 3. Sin(16t)

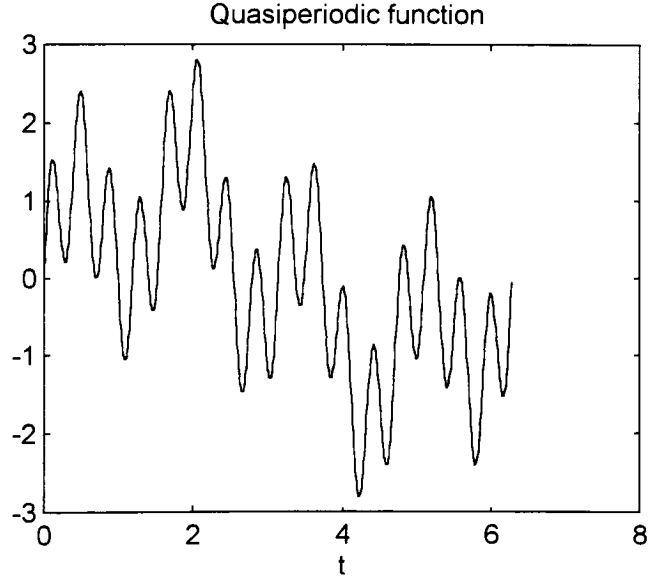


Figure 4. Quasi-periodic system, linear combination of periodic modes.

modes, $y_m(t)$, have been determined, the summation, $x(t)$, may be evaluated at any t. While quasi-periodic systems are more complex than periodic systems, the determination of predictive equations remains elementary.

For an example $y_1(t)$ is sin(t) as shown in Figure 1, $y_2(t)$ is sin(4t) as shown in Figure 2, and $y_3(t)$ is sin(16t) as shown in Figure 3. The summation of the three periodic modes results in a much more complex system, shown in Figure 4.

The summation of the three periodic modes yields a simple quasi-periodic system. As the number of modes increases, the complexity of the system increases until it is not obvious by inspection that the system is made up of multiple modes. The system represented in Figure 4 is fairly simple, yet when a small piece is shown or if there were more exited modes, the periodic basis is not evident.

Nonlinear systems

Nonlinear systems cannot be represented by the linear combination of a finite number of periodic modes. Instead the equations governing evolution of the system contain nonlinear elements or are nonlinear combinations of the state variables. If all the state variables are known and measurable, it is a simple process to determine the system of equations that govern system evolution.

A more troublesome and customary case occurs when all state variables can not either be identified or measured. Mathematicians first hypothesized that the measurement of a single state variable in a nonlinear system is sufficient to reconstruct the dynamics of the system. As a consequence of the system's nonlinearity, every state variable carries all the information necessary to reconstruct the geometry. Since all the necessary information is present in each variable, and the computational tools have been developed to extract it, a set of governing equations can be obtained with some work. The set of equations contains a single equation which predicts the evolution of the state variable that was originally sampled. Since in theory each state variable contains the information, it is not important which one is used. In practice there are small (but significant) variations in the reconstruction from one variable to another.

These small variations become significant when the character of non-linear systems is considered. The trajectories within non-linear systems are highly dependent on initial conditions. Consider then that any instantaneous position, based on the reconstruction, is the initial condition for the rest of the evolution. Slight variations in the position at any given instance result in trajectories which will differ greatly as the system evolves giving rise to considerably different results. This sensitivity to initial conditions provides one of the greatest challenges in the implementation of reconstruction

techniques, the small errors associated with numerical differentiation and integration give rise to large errors in the prediction.

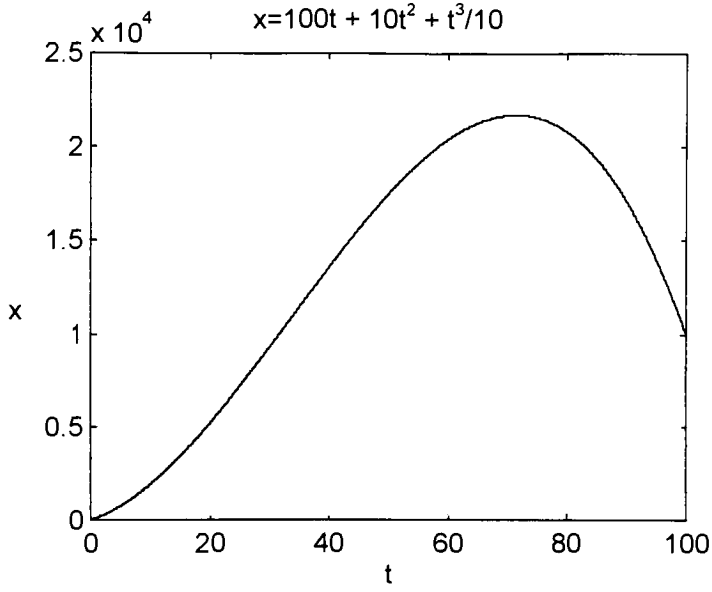


Figure 5. Non-linear function (cubic).

Stochastic systems

The trajectories in stochastic systems are governed primarily by external random noise. The intrinsic properties of the system are either insignificant or indiscernible from the stochastic signal's influence. Analytical methods are not applicable to stochastic systems since the collection of additional data does not contribute additional information.

It is important to test data that appears random for underlying periodic, quasi-periodic, and nonlinear systems. One must be careful not to dismiss a system as stochastic if there is structure within the signal. Often It is possible

to remove stochastic noise from a deterministic signal using standard signal processing techniques.

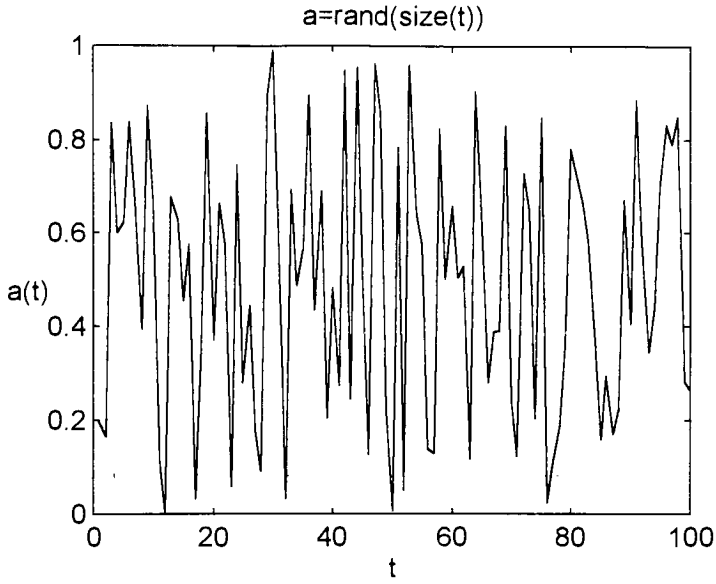


Figure 6. Random function generated using MATLAB's RAND function.

CHAPTER 2

PHASE SPACE

The phase space is the geometrical domain required to describe the instantaneous state of a system. Each orthogonal coordinate of the phase space corresponds to a system variable. Orthoganality is not a requirement, but it eliminates coordinate dependence. Normally the phase space consists of familiar quantities such as position, velocity, and higher order derivatives for each dimension in the occupied space. To fully describe three dimensional motion, a minimum of six coordinates is necessary $x(t)$, dx/dt, $y(t)$, dy/dt, $z(t)$ and dz/dt. Usually a phase plot uses only two coordinates hence the designation of phase planes.

Construction of phase portrait

The phase portrait is constructed in an artificial phase space for use in system identification. The geometry exhibited by the system may be compared to other systems allowing identification of similar characteristics. This may be done without *a priori* knowledge of the phase space in which the data originated. The artificial phase portrait may or may not look like the

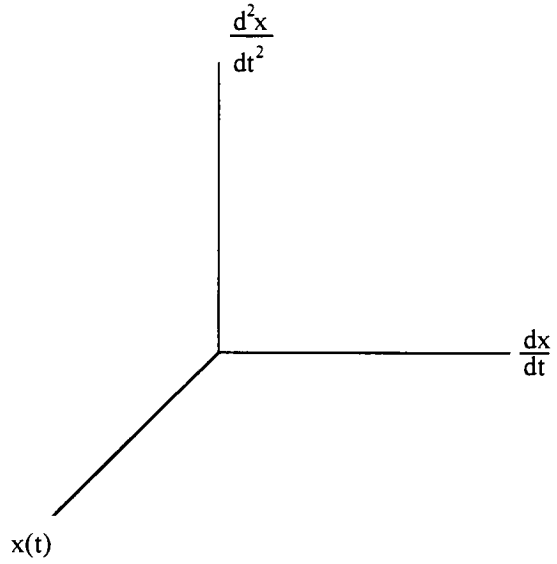


Figure 7. A 3-D normal phase space created using derivatives for coordinates.

actual phase portrait for the system. Lack of similarity is due to the nonlinear combinations of phases inherent in the delay coordinates of the artificial phase space.

Phase trajectory

The path taken on a phase plane is the phase trajectory. Trajectories that correspond to similar energies pass closely but do not cross. If the trajectories crossed at time t the system would be rendered indeterminate by the ambiguity of past and future states. Figure 8 shows the phase trajectory from a simple harmonic oscillator, the position x(t) corresponds to a sin function, while dx/dt corresponds to the cosine derivative. The trajectory forms a unit circle. Figure 9 shows the same system, but with three different

initial conditions. The trajectories do not cross since they correspond to similar energies.

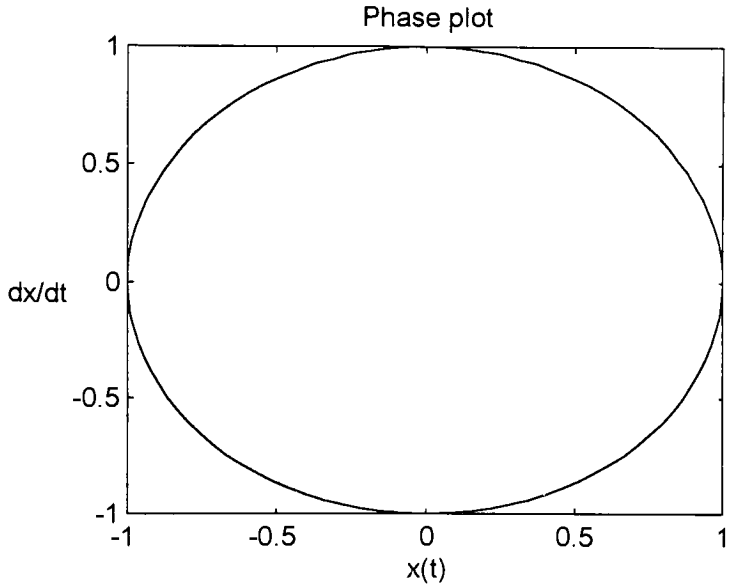


Figure 8. Phase plot generated from Figure 1.

Phase areas

Phase areas are bounded by a set of points in a given area of the phase space. For conservative systems, the trajectories are area preserving. The area preserving quality means that the area of the phase space bounded by a fixed set of points is constant for all times past and future. Dissipative systems have convergent trajectories that decrease the phase area as time progresses. Figures 10 and 11 are examples of conservative and dissapative systems respectively. The energy in the dissapative system tends to zero with time as does the area.

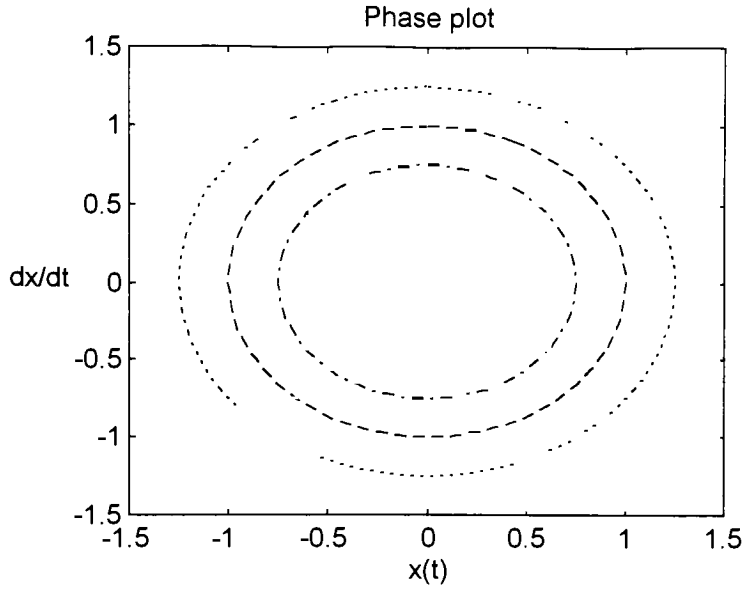


Figure 9. Phase plot showing different initial energies.

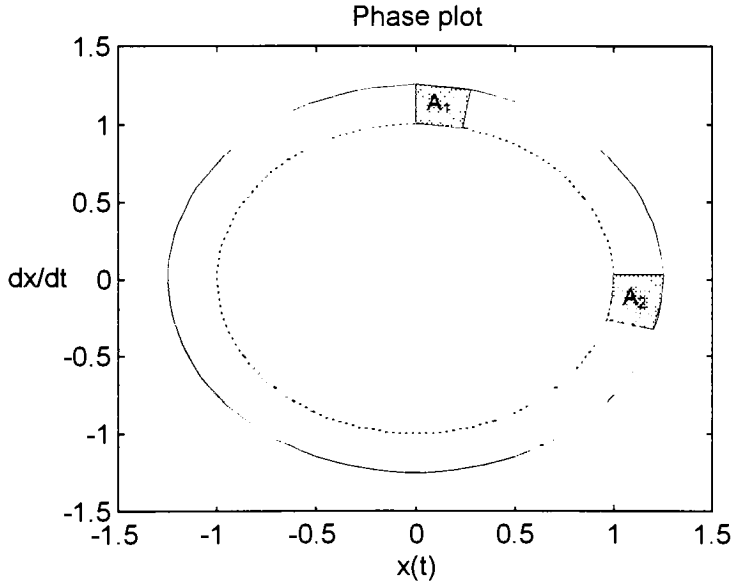


Figure 10. Conservative system, area preserving such that A1=A2.

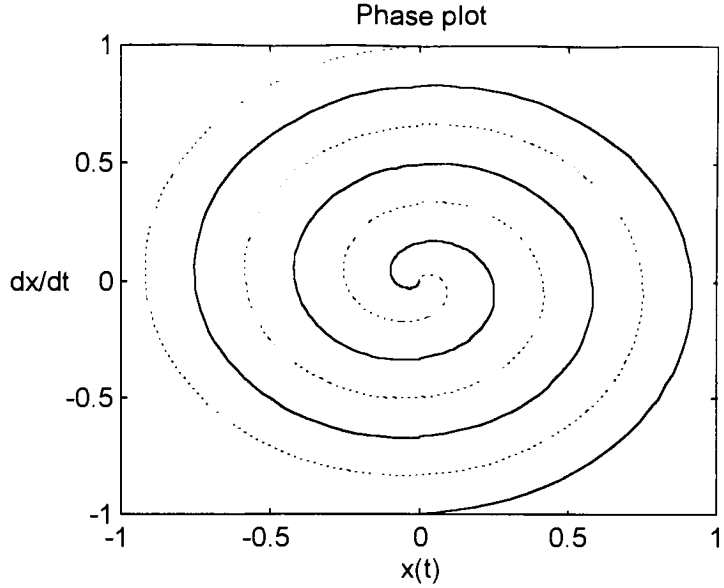


Figure 11. Non-conservative (dissapative) system, area diminishes to zero.

Artificial phase space

A phase space is considered artificial when the coordinates of the space do not have the usual temporal significance. For the purposes of embedding sampled data, an artificial phase space is generated by creating phase coordinates based on a set of time delays τ_n yielding:

$$y(n) = \left[x(t), x(t+\tau_1), x(t+\tau_2), \ldots, x\left(t+\tau_{(d-1)}\right)\right] \tag{3}$$

Although each time delay may be at least in principle picked arbitrarily[iii,iv] it is easier to pick a single value of τ and use multiples of τ for the embedding coordinates. This yields artificial phase space vectors of the form:

[iii] Takens, F., *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics, 898, 366, (1981)

$$y(n) = \left[x(t), x(t+\tau), x(t+2\tau), \ldots, x\left(t+(d-1)\tau\right)\right] \quad (4)$$

This substitution for the usual phase coordinates is acceptable[v] since the time lagged coordinate elements of *y(n)* are nonlinear combinations of the local time derivatives.

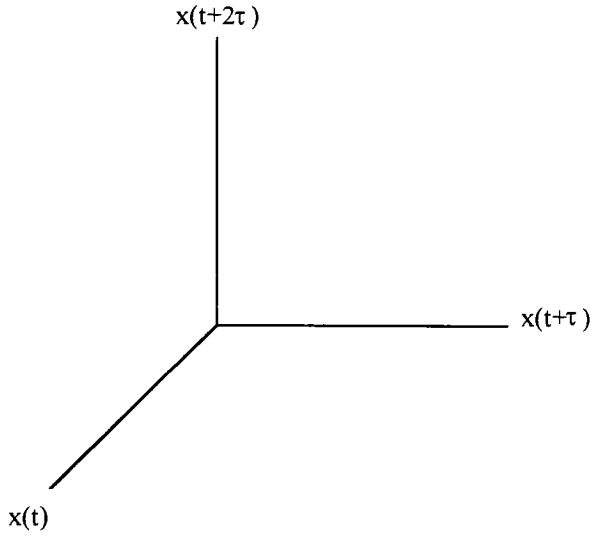


Figure 12. A 3-D artificial phase space created using delay parameters for the coordinates.

[iv] R. Mañé, *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics, 898, 230 (1981)

[v] Eckmann J.-P., Ruelle, D., Ergodic theory of chaos and strange attractors, Reviews of Modern Physics, 57, 617 (1985)

CHAPTER 3

ATTRACTORS

An attractor is a phase space entity that attracts phase trajectories lying within its basin of attraction. It is a set or sub-space on which analytic or experimental data accumulates. The phase trajectory asymptotically approaches the attractor as $t \to \infty$. Transients are the result of an initial condition that is within the basin of attraction, but that is not on the attractor. The attractor may be a map, a set of discrete points, or a flow, a continuous sub-space region.

Phase space maps

Maps are the result of data from discrete systems. In discrete systems each future set of coordinates is mapped directly from the preceding set. The system exists only at discrete states, the set of which is predetermined by the initial conditions. The Henon attractor is an example of a discrete mapping. The Henon attractor is generated by the following set of iterative relations:

$$\left\{\begin{array}{l} x_1(n+1) = x_2(n) + 1 - 1.4x_1(n)^2 \\ x_2(n+1) = 0.3x_1(n) \end{array}\right\} \quad (5)$$

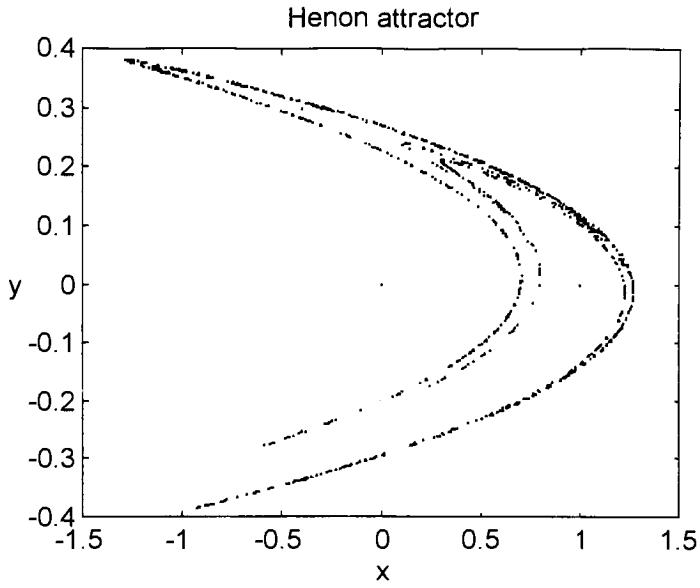


Figure 13. Henon attractor, an example of a mapping. Two initial transient points may be seen at [0,0] and ~[1,0].

Initial values are selected for $x_i(n)$, after which any number of points may be calculated. The choice of initial conditions depends on the interest in the system. If the only interest is in the geometry of the system the initial conditions are unimportant as long as they are within its basin of attraction. The trajectory will approach the attractor given enough iterations for transients to disappear. If a future or exact state of the system is of interest, then the current state of the system must be known.

Phase space flows

Flows are the result of continuous systems. In continuous systems, the trajectory is the result of a continuously differentiable system. The state of

the system can be determined for any time and set of initial conditions. The Rossler attractor is an example of a flow mapping. The Rossler attractor is constructed using the following set of differential equations:

$$\left.\begin{cases} \dot{x} = -(y+z) \\ \dot{y} = x + ay \\ \dot{z} = b + xz + cz \end{cases}\right\} \tag{6}$$

Typically for the Rossler attractor $a=0.2$, $b=0.2$, and $c=-5.7$, although these are not the only possible values. As with the phase space maps, if the only interest is the geometry of the system, the choice of initial conditions is unimportant assuming that they lie within the basin of attraction. If a future state of the system at a given time is of interest, then the initial state becomes important.

Normal attractors

Normal attractors are not sensitive to initial conditions. Small changes in the initial conditions have little effect on the trajectory as $t \rightarrow \infty$.

Point attractor

The trajectory of the point attractor is invariant with time after the transients in the system have vanished. The choice of initial conditions must place the trajectory within the attractors basin of attraction. The basin of attraction is the set of initial states that will converge to the attractor. A good example of a point attractor is a bowl. If a marble is

placed in anywhere within the bowl it will come to rest in the center every time. The center is the point attractor and the volume of the bowl is the basin of attraction. This is shown in Figure 11 of Chapter 2.

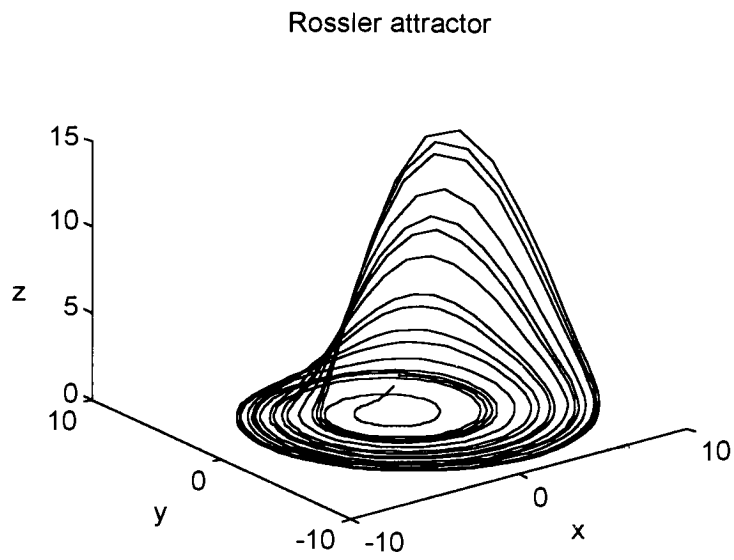


Figure 14. Rossler attractor, an example of a flow. Integrated with a time step of 0.1 seconds from an initial condition of [1 1 1].

Periodic (circle)

The trajectory of the periodic attractor forms a circle. After the transients have vanished the state of the system continues to repeat with period T such that $\mathbf{x}(t)=\mathbf{x}(t+T)$. The trajectories from initial conditions within the circle spiral outward, asymptotically approaching the circle. The trajectories from initial conditions which lie outside the circle spiral inward approaching the attractor with time.

Other Attractors

As with dynamical systems there are corresponding increasingly complex attractors. The torus is the attractor analogous to the quasi-periodic system. It has greater complexity than the circle while maintaining linearity. The complexity of quasi-periodic attractors increases until nonlinear elements are present.

Chaotic (Strange) attractors

The trajectories of strange attractors are sensitive to variation or uncertainty in the initial conditions. Variation in the initial conditions should be thought of as a different initial state, whereas uncertainty could be uncertainty in measurement or computational error. Trajectories within the strange attractor are exponentially divergent, that is points which are very close initially will have greatly different evolutions. Therefore the uncertainty associated with determining the state of a system has direct bearing on the validity of prediction of future states

CHAPTER 4

EMBEDDING DELAY CALCULATION

The first efforts in the analysis involved estimating either the embedding delay or the embedding dimension. Early methods were prone to errors from noisy data or an improper choice of time delay. Subsequent methods have offered some promise in correcting these shortfalls, but no method has been shown to work well for every system.

Autocorrelation function (ACF)

The first attempt to determine an optimal embedding delay used the autocorrelation function. Although the autocorrelation function is suitable for large and small data sets, the embedding delay τ is usually selected using an autocorrelation function if N_{dat} is less than 10,000. The application to smaller data sets is possible because the autocorrelation process picks a smaller τ than would mutual information theory. The autocorrelation is performed to determine the first local minimum or decorrelation time. The decorrelation time is the time at which the autocorrelation function reaches its first zero

value. The embedding delay is then selected such that τ is 1/10 to 1/20 of the first local minimum[vi] or zero. This is an arbitrary choice that may work well in practice. The sampling rate should correspond to approximately the same value as τ. For dynamical time series resulting from mappings, such as the Henon set, the first local minimum of the autocorrelation occurs at the first point. With no other intermediate points available, the first point is selected. The first zero or local minimum of the autocorrelation was thought to ensure linearly independent coordinates. In application to higher order systems the ACF can yield incorrect results[vii] due to correlations between other coordinates and may lead to a collapse of the attractor in one or more coordinates.

Mutual information theory

Mutual information, I, theory is usually applied to larger data sets. The mutual information theory approach looks at the general dependence of two variables, overcoming the autocorrelation's limitation to measuring linear dependence. For the creation of phase portraits, the first local minimum of I is used. The mutual information theory generally picks τ significantly larger than autocorrelation would and is therefore not suitable for use with small

---

[vi] Abarbanel, H.D., Brown, R., Kadtke, J.B., *Prediction in chaotic nonlinear systems: Methods for time series with broadband Fourier spectra*, Physical Review A, 41, 1782 (1990)

[vii] Buzug, Th., Pfister, G., *Optimal delay time and embedding dimension for delay-time coordinates by analysis of the global static and local dynamical behavior of strange attractors*, Physical Review A, 45, 7073 (1992)

data sets. Fraser and Swinney applied mutual information theory[viii] to the Roux and Rossler equations in 1986. The embedding delay τ may be selected using mutual information theory for

$$N_{dat} \geq 10,000. \tag{7}$$

The theory works well for low embedding dimensions but becomes impractical above an embedding dimension of four. In addition to the embedding dimension limitation the algorithm to calculate the mutual information is recursive and not easily coded using MATLAB, so it is not employed in this work.

Other methods

Other methods employ different techniques such as pseudocycle[ix] based time delays and the first minimum logarithm of a generalized correlation integral.[x] All the methods work to a greater or lesser degree, but none supply information about an optimal embedding dimension.

Shortfalls

The largest shortfall with all methods is the failure to pick the best embedding delay. All of the methods in general pick the delay, τ, arbitrarily. In theory, with an infinite supply of noise free data, the choice of delay may in

---

[viii] Fraser, A.M., Swinney, H.L., *Independent coordinates for strange attractors from mutual information*, Physical Review A, 33, 1134 (1986)

[ix] Destexte, A., et al, Physics Letters A, 132, 101 (1988)

[x] Schuster, H.G., Liebert, W., Physics Letters A, 142, 107 (1988)

fact be almost arbitrary. If the delay picked is to short, the attractor lies along the diagonal of the artificial space and its characteristics are difficult to distinguish. This problem is even more evident in low dimensional systems where noise is present. If the delay picked is too long then the points are not strongly correlated due to the sensitivity of the trajectory as explained in Chapter 1. In practice all methods give a delay which may be used to qualitatively reproduce the system. The effects of the delay selection on reconstruction are illustrated below using 2000 points of Lorenz-x data.

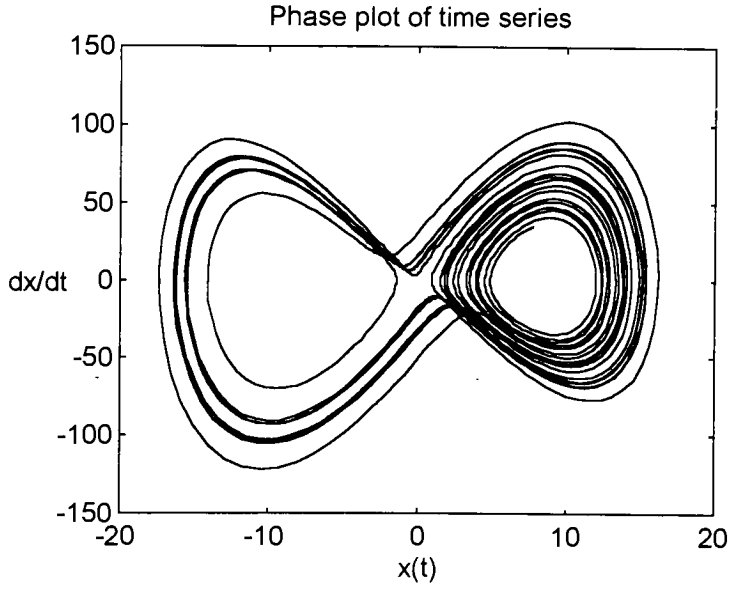


Figure 15. Actual phase plot using centered difference derivative. The embedding in the artificial phase space should look as much like this as possible.

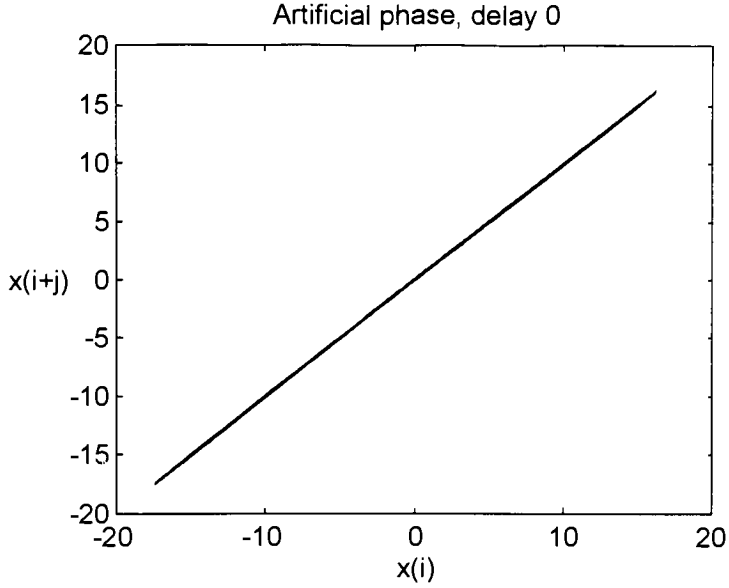


Figure 16. Artificial phase plot with a delay of zero causes the complete collapse onto the hyper-diagonal with the consequent loss of information.

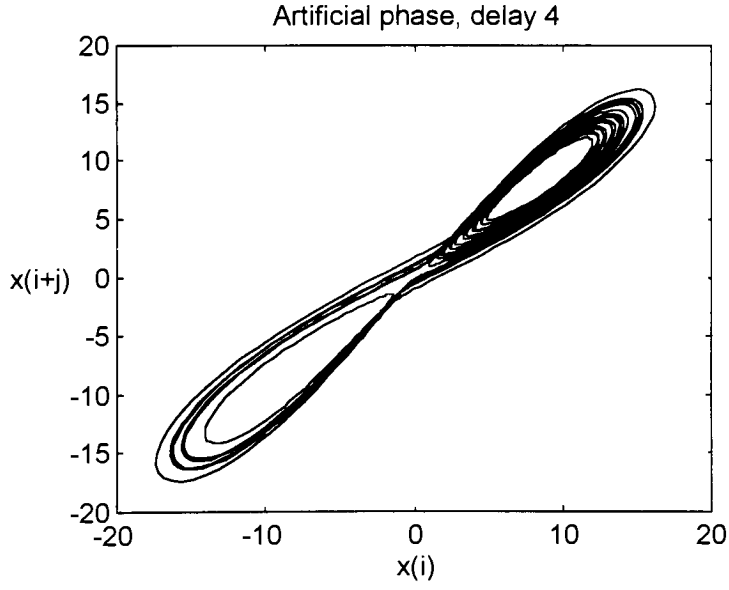


Figure 17. The delay of 4, 1/20th of the decorrelation causes expansion into the artificial phase space and allows information to be extracted. The embedded time series is still largely collapsed upon the hyper-diagonal. There is also a general rotation from the actual phase which will be visible at every delay.

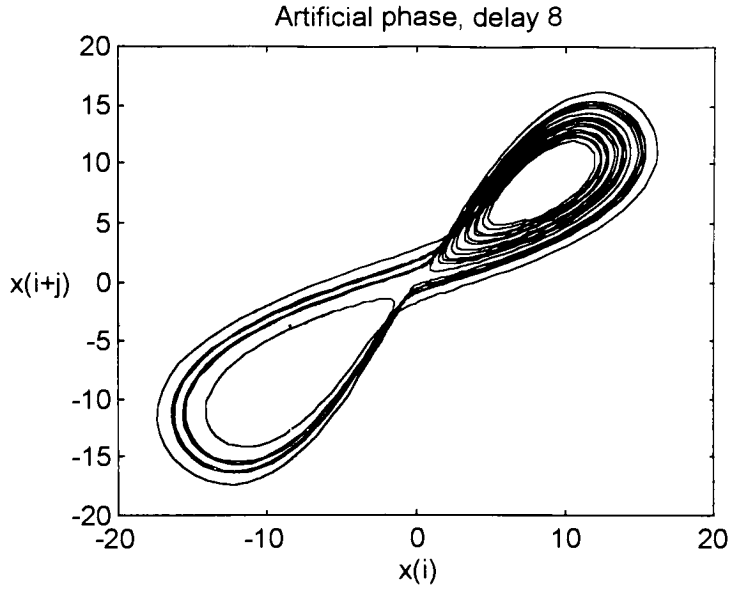


Figure 18. Delay of 8, 1/10th the decorrelation value, gives a reasonable expansion. Notice as the loops expand away from the diagonal, they contract along it.

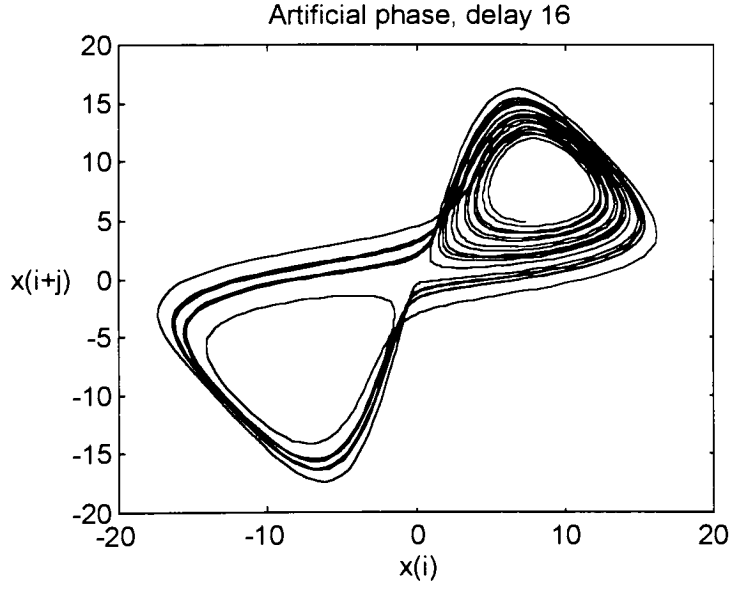


Figure 19. The continued expansion in one direction is associated by a contraction in the other direction at 1/5th of the decorrelation value. It is not possible to tell graphically what the optimum delay is.

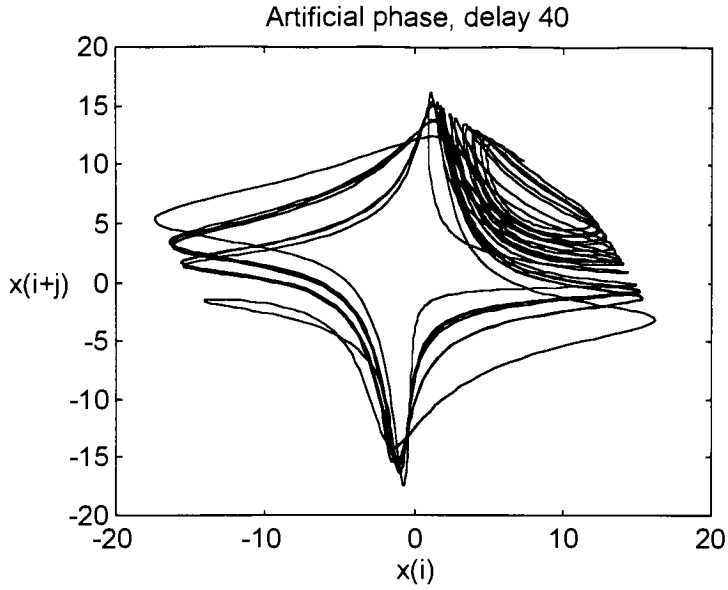


Figure 20. At a delay of 40, one half the decorrelation value, the embedding shows symmetrical folding and information is being lost. The reconstruction has lost most of its geometrical similarity to the phase plot.

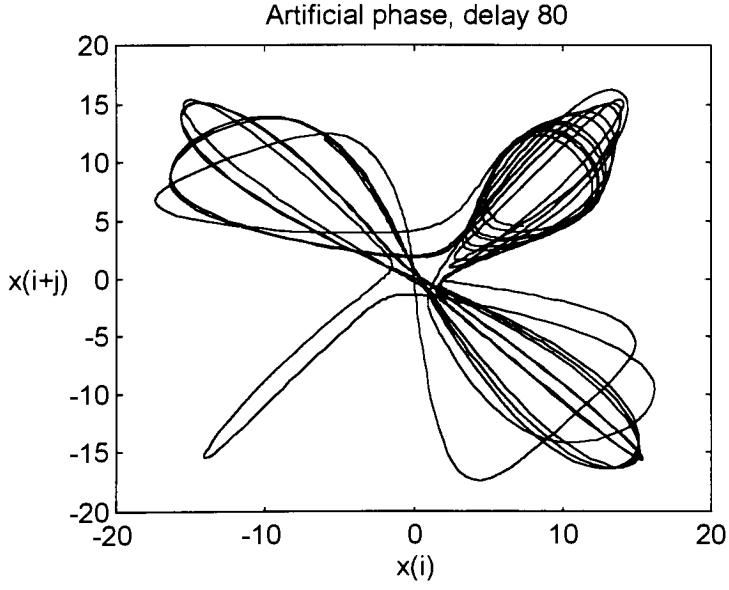


Figure 21. At the decorrelation delay, the folding due to periodicity is so severe that very little information would be gained from analysis. The geometrical character is no longer comparable to the actual phase plot for the time series.

EMBEDDING DIMENSION DETERMINATION

For problems with N large enough, $d_A(N,d)$ will become independent of d when the attractor is properly embedded in R^d. Formal analytical results[xi] dictate the embedding dimension $d_E>2d_A+1$ to ensure that all of the geometric information about the attractor is exposed in the embedding space. Usually in practice however, $d_E>d_A$ is sufficient. Operationally, d_E is increased until d_A becomes a constant. The first value of d_E for which d_A becomes constant then becomes the embedding dimension that is used.

One of the first methods[xii] was suggested by Packard et al. in 1980. The method consisted of embedding data into a phase space using the time series and its successive derivatives for coordinates. The state vectors in the d_E-dimensional embedding space are given by

$$\mathbf{x}(t_s)=\begin{bmatrix} \xi(t_s) \\ \dfrac{d\xi(t_s)}{dt} \\ \\ \cdot \\ \dfrac{d^{d_E-1}\xi(t_s)}{dt^{d_E-1}} \end{bmatrix} \tag{8}$$

---

xi Takens, F., *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics, 898, 366, (1981)

xii Packard, N.H., et al., *Geometry from a time series*, Physical Review Letters, 45, 712 (1980)

The derivative based method worked, but was computationally intensive, very sensitive to noise and not necessarily optimal.

Takens published a very promising theorem in 1980 which is the basis for most analysis today. His method employs time delay coordinates which may be chosen arbitrarily at least in theory. The attractor is reconstructed from a scalar time series $\{\xi(t_k)\}$, where $k \in K_0$ and

$$K_0 = \{k \in \mathrm{N}_0 ; k < N_{\mathrm{dat}}\}. \tag{9}$$

ξ is the observed quantity and N_{dat} is the number of data points. The state vectors for a d_E-dimensional embedding space are given by

$$\mathbf{x}(t_s) = \begin{bmatrix} \xi(t_s) \\ \xi(t_s + \tau) \\ \cdot \\ \cdot \\ \xi(t_s + \tau(d_E - 1)) \end{bmatrix} \tag{10}$$

where $t_s = sT_a$ and $s \in S_0$ for

$$S_0 = \{s \in \mathrm{N}_0 ; s < N_{\mathrm{dat}} - (\tau / T_a)(d_E - 1)\} \tag{11}$$

with sampling time T_a and delay time τ.

Strict theorem requirements necessitate an infinite number of noise free data points with infinite resolution. The theorem also requires $d_E \geq 2n+1$ (where n is the dimension of the flow in original space) for there to be a topological mapping from the original phase space to the embedding space. In application, an infinite amount of noise free data is not required and $d_E > n$ is usually sufficient. The

method is computationally intensive and does not indicate an optimal embedding dimension.

Broomhead and King used singular value decomposition in 1986.[xiii] Using SVD on the trajectory matrix they were able to obtain the number of non zero singular values corresponding to an optimal embedding dimension.

Several others[xiv,xv] have mentioned methods based on the requirement of *n* initial conditions to uniquely solve a set of *n* ordinary differential equations. The method determines the minimum number of conditions necessary to set up the attractor trajectory.

All of the methods above are capable of embedding the time series into a phase space. Some of the methods yield an optimal dimension for embeddings, but all are computationally intensive.

---

xiii Broomhead, D.S., King, J.P., Physica D, 20, 271 (1986)
xiv Bumeliene, S., et al., Liet Fiz. Rinkinys, 28, 569 (1988)
xv Censys, A., Pyragas, K., Physics Letters A, 129, (1988)

CHAPTER 5

HARDWARE & SOFTWARE

Hardware

The twenty six cases which ran to a normal termination required a total of 123 billion floating point operations (flops) for an average of 4.7 billion flops per case. Two computers were used to perform the analyses, a Gateway 486/25 DX and a Gateway 486/66 DX2. The 486/25 performed calculations at an average rate of ~77 kflops/s while the 486/66 performed the calculations at an average rate of ~189 kflops/s. The combination of computers used over 290 hours of processing time to run the 26 cases.

Software

The first version of MATLAB was written at the University of New Mexico and Stanford University in the late 1970's. MATLAB was intended to be used for courses in matrix theory, linear algebra, and numerical analysis. The developers had been involved with LINPACK and EISPACK, Fortran

subroutine packages for matrix manipulation, but wanted a tool that did not require the user to write programs in Fortran. The programming syntax still has the look and feel of Fortran but does not require any knowledge of Fortran or programming in general.

The program has now returned to its origins with the publication of an inexpensive Student Edition of MATLAB. There are also other versions of MATLAB which are available on a variety of platforms. The software which has been developed here is meant to be run using a Pro-MATLAB version. The Student Edition has all the necessary functions to perform the analysis, but was never meant to handle the large data sets.

Many different programs were written to accomplish this work. Some of the original programs no longer exist or are not necessary, while new programs have been written and others have been adapted. All of the current programs run properly under MATLAB v4.2b for Windows. They have been adapted to utilize some of the graphical abilities of Windows, but by no means do they accomplish this fully. The Windows version is a major improvement over the DOS based Student versions capabilities, although I have not had time to explore all of the new features. Due to changes in the MATLAB software some of the programs may not run on older versions of MATLAB due to the elimination of certain commands. I do not know how the graphical commands will be interpreted on older versions.

Most of the programs are documented to an extent. A brief statement is included to give the basic purpose of each function or program. Within each function or program, in the code, comments have been included specifically where important user configurable parameters are. Additionally, there are description below for each module to provide greater insight.

AC.M

The AC.M function performs an autocorrelation on the sampled data and returns the decorrelation index for use in selecting the embedding delay. The autocorrelation for continuous data looks like the following:

$$AC(\tau) = \frac{1}{T}\int_0^T x(t)\cdot x(t+\tau)dt \quad 0 \le \tau \le T \tag{12}$$

This form however is not very useful for a discrete time series. Instead the following discretized form is used, where τ is an index rather than a time as above:

$$AC(\tau) = \frac{1}{N-\tau}\sum_{i=1}^{N-\tau} x_i \cdot x_{i+\tau} \quad 0 \le \tau \le N \tag{13}$$

AC.M is a shorter, faster, slightly less precise version of AUTOCORR.M. To reduce the computational effort, AC.M only processes the first 1000 points. Experiments with data sets of different sizes showed that the reduction in size to 1000 points did not significantly affect the decorrelation index that the function returns. AC.M uses the Hankel matrix function, HANKEL.M available in MATLAB. For this work AC.M was used, although it

is a simple matter to change MAIN.M so that AUTOCORR.M is called instead since both functions use the same arguments.

```
%       AC.M
%       MATLAB function that performs an autocorrelation on the input vector
%       The index of the first zero or local minimum is returned. AC.M is a shorter
%       version of AUTOCORR.M, using only the first 1000 points to reduce the
%       computational effort. Reduction to 1000 points also allows use of HANKEL.M
%       which speeds calculation.
%
%       function decorr = ac(a)

%       Copyright (c) 1992-1994 William Robertson

        function decorr = ac(a)
%       Take first 1000 points for autocorrelation
        if max(size(a)) > 1000; a=a(1:1000); end
        ac=a*hankel(a);
        x=max(size(a)):-1:1;
        ac=ac./x;
        plot(ac); title('Autocorrelation Plot'); xlabel('Index');

%       Find first zero or local minimum
        N=max(size(ac));
        for i=1:N
                if ac(i) <= 0
                        decorr=i;
                        disp(['Autocorrelation zero at lag ',int2str(i), '.'])
                        return
                elseif (ac(i)-ac(i+1)) <0
                        decorr=i;
                        disp(['Autocorrelation local minimum at lag ',int2str(i), ' '])
                        return
                end
        end
```

ARTSPACE.M

The ARTSPACE.M program was written to create artificial phase spaces with sequential delays. This allows the user to visualize the effect of the

embedding delay on the embedded time series. ARTSPACE.M was used to generate Figures 16-21 in Chapter 4. The program calls NDIFF.M to calculate the first time derivative of the series. This is used to generate a phase plot so that subsequent artificial phase plots using different embedding delays may be compared to the real phase plot.

```
%       ARTSPACE.M
%       MATLAB program that creates artificial phase spaces based on Takens' delay
%       vectors. One phase space is created for each delay below the decorrelation
%       index. This program calls the functions NDIFF.M and AC.M.

%       Copyright (c) 1992-1994 William Robertson
        clc
        clear all
        flops(0);
        tic;
%       load the data from a .mat file, alternatively the data could be calculated
        load lorenzx
        x(:,1)=time;
        x(:,2)=value;
        clear time value;
        N=max(size(x));
        disp(['Read ',int2str(max(size(x))),' points from time series.'])
%       the following loop reduces the number of points to 2000, it is not necessary to do
this
        if N>2000
                disp('Using last 2,000 points of time series.')
                y(1:2000,:)=x(N-1999:N,:);
        end
        disp('Calculating first phase for time series.')
        dt=abs(y(1,1)-y(2,1)); % assumes a uniform time step
        x=y(:,2);
        clear y
        [x,dx]=ndiff(2,4,x,dt);
        N=max(size(x));
        figure; plot(x,dx,'g');   % phase plot from derivative calculation
        title('Phase plot');
        xlabel('x');
        ylabel('dx');

        disp('Calculating autocorrelation function.')
```

```
        [r c]=size(x); if r>c; x=x'; end
        decorr=ac(x);
%       generates all artificial phases from delay 1 to decorrelation delay this may cause
problems if
%       decorr is large, second argument in for loop is increment, increase to reduce
number of plots
        for i=1:1:decorr
                y=x(1:N-i); z=x(1+i:N);
                figure; plot(y,z,'g');
                title(['Artificial phase, delay ',num2str(i)])
                xlabel('x(i)'); ylabel('x(i+j)');
        end
        disp(['Elapsed time ',num2str(toc),' seconds.'])
        disp(['Floating point operations = ', num2str(flops),'.'])
        disp('Normal termination of ARTSPACE')
```

AUTOCORR.M

The AUTOCORR.M function performs an autocorrelation on the input vector and returns the decorrelation index. Unlike AC.M which is based upon it, AUTOCORR.M uses the entire input vector, ultimately yielding a better estimate of the decorrelation index. This causes a higher computational effort due to the larger size. The larger size further requires the use of a less efficient algorithm, increasing the computational effort.

```
%       AUTOCORR.M
%       MATLAB Function that performs an autocorrelation on the input vector
%       The decorrelation index is returned indicating the first local minimum or zero of
%       the autocorrelation function.
%
%       function decorr = autocorr(a)

%       Copyright (c) 1992-1994 William Robertson

        function decorr = autocorr(a)
        disp('Calculating autocorrelation function')
        a=a(:);
        N=max(size(a));
        decorr=[];
```

```
        acorr=[1 1];
        acorr(1)=sum(a.*a)/N;
%       because of the possibility of a very large time series the HANKEL function cannot
be used,
%       it requires N squared elements and storage requirements become prohibitive. The
loop method
%       employed below is however very inefficient in MATLAB, so with a better
estimate, comes the
%       price of greater time
        for j=0:N
                i=1:N-j;
                acorr(2)=sum(a(i).*a(i+j))/(N+1-j);
%       check for first zero
                if acorr(2) <= 0
                        decorr=j;
                        disp(['Autocorrelation zero at lag ',int2str(j)])
                        return
%       check for first local minimum
                elseif (acorr(1)-acorr(2)) <0
                        decorr=j;
                        disp(['Autocorrelation local minimum at lag ',int2str(j)])
                        return
                end
                acorr(1)=acorr(2);
        end
```

CORRINT.M

The CORRINT.M function calculates the correlation integral for use in estimating the limit of the attractor dimension. As the embedding dimension increases the estimate of the attractor dimension asymptotically approaches the fractal dimension of the attractor. The method was first proposed by Grassberger and Procaccia after which it is named. The Grassberger-Procaccia algorithm has the form:

$$D(r,N,d_E)=\frac{2}{N(N-1)}\sum_{i=1}^{N}\sum_{j=1}^{N}\Theta\left(r-\left|y(j)-y(i)\right|\right)\quad i\neq j \tag{14}$$

where $\Theta(x)$ is the Heaviside function. The operation of the Heaviside evaluation is given by the following:

$$\Theta(r-S)=\begin{cases}0 & r<S\\ 1 & r\geq S\end{cases} \tag{15}$$

Theiler modified the algorithm to reduce the nearest neighbor effects on the correlation integral which may cause errors in the estimation of the attractor dimension. A term, W, was introduced, changing the form to:

$$C(W,N,r)=\frac{2}{(N+1-W)(N-W)}\sum_{n=W}^{N-1}\sum_{i=0}^{N-1-n}\Theta\left(r-|x_i-x_{i+n}|\right) \tag{16}$$

The term W is calculated by the following:

$$W=\left(\frac{2}{N}\right)^{\frac{2}{m}}+1 \tag{17}$$

where *N* is the number of points in the time series, and *m* is the embedding dimension. W must be rounded up to the next integer value since it is to be used in the summation as an index.

The function of the correlation integral is to sum all of the inter-point distances, *S*, greater than or equal to *r* for each *r*. CORRINT.M evaluates the correlation integral at each r defined by the parameters *rmin*, *rmax*, and *numbin*. The parameter *rmin* determines the lower scaling region and *rmax* determines the upper scaling region. The parameter *numbin* determines how many points including *rmin* and *rmax* there will be. The choice of *rmax* and *numbin* are not critical. The choice of *rmin* is of great importance. If *rmin* is

set too large part or all of the linear scaling region will be lost. Conversely if *rmin* is set too small, statistical aberrations at the lower points may affect the dimension estimate.

```
%       CORRINT.M
%
%       MATLAB function that calculates correlation integral function for plotting
%       log C(N,r) vs. log r. The correlation integral is used to approximate the
%       dimension of a non-linear or chaotic time series.
%
%       function [r,CNr]=corrint(xn,W)

%       Copyright (c) 1992-1994 William Robertson

        function [r,CNr]=corrint(xn,W)
        if nargin==1; W=0; end;
        [R,C]=size(xn);
        if C>R; xn=xn'; [R,C]=size(xn); end;

%       The following two lines are used to set the scaling region for the interpoint
%       distance calculations. The lower and upper limits of 5% and 60% respectively
%       have worked well, but there is no analytical significance. The expression
%       in parenthesis is a quick (& dirty) way to approximate the hypersphere radius
%       of the embedded time series. It is much faster than calculating the Euclidean or
%       taxicab diameter.

        rmin=0.05*sqrt((max(xn(:,1))^2)*C);
        rmax=0.60*sqrt((max(xn(:,1))^2)*C);
%       the following set the number of evenly spaced radii for the correlation integral
%       based upon rmin and rmax
        numbin=25;
        r=linspace(rmin,rmax,numbin);
        xn2=xn;
        CNr=zeros(size(numbin));
        for i=1:R-1;
                b=xn2(1,:); xn2(1,:)=[]; xn2(R,:)=b;
                d=sqrt(sum(((xn-xn2).*(xn-xn2))'));
                for k=1:numbin; C(k)=sum(d <= r(k)); end
                CNr=CNr+C;
        end
        r=log(r);
        CNr=log(CNr/R^2);
```

EMBED.M

EMBED.M is a MATLAB function that maps the sampled time series into an n-dimensional space. The embedding is performed using a delay, associated with the decorrelation index. The function checks to make sure there are enough points after embedding. If the number of points after embedding would be less than half of the number of points in the original time series, then the function terminates. The embedding is accomplished by indexing a column vector by the delay and inserting it sequentially into columns of the artificial space array.

```
%       EMBED.M
%
%       MATLAB function that maps x vector into n dimensions using delay specified at
%       the function calling. If the vector is too small for the number of dimensions
%       an error is generated and the function terminates.
%
%       function [xn]=embed(x,n,delay)

%       Copyright (c) 1992-1994 William Robertson

        function [xn]=embed(x,n,delay)
%       Calculate number of points for embedding
        [N,c]=size(x);
        if c>N; x=x'; N=c; end
        npts=N-(n-1)*delay;
        if npts < N/2
                fprintf('Insufficient number of points for embedding in %3.0f
dimensions.',n)
                return
        end
%       Embed data into column coordinate vectors
        for j=1:n
                xn=[xn x((j-1)*delay+1:N-(n-j)*delay)];
        end
        disp(['Number of embedded points = ',num2str(npts)])
        return
```

LINREG.M

LINREG.M is a MATLAB function that computes the coefficients for a linear function in the least squares sense on a preselected range of the correlation integral results. The correlation integral, (CORRINT.M), yields log(r) vs. log(CNr) data. The lower region of this data, corresponding to smaller inter-point distances, has a slope which scales with the attractor dimension.

The range selection is made in the line that says *xsub=x(1:6)*. The contents of x are predetermined by CORRINT.M. The coefficients of the fit are calculated without regard for the quality of fit. It is assumed that the user has had some experience with the time series and has set up CORRINT.M properly. If this is not the case then ULINREG.M should be substituted for LINREG.M in MAIN.M.

The least squares formulation for a linear fit starts with the following:

$$S_y = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} a_j \cdot z_{ji} \right) \tag{18}$$

S_y is the sum of the squares of the residuals. This function can be minimized with respect to a_j by taking the appropriate partial derivatives. The most common form of this is usually shown as:

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2}$$
$$a_0 = \frac{\sum y_i}{n} - a_1 \frac{\sum x_i}{n} \tag{19}$$

Or the result of the process can be expressed in matrix form as:

$$\left[[Z]^T[Z]\right]\{A\} = [Z]^T\{Y\} \tag{20}$$

Close inspection of the function's coding reveals the simplicity of calculating the linear least squares using the matrix method.

```
%       LINREG.M
%
%       MATLAB function that computes least squares linear fit to the region of data
%       preselected by the user, usually the first few points of the correlation
%       integral. The slope[a] and intercept[b] are returned to the calling program.
%       Linear equation [y]=a[x]+b. xs and ys are used to draw a line through the
%       fitted data points. Similar to ULINREG.M but not interactive.
%
%       function [xs,ys,a,b]=linreg(x,y)

%       Copyright (c) 1992-1994 William Robertson

        function [xs,ys,a,b]=linreg(x,y)
        x=x(:);
        y=y(:);
        figure
        plot(x,y,'+g')
        title('Correlation integral plot')
        xsub=x(1:6);
        ysub=y(1:6);
        xsub=[xsub,ones(size(xsub))];
        A=xsub'*xsub;
        B=xsub'*ysub;
        if det(A)~=0; a=A\B; end
        b=a(2);  % intercept
        a=a(1);  % slope
        xs=[xsub(1);xsub(max(size(xsub)))];
        ys=a*xs+b;
        clg
        plot(x,y,'+g',xs,ys,'--r')
        title(['Attractor dimension approximated to be ',num2str(a)])
```

LORENZ.M

LORENZ.M calculates the time derivatives which define the Lorenz attractor. The velocities are calculated from position using the following set of coupled differential equations:

$$\left\{\begin{array}{l} \dfrac{dx}{dt} = -ax + ay \\ \dfrac{dy}{dt} = -xz + bx - y \\ \dfrac{dz}{dt} = xy - cz \end{array}\right\} \tag{21}$$

where typically *a=10*, *b=8/3*, and *c=28* although the coefficients may be different.

LORENZ.M is called by an integrator such as RUNGE.M for evaluation. The data generated by LORENZ.M was used for most of the work on this thesis. A total of 300,001 points were calculated using a time step of 0.01 seconds and the initial position [1 1 1]. The Lorenz attractor has a fractal dimension of 2.067.

```
%       LORENZ.M
%
%       This MATLAB program calculates the derivatives of the Lorenz attractor.
%       The derivatives are used by a numerical integration routine such
%       as Runge-Kutta.
%
%       Lorenz attractor set of differential equations
%       dx/dt= -ax + ay
%       dy/dt= -xz + bx - y
%       dz/dt= xy - cz
%
%       [xdot]=lorenz(t,x)  t in input is a dummy argument for integrator

%       Copyright (c) 1992-1994 William Robertson
```

```
function [xdot]=lorenz(t,x)
a=10;
b=8/3;
c=28;
dx= -a*x(1) + a*x(2);
dy= -x(1)*x(3) + c*x(1) - x(2);
dz= x(1)*x(2) - b*x(3);
xdot=[dx dy dz];
```

MAIN.M

The MAIN.M routine provides the basic control structure for the collection of programs and functions that perform the analysis for the time series. The MAIN.M program also has some configurable parameters such as the embedding delay and the z factor which determines the secondary sampling of the time series. A diary file is opened and all screen output is also written to a text file. The text file can be saved as a record of how the data was processed and what the various parameters were set to.

```
%       MAIN.M
%
%       Main control routine for system identification programs using
%       MATLAB.

%       Copyright (c) 1992-1994 William Robertson
%
%       A single time series (t, F(t)) is read from a data file or created
%       using a MATLAB function.
%
%       An autocorrelation is performed on the data to determine
%       the decorrelation time for use in embedding the data.
%
%       The dimension of the system is estimated by embedding the data in
%       higher dimensional spaces until convergance of the correlation
%       function slopes is obtained. (Slope of lnC(r) vs. ln(r) converges
%       with increasing embedding dimension N)
```

```
%
%       The file DATA.MAT was created before the execution of MAIN.M .  In the
%       file DATA.MAT two vectors were created, TIME and VALUE such that indexed
%       pairs represent a state.
%

        clc
        clear all
        flops(0);
        diary session.txt
        tic;
        disp('Time series analysis program by William Robertson.')
        disp([date])
        t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))])
        disp('Loading time series.')
        disp('Data being loaded from DATA.MAT.')
        load data
        dt=time(2)-time(1);
        N=max(size(value));
        disp(['Read ',int2str(N),' points from time series.'])
%       z# selects every zth point
        if N>5000
                z=1;
                disp(['Using 5,000 points of time series, every ',num2str(z),' point(s).'])
                i=1:5000;
                j=i*z;
                x(i)=value(N-j);
        end
        clear value;
        disp('Calculating first phase for time series.')
%       See NDIFF.M for explanation of calling parameters
        [x,dx]=ndiff(2,4,x,dt);
        plot(x,dx,'g')
        title('Phase plot of time series')
        xlabel('x(t)')
        ylabel('dx/dt')
        drawnow

%       An autocorrelation is performed to determine the decorrelation time
%       which is used in determining the delay increment associated with the embedding
%       process.
%
        disp('Calculating autocorrelation function.')
        decorr=ac(x);
        t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))])
```

```
        disp(['Elapsed time ',num2str(toc),' seconds'])
%
%       Embed the time series into successively higher dimensions. The embed-
%       ing stops when the embedding dimension n is greater than 2a+1, where
%       a is an approximation of the attractor dimension.

        m=1;
        a=0;
        while m <= (2*a+1);
                m=m+1;
                disp(' ')
                disp([num2str(m),'-Dimensional embedding.'])
                delay=ceil(decorr/2);
                disp(['Using embedding delay of ',int2str(delay), '.'])
                [xm]=embed(x,m,delay);
%       W is Theiler modification to Grassberger-Procaccia algorithm, set W=1 to
eliminate effect
                W=ceil(decorr*(2/max(size(x)))^(2/m))+1;
                disp(['Using W factor of ',int2str(W), '.'])
                [r,CNr]=corrint(xm,W);
                disp('Finished correlation integral.')
                [rsub,CNrsub,a,b]=linreg(r,CNr);
                disp('Finished LINREG.')
                R=[R r];
                CNR=[CNR CNr];
                rs=[rs rsub];
                CNrs=[CNrs CNrsub];
                A=[A a];
                t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))])
                disp(['Elapsed time ',num2str(toc),' seconds.'])
        end
        disp('Embedding procedure complete.')
%       Show results of embeddings.
        figure; plot(R,CNR,'+g',rs,CNrs,'--r')
        text(.80,.15+.05*(m-1),'De  Da','sc')
        xlabel('ln(r)')
        ylabel('ln(CNr)')
        title('Correlation integral results, attractor dimension')
        for i=2:1:m
        msg=([int2str(i),'  ',num2str(A(i-1))]);
        text(.8,.15+((m-i)*.05),msg,'sc');
        end
        disp(' ')
        disp('Embedding   Attractor')
        disp('Dimension   Dimension')
```

```
for i=2:m
        disp([' ',int2str(i),'         ',num2str(A(i-1))])
end
drawnow
disp(' ')
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))])
disp(['Elapsed time ',num2str(toc),' seconds.'])
disp(['Floating point operations = ', num2str(flops),'.'])
disp('Normal termination of MAIN.M')
diary off
```

NDIFF.M

NDIFF.M is a numerical differentiation function that was originally written to provide real phase plots. It can perform backward, centered, and forward differences on equi-spaced data. Forward and backward differences are performed using 2 or 3 point formulas, while the centered difference can be performed using 2, 4, or 6 points. NDIFF.M returns two vectors, a subset of the original time series and the time derivatives that correspond to the subset. The usage of multiple point differentiation formulas means that not all the derivatives can be evaluated, thereby reducing the useable number of points. The two point backward difference formula is based upon the following finite divided difference:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + Oh \tag{22}$$

The three point backward difference is:

$$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h} + Oh^2 \tag{23}$$

The two point centered difference is based upon:

$$f'(x_i) = \frac{f(x_{i-1}) - f(x_{i-1})}{2h} - Oh^2 \quad (24)$$

The four point centered difference is based upon;

$$f(x_i) = \frac{f(x_{i-2}) - 8f(x_{i-1}) + 8f(x_{i+1}) - f(x_{i-2})}{12h} + Oh^4 \quad (25)$$

The six point centered difference is based upon:

$$f'(x_i) = \frac{-f(x_{i-3}) + 9f(x_{i-2}) - 45f(x_{i-1}) + 45f(x_{i+1}) - 9f(x_{i+2}) + f(x_{i-3})}{60h} + Oh^6 \quad (26)$$

The two point forward difference is based upon:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - Oh \quad (27)$$

The three point forward difference is based upon:

$$f'(x_i) = \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h} - Oh^2 \quad (28)$$

```
%       NDIFF.M
%
%       This program uses backward, centered, or forward divided difference tech-
%       niques to numerically estimate derivatives, dx/dt. The input arguments are:
%
%       type  -one character string giving type of divided difference to perform
%       npts  -number of data points to use in calculating the divided difference
%       x     -vector with x values
%       dt    -dt value
%
%       Examples of valid combinations
%       dx=ndiff(1,2,x,dt)   dx=ndiff(1,3,x,dt)
%       dx=ndiff(2,2,x,dt)   dx=ndiff(2,4,x,dt)   dx=ndiff(2,6,x,dt)
%       dx=ndiff(3,2,x,dt)   dx=ndiff(3,3,x,dt)
%
%       function [xs,dxs]=ndiff(type,npts,x,dt)

%       Copyright (c) 1992-1994 William Robertson
```

```
function [xs,dx]=ndiff(type,npts,x,dt)
np=max(size(x));

if type==1      % backward differenc
        if npts==2
                dx=(x(2:np)-x(1:np-1))/dt;
                xs=x(2:np);
        elseif npts==3
                dx=(x(3:np)-4*x(2:np-1)+3*x(1:np-2))/(2*dt);
                xs=x(3:np);
        end
end

if type==2      % centered difference
        if npts==2
                dx=(x(3:np)-x(1:np-2))/(2*dt);
                xs=x(2:np-1);
        elseif npts==4
                dx=(-x(5:np)+8*x(4:np-1)-8*x(2:np-3)+x(1:np-4))/(12*dt);
                xs=x(3:np-2);
        elseif npts==6
                dx=+x(7:np)-9*x(6:np-1)+45*x(5:np-2)-45*x(3:np-4)+9*x(2:np-
5)-x(1:np-6);
                dx=dx/(60*dt);
                xs=x(4:np-3);
        end
end

if type==3      % forward difference
        if npts==2
                dx=(x(2:np)-x(1:np-1))/dt;
                xs=x(1:np-1);
        elseif npts==3
                dx=(-x(3:np)+4*x(2:np-1)-3*x(1:np-2))/(2*dt);
                xs=x(1:np-2);
        end
end
```

ROSSLER.M

ROSSLER.M is a function much like LORENZ.M. It is called by an integrator for evaluation. The Rossler velocity equations are:

$$\left\{\begin{array}{l}\dfrac{dx}{dt} = -y - z \\ \dfrac{dy}{dt} = x + ay \\ \dfrac{dz}{dt} = b + xz - cz\end{array}\right\} \tag{29}$$

where typically $a=0.2$, $b=0.2$, and $c=4.6$.

```
%       ROSSLER.M
%
%       MATLAB function that calculates the derivatives of the Rossler attractor.
%       The derivatives are used by a numerical integration routine such
%       as RUNGE.M.
%
%       [xdot]=rossler(t,x)

%       Copyright (c) 1992-1994 William Robertson

%       Rossler attractor
%       dx/dt= -(x + z)
%       dy/dt= x + ay
%       dz/dt= b + xz - cz
%
%       [xdot]=rossler(t,x)  t in input is a dummy argument for integrator
        function [xdot]=rossler(t,x)
        a=0.2;
        b=0.2;
        c=4.6;

        dx= -x(2) - x(3);
        dy= x(1) + a*x(2);
        dz= b + x(1).*x(3) - c*x(3);
        xdot=[dx dy dz];
```

RUNGE.M

RUNGE.M is the MATLAB implementation of a classical 4th order Runge Kutta integration routine. A set of parameters, k_i are calculated based on the current function value and the integration step size. It uses a fixed time step

selected by the *dt* argument in the calling function. The next value of the function is calculated by:

$$y_{t+1} = y_t + \left[\frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)\right]dt \tag{30}$$

where the k_i are calculated by:

$$\begin{Bmatrix} k_1 = f\left(t_t, y_t\right) \\ k_2 = f\left(t_t + \frac{dt}{2}, y_t + k_1\frac{dt}{2}\right) \\ k_3 = f\left(t_t + \frac{dt}{2}, y_t + k_2\frac{dt}{2}\right) \\ k_4 = f\left(t_t + dt, y_t + k_3 dt\right) \end{Bmatrix} \tag{31}$$

A fixed time step is used to simplify the analysis. RUNGE.M was used to integrate both the Lorenz and Rossler differential equations. A higher order integrator could be used, but it is probably not worth the additional computational effort. The integration tends to smooth out any noise in the data where as the differentiation function will exaggerate noise and cause further propagation.

```
%     RUNGE.M
%
%     MATLAB function that implements the 4th order Runge-Kutta routine for
%     numerical integration. Performs integration on polynomials passed from
%     a main control program. Returns the integrated functions to the main program.
%
%     F      -function file used to evaluate the function
%     ti     -initial time
%     dt     -integration step size
%     tf     -final time
%     y0     -initial value vector of state variables
%     t      -returned time vector
%     y      -returned solution, one column per state variable
%
%     function [t,y]=runge(F,ti,dt,tf,y0)
```

```
%       Copyright (c) 1992-1994 William Robertson

        function [t,y]=runge(F,ti,dt,tf,y0)
        t=(ti:dt:tf)';
        y=zeros(length(t),length(y0));
        y(1,:)=y0;
        for i=1:1:length(t)-1
                k1=feval(F,t(i),y(i,:));
                k2=feval(F,t(i)+dt/2,y(i,:)+dt*k1/2);
                k3=feval(F,t(i)+dt/2,y(i,:)+dt*k2/2);
                k4=feval(F,t(i)+dt,y(i,:)+dt*k3);
                y(i+1,:)=y(i,:)+dt*(k1+2*k2+2*k3+k4)/6;
        end
```

ULINREG.M

ULINREG.M is a MATLAB function that computes the linear least squares fit to data from the correlation integral. The data from the correlation integral is plotted and the user interactively selects the region of data to curve fit. After the range is selected, the slope is calculated. The figure is re-plotted showing the slope and the best fit line. The user can then accept or reject the selection. If the selection is rejected, the user is prompted to re-select.

```
%       ULINREG.M
%
%       MATLAB function computes least squares linear fit to the region of data
%       selected graphically by the user. The user may accept or reject the selection
%       before proceeding. The slope[a] and intercept[b] are returned to the calling
%       program. xs and ys are subsets represent the best fit line for the selection.
%       Linear equation [y]=a[x]+b.
%
%       function [xs,ys,a,b]=ulinreg(x,y)

%       Copyright (c) 1992-1994 William Robertson

        function [xs,ys,a,b]=ulinreg(x,y)
        x=x(:);
        y=y(:);
```

```
        button=0;
%       Select region of data to curve fit.
        while button~=2
                figure
                plot(x,y,'+g')
                title('Correlation integral plot')
                xlabel('Pick two points in linear region of graph')
                [xloc,yloc,button]=ginput(2);
                index=find(x >= min(xloc) & x <= max(xloc));
                xsub=x(index);
                ysub=y(index);
                xsub=[xsub,ones(size(xsub))];
                A=xsub'*xsub;
                B=xsub'*ysub;
                if det(A)~=0;a=A\B;end
                b=a(2);  % intercept
                a=a(1);  % slope
                xs=[xsub(1);xsub(max(size(xsub)))];
                ys=a*xs+b;
                clg
                plot(x,y,'+g',xs,ys,'--r')
                title(['Attractor dimension approximated to be ',num2str(a)])
                xlabel('Left button to try again, Right button to continue')
                [xloc,yloc,button]=ginput(1);
        end
```

CHAPTER 6

EXPERIMENTAL PROCEDURE

The original plan was to perform the battery of experiments on both Rossler and Lorenz data. This would have doubled the run time of the computers and there were some inherent difficulties in using the Rossler attractor.

Data generation

The Rossler equations were integrated using the function RUNGE.M for 60 seconds with a dt=0.01 seconds from an initial condition of [0 0 0]. The initial integration was to remove the transients caused by an initial condition within the basin of attraction, but not on the attractor. A second integration was performed using a larger time step of 0.05 seconds to provide a working data set of 60,001 points spanning a total of 3000 seconds.

The Lorenz data was similarly generated using RUNGE.M from an initial condition of [1 1 1], with dt=0.01 seconds. Inspection of the equations show a point attractor for the initial condition of [0 0 0]. The Lorenz data also spanned 3000 seconds, but had to be integrated with a smaller time step, and therefore yielded a very large data set of 300,001 points.

To speed up the programs, the data for each attractor was written to a .MAT file. This allowed MATLAB to read the data from the file rather than recreate it for each case. Although the investigations used very large data sets, this method yields significant time savings even for small data sets.

Abandonment of Rossler

The Rossler attractor was only used briefly in this investigation due to its weak non-linearity. The fractal dimension of the Rossler attractor is only 2.027, a result of the weak non-linearity. The correlation integral estimate of the attractor dimension from this analysis should approach the fractal dimension asymptotically from below. For the Rossler attractor this requires that the estimation be 99% of the actual dimension in order to accurately indicate the need for three coordinates to describe the attractor. Most combinations of embedding delay and secondary sampling yielded convergence to a dimension estimate just less than two.

Secondary sampling

To further reduce the effects of any transients, all points used for the analysis were taken from the end of each data set. Seven secondary sampling schemes were used, every point, every second point, every third point, every fifth point, every tenth point, every 25^{th} point, and every 50^{th}

point. Even at every 50^{th} point, this left more than 50,000 points for transients to settle out in the Lorenz data.

Embedding delays

Four different embedding delay fractions were used at each secondary sampling rate. The embedding delay fractions were one, one half, one fifth, and one tenth of either the first local minimum or zero. In two cases this led to non-convergence, and in one case, the duplication of results. In the 28 cases which were run, 26 resulted in normal termination. The duplication caused by the embedding delay selection resulted in there being only 25 unique cases in the study.

Configuration of MAIN.M

For each case, several parameters had to be set. The first parameter set the number of points for the data set that would be used for the analysis. The second parameter was the secondary sampling rate. The secondary sampling rate, Z, affects how the data is sampled from the storage file. The storage file is read from the disk and then the program would take every Z^{th} point from the end of the file until it has the specified number of points. The delay parameter controls the embedding delay selection based upon the decorrelation time as returned from AC.M or AUTOCORR.M to the MAIN.M

program. The delay parameter divided the decorrelation value by 1, 2, 5, or 10 depending on the value it was set to.

RESULTS

The matrix of delay and secondary sampling values yielded 25 unique cases out of a possible 28. The results of the different parameters were wide spread in their affect on the maximum embedding dimension and the maximum estimate of the attractor dimension. The tables below summarize important characteristics of the cases that were investigated.

Case number matrix

	delay/1	delay/2	delay/5	delay/10
z=1	1	2	3	4
z=2	5	6	7	8
z=3	9	10	11	12
z=5	13	14	15	16
z=10	17	18	19	20
z=25	21	22	23	24[†]
z=50	25[‡]	26[§]	27	28

[†] Case 24 yielded the same results as case 23 due to duplication of the delay parameter.
[‡] Case 25 was terminated by the author.
[§] Case 26 was terminated by the author.

The case number matrix shows the two parameters for each case, the delay values across the top and the secondary sampling rates down the first column. This is provided as a reference for comparison of the results presented in the figures from the analysis.

The termination matrix shows the position of the two cases which were terminated by the author. Both cases exceeded an embedding dimension of eight, and did not show any signs of impending convergence.

Highest embedding dimension matrix

	delay/1	delay/2	delay/5	delay/10
z=1	5	5	5	5
z=2	8	6	5	5
z=3	7	6	5	5
z=5	7	6	6	5
z=10	7	6	5	5
z=25	7	5	5	5
z=50	N/A	N/A	7	6

The highest embedding dimension matrix shows the highest dimension that each case was embedded in. Highlighted cells indicate an embedding dimension of six or greater, in order to achieve an attractor dimension estimate greater than two, based on the criteria established earlier. The cases which terminated with

an embedding dimension of five, all have an attractor estimate of less than two which will be shown in the following.

Highest attractor dimension estimate matrix

	delay/1	delay/2	delay/5	delay/10
z=1	1.917	1.915	1.843	1.702
z=2	3.406	2.055**	1.937	1.773
z=3	2.752	2.109	1.983	1.808
z=5	2.955	2.086	2.018	1.847
z=10	2.760	2.038	1.983	1.841
z=25	2.974	1.998	1.950††	1.950
z=50	N/A	N/A	2.918	2.044

The highest attractor dimension estimate matrix shows the maximum estimate for the attractor dimension in each case. For most cases, as would be expected the maximum estimate occurs at the highest embedding dimension. In two instances this was not the case, with the peak estimate occurring at one less than the maximum embedding dimension. In both cases there was a slight decrease in the estimate in the final embedding dimension.

Highlighted cells correspond to values within 5% of the actual attractor dimension. This is meant to include cases which were still approaching from below or which may have slightly over estimated the attractor dimension.

** At De=5, 2.018 at De=6, the final embedding dimension.
†† At De=4, 1.947 at De=5, the final embedding dimension. Also case 24.

Attractor dimension estimate at De=5 matrix

	delay/1	delay/2	delay/5	delay/10
z=1	1.857	1.915	1.843	1.702
z=2	2.886	2.055	1.937	1.773
z=3	2.502	2.062	1.983	1.808
z=5	2.518	2.066	2.008	1.847
z=10	2.399	2.025	1.983	1.841
z=25	2.528	1.998	1.947	1.947
z=50	N/A	N/A	2.531	2.033

The table above shows the estimate of the attractor dimension at an embedding dimension of five. This allows inspection of the progress of the estimate at one dimension less than the minimum. This time the shaded cells correspond to estimates of the attractor dimension which are within zero to five percent below the attractor dimension. The highlighted cells are the same in both tables. For the highlighted cases, very little information is gained by the successive embeddings. This result validates the stopping criteria used for the embedding dimension.

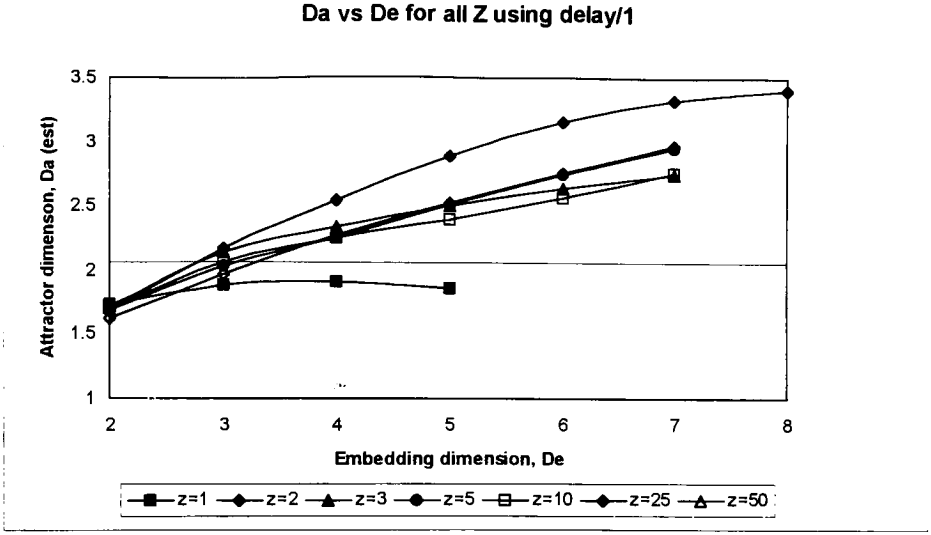


Figure 22. Plot of the attractor dimension estimate as a function of the embedding dimension and the secondary sampling rate for the embedding delay equal to the decorrelation time. Only z=1 and z=2 show convergent behavior, neither to the correct value. The choice of delay is poor.

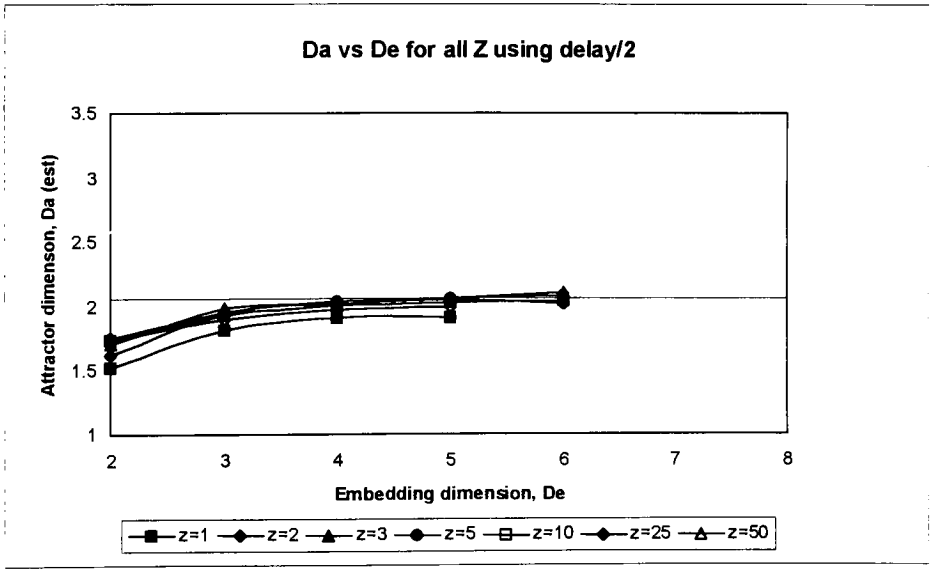


Figure 23. Plot of the attractor dimension estimate as a function of the embedding dimension and the secondary sampling rate for the embedding delay equal to one half the decorrelation time. All Z's converge, but Z=1 underestimates the attractor dimension. The delay is good, and the choice of secondary sampling rate is virtually unimportant.

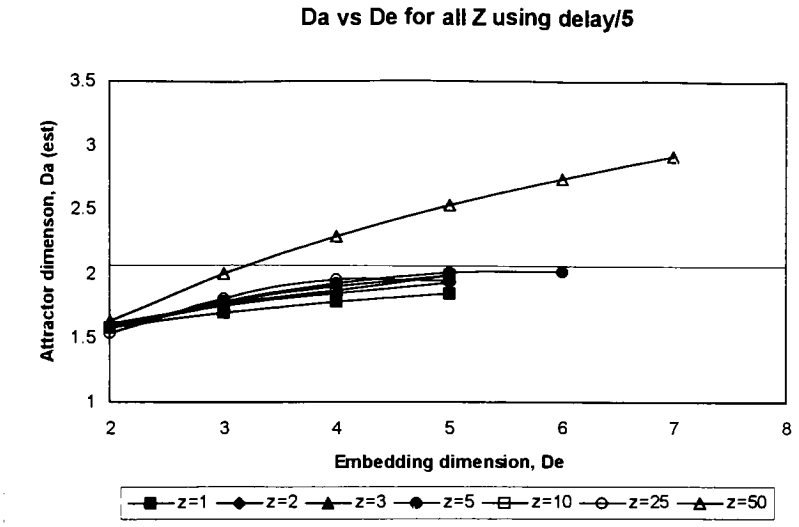


Figure 24. Plot of the attractor dimension estimate as a function of the embedding dimension and the secondary sampling rate for the embedding delay equal to one fifth the decorrelation time. For Z<50 the estimates all still are approaching the correct value albeit slowly. Again the delay is probably acceptable, and the choice of secondary sampling is almost unimportant.

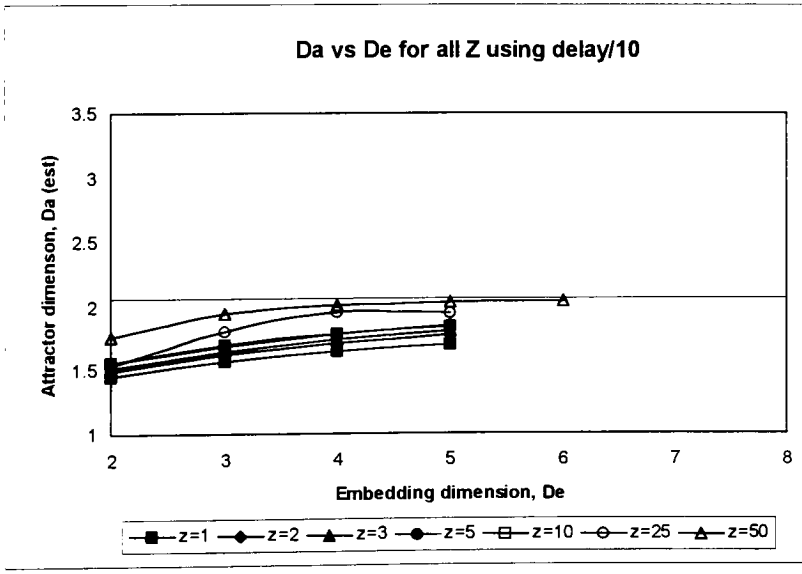


Figure 25. Plot of the attractor dimension estimate as a function of the embedding dimension and the secondary sampling rate for the embedding delay equal to one tenth the decorrelation time. Only Z=50 approaches the correct value. The choice of delay is poor, almost no choice of secondary sampling rate estimates the attractor dimension

ANALYSIS OUTPUT

Each figure in the following section corresponds to a single case. They allow the visual comparison of each embedding dimension to the others for the selection of secondary sampling and delay parameters. The caption below each figure gives the pertinent details of the analysis that generated the figure. In addition Appendix A contains the full output from each analysis. The caption also references the appropriate pages of the Appendix should further information be required.

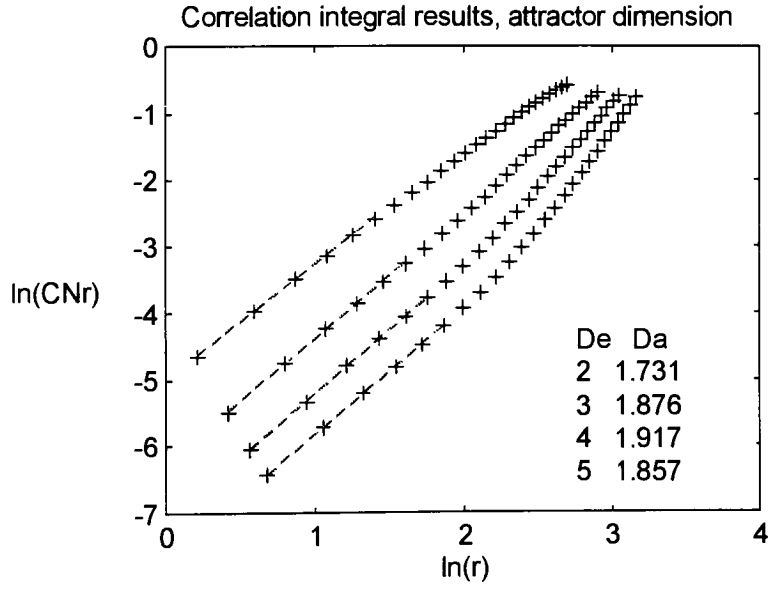


Figure 26. Case 1. Last 5000 points, every point, embedding delay equal to first local minimum. Appendix pages A-1 to A-3.

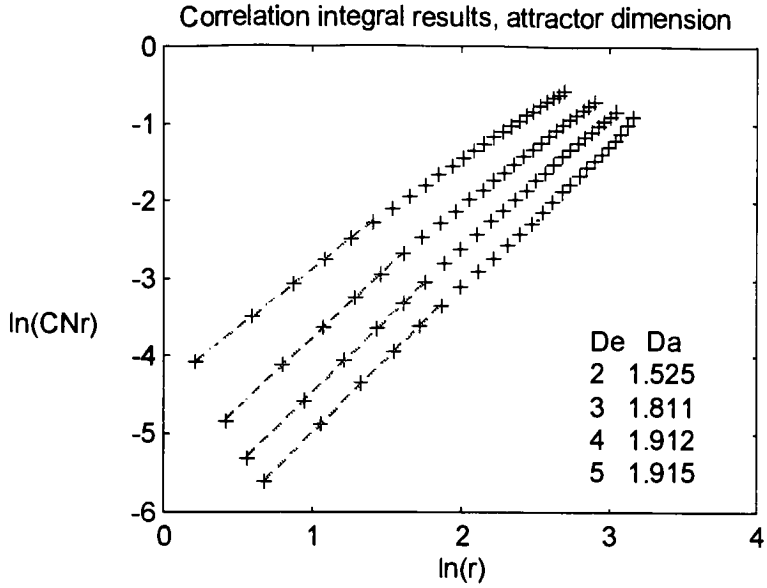


Figure 27. Case 2. Using last 5000 points, every point, embedding delay at one half of first local minimum. Appendix pages A-4 to A-6.

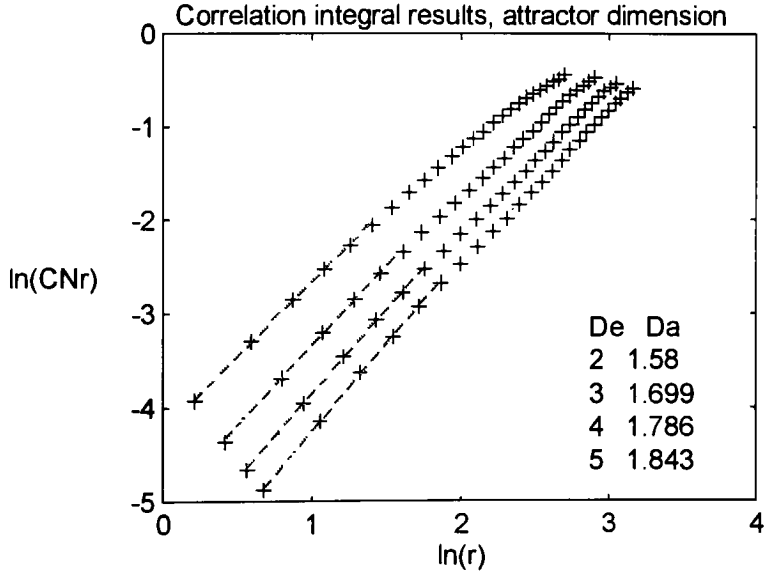


Figure 28. Case 3. Using last 5000 points, every point, embedding delay at one fifth of first local minimum. Appendix pages A-7 to A-9.

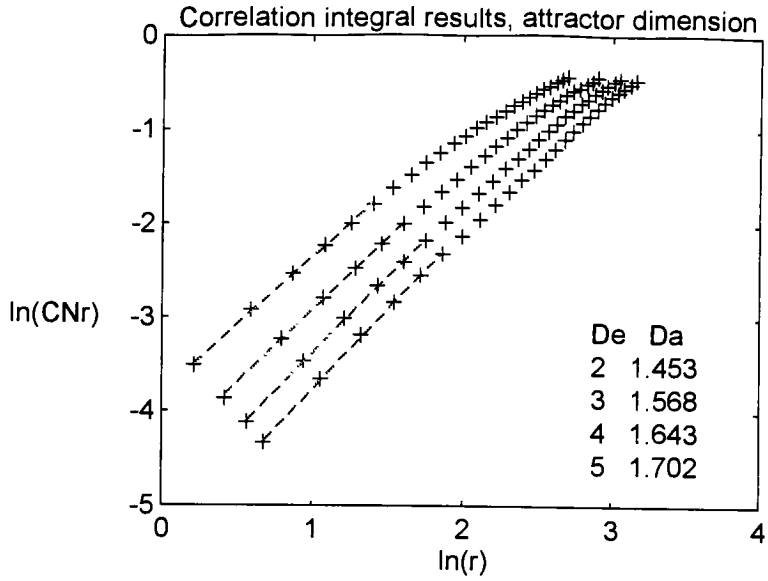


Figure 29. Case 4. Using last 5000 points, every point, embedding delay at one tenth of first local minimum. Appendix pages A-10 to A-12.

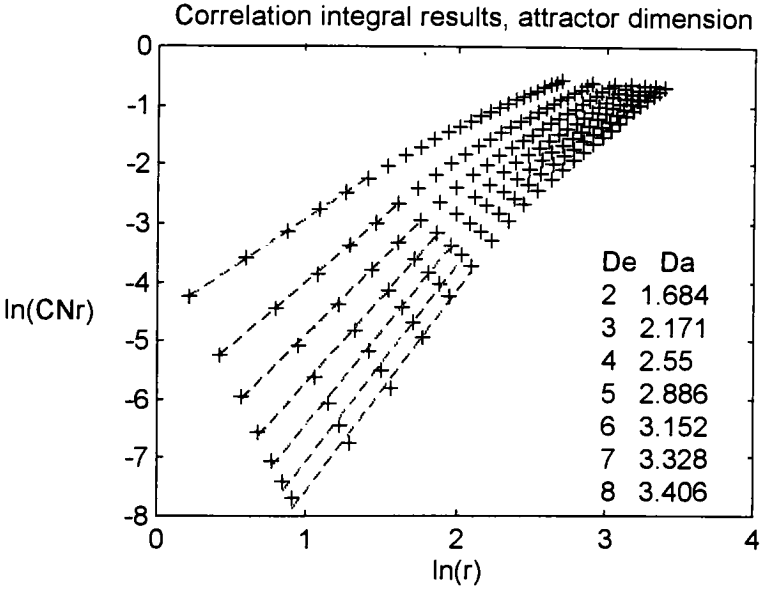


Figure 30. Case 5. Using 5000 points, every other point, embedding delay of first zero. Appendix pages A-13 to A-17.

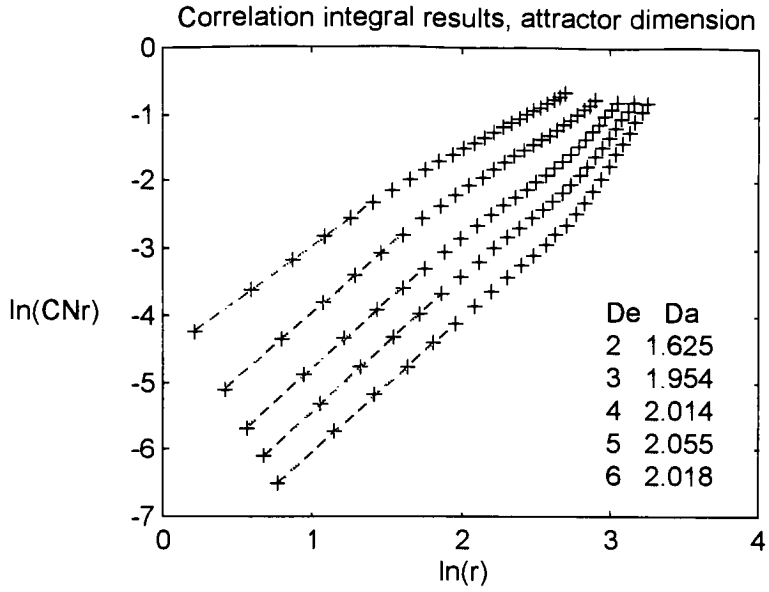


Figure 31. Case 6. Using 5000 points, every other point, embedding delay of one half of the first zero. Appendix pages A-18 to A-21.

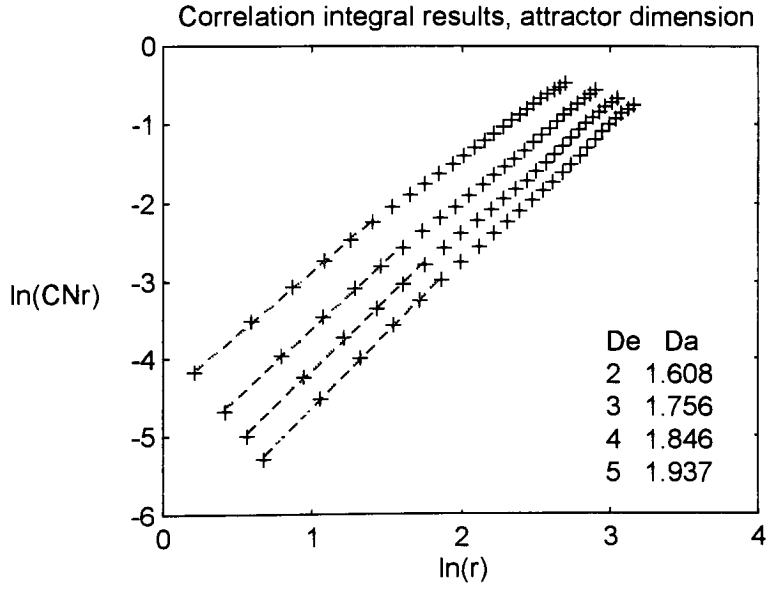


Figure 32. Case 7. Using 5000 points, every other point, embedding delay of one fifth of the first zero. Appendix pages A-22 to A-24.

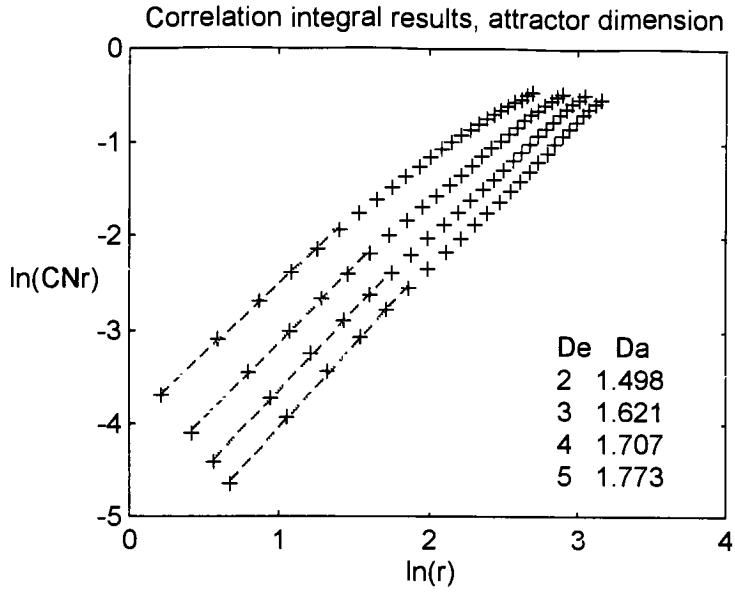


Figure 33. Case 8. Using 5000 points, every other point, embedding delay of one tenth of the first zero. Appendix pages A-25 to A-27.

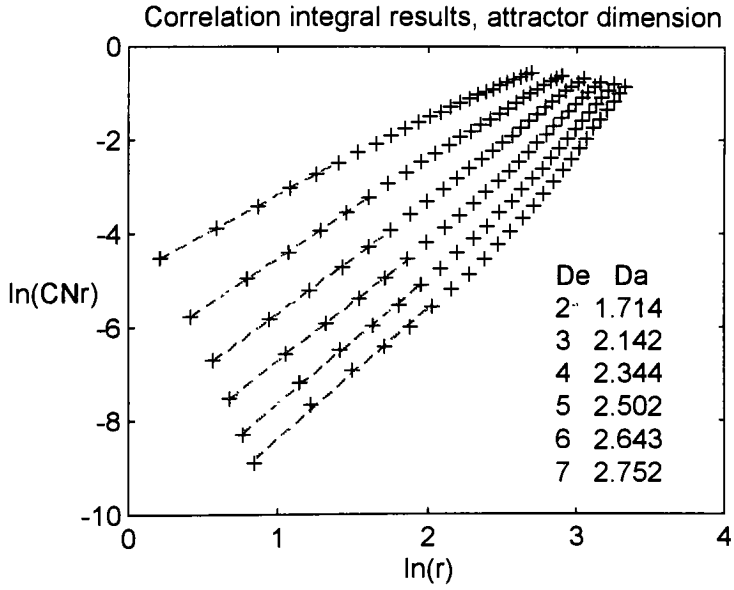


Figure 34. Case 9. Using 5000 points, every third point, embedding delay of the first local minimum. Appendix pages A-28 to A-32.

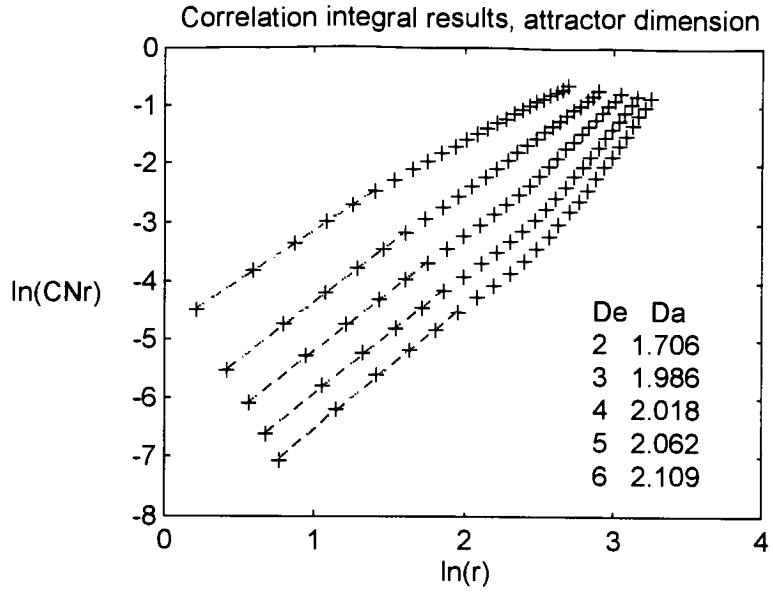


Figure 35. Case 10. Using 5000 points, every third point, embedding delay of one half the first local minimum. Appendix pages A-33 to A-36.

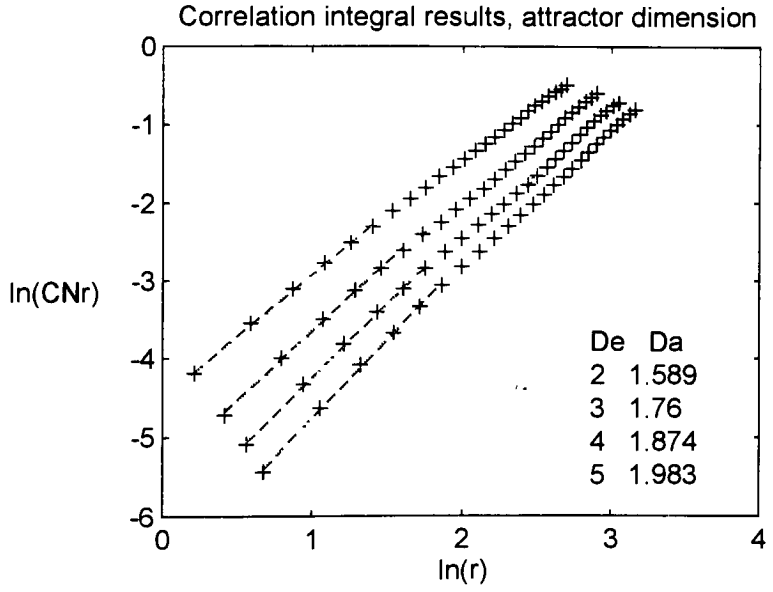


Figure 36. Case 11. Using 5000 points, every third point, embedding delay of one fifth the first local minimum. Appendix pages A-37 to A-39.

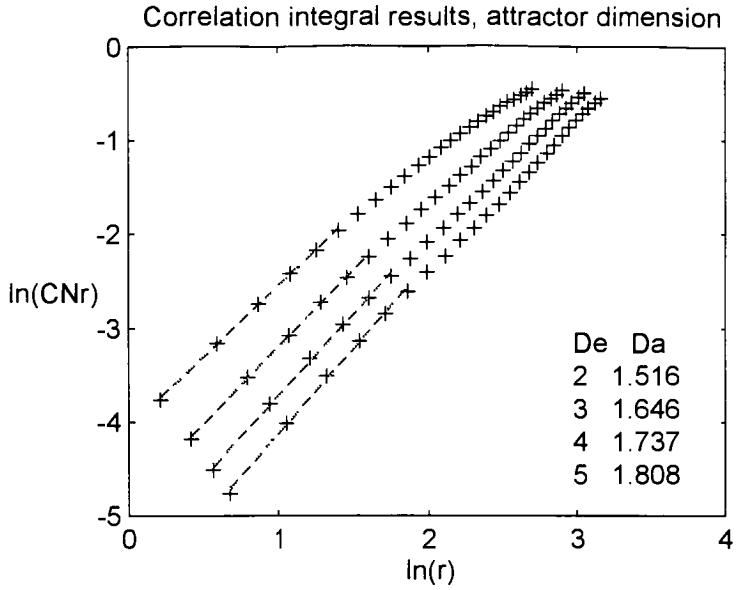


Figure 37. Case 12. Using 5000 points, every third point, embedding delay of one tenth the first local minimum. Appendix pages A-40 to A-42.

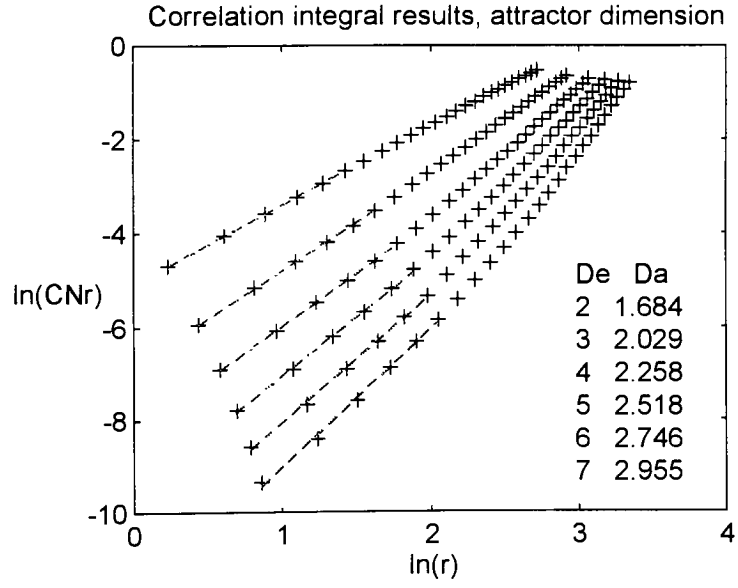


Figure 38. Case 13. Using 5000 points, every fifth point, embedding delay of the first local minimum. Appendix pages A-43 to A-47.

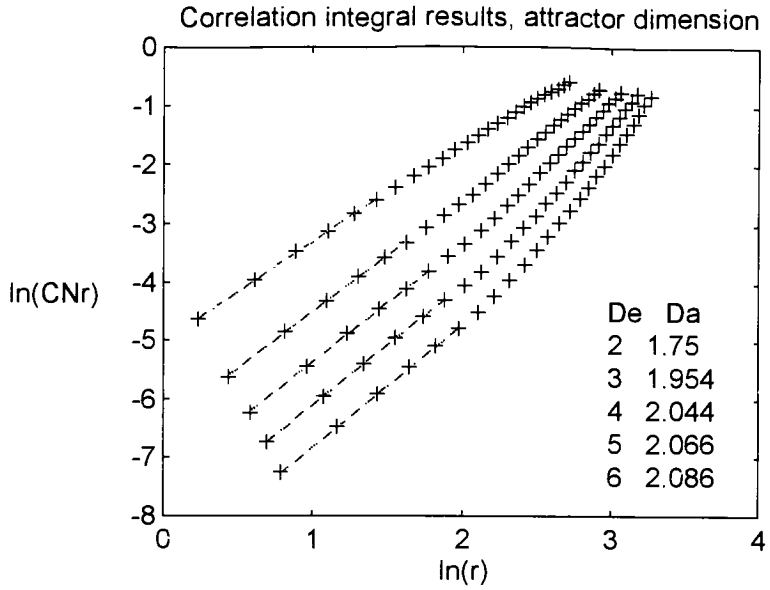


Figure 39. Case 14. Using 5000 points, every fifth point, embedding delay of one half the first local minimum. Appendix pages A-44 to A-51.

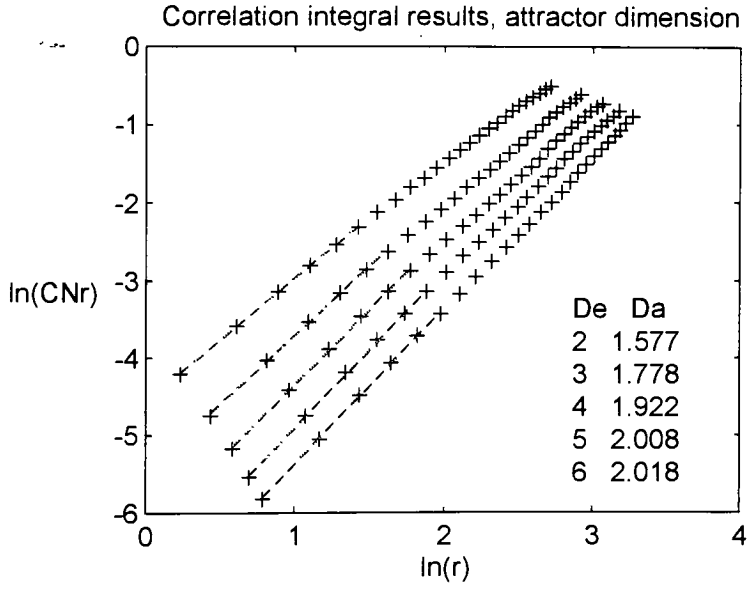


Figure 40. Case 15. Using 5000 points, every fifth point, embedding delay of one fifth the first local minimum. Appendix pages A-52 to A-55.

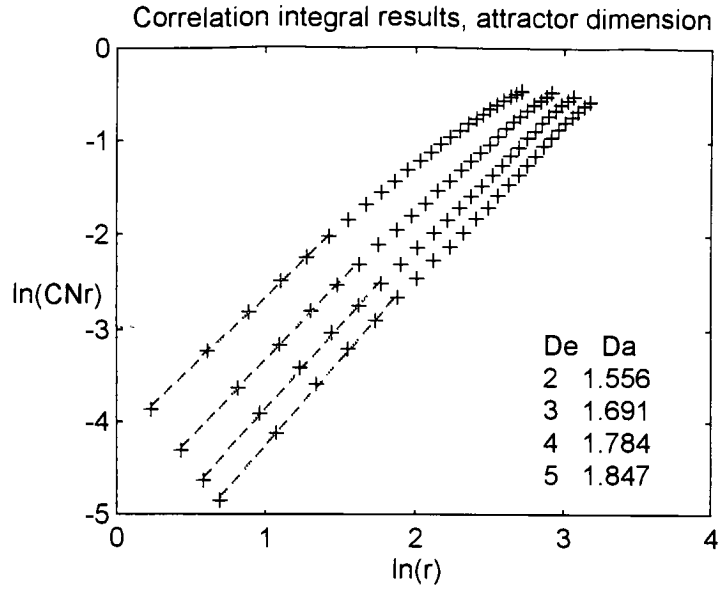


Figure 41. Case 16. Using 5000 points, every fifth point, embedding delay of one tenth the first local minimum. Appendix pages A-56 to A-58.

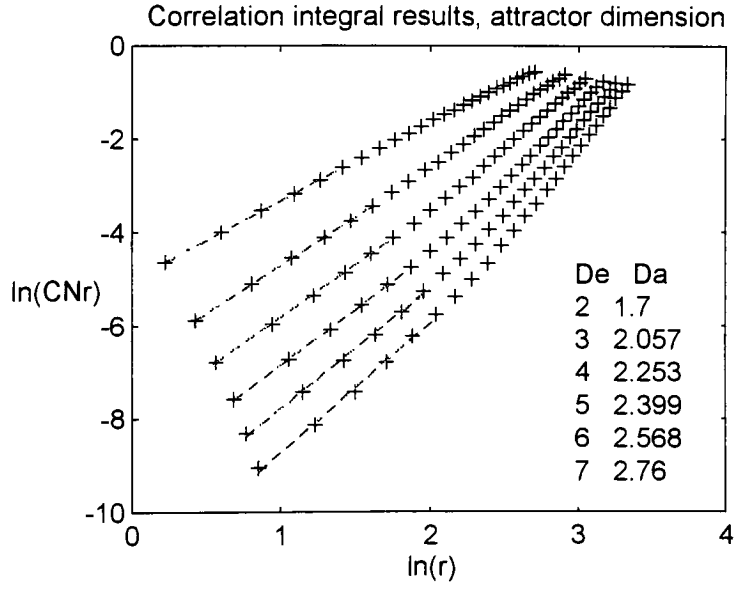


Figure 42. Case 17. Using 5000 points, every tenth point, embedding delay of the first zero. Appendix pages A-59 to A-63.

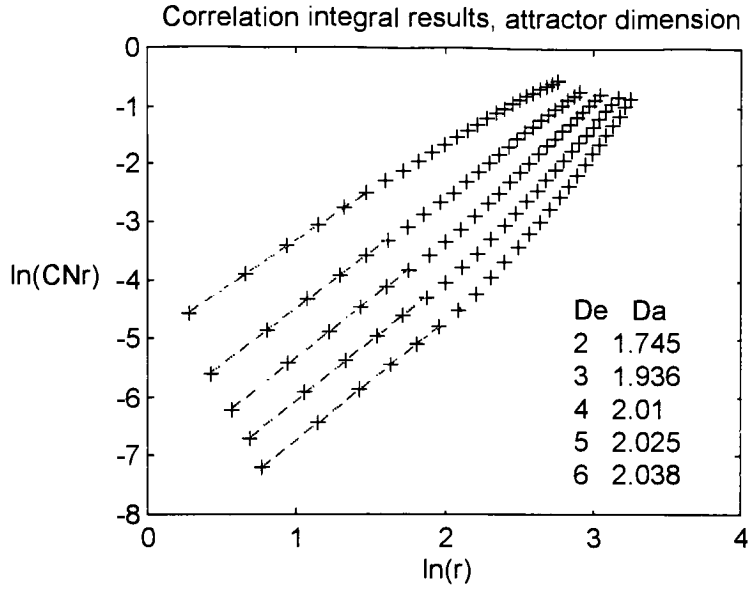


Figure 43. Case 18. Using 5000 points, every tenth point, embedding delay of one half of the first zero. Appendix pages A-64 to A-67.

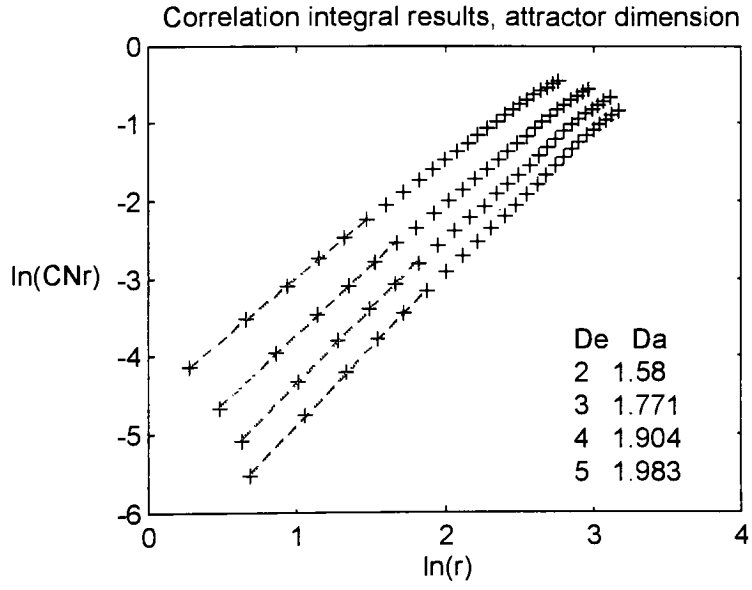


Figure 44. Case 19. Using 5000 points, every tenth point, embedding delay of one fifth of the first zero. Appendix pages A-68 to A-70.

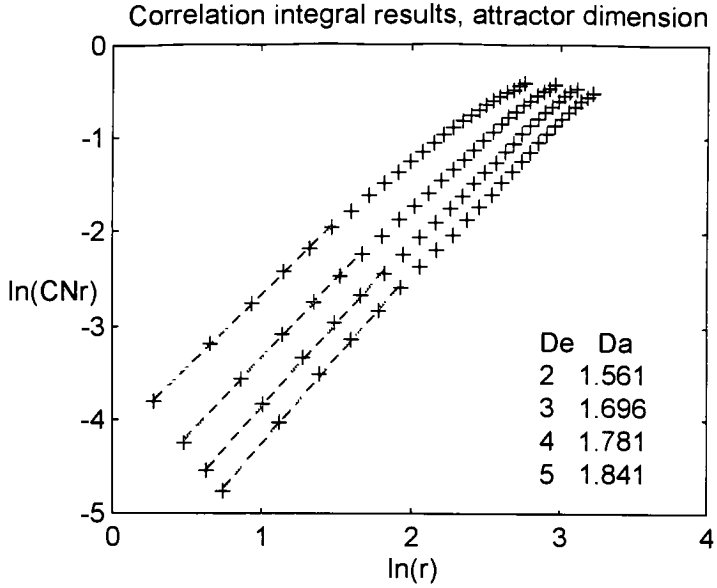


Figure 45. Case 20. Using 5000 points, every tenth point, embedding delay of one tenth of the first zero. Appendix pages A-71 to A-73.

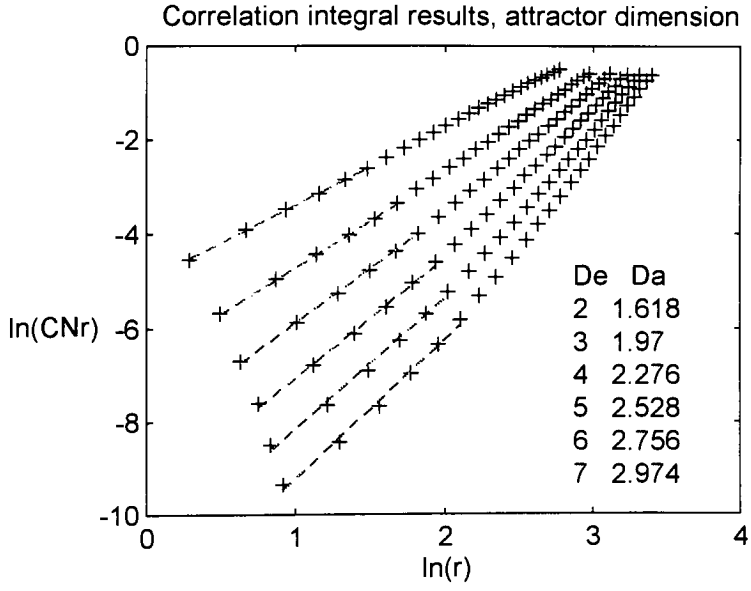


Figure 46. Case 21. Using 5000 points, every 25th point, embedding delay of the first local minimum. Appendix pages A-74 to A-78.

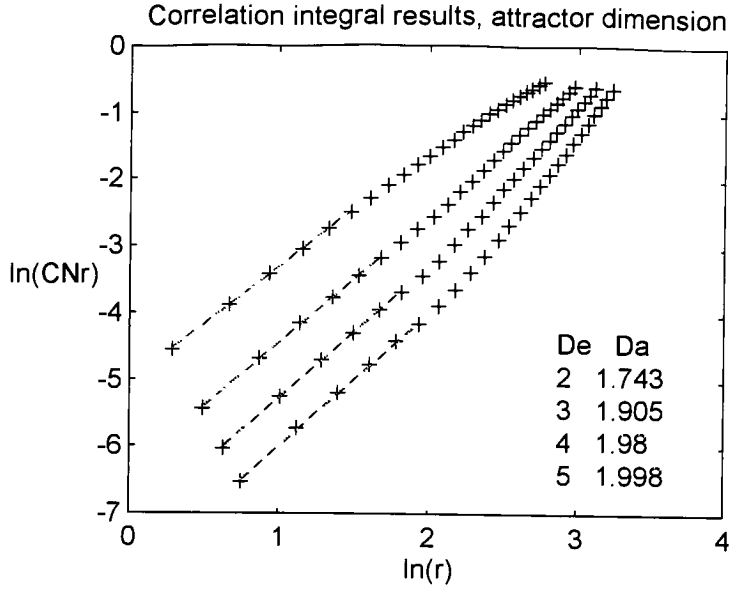


Figure 47. Case 22. Using 5000 points, every 25th point, embedding delay of one half the first local minimum. Appendix pages A-79 to A-81.

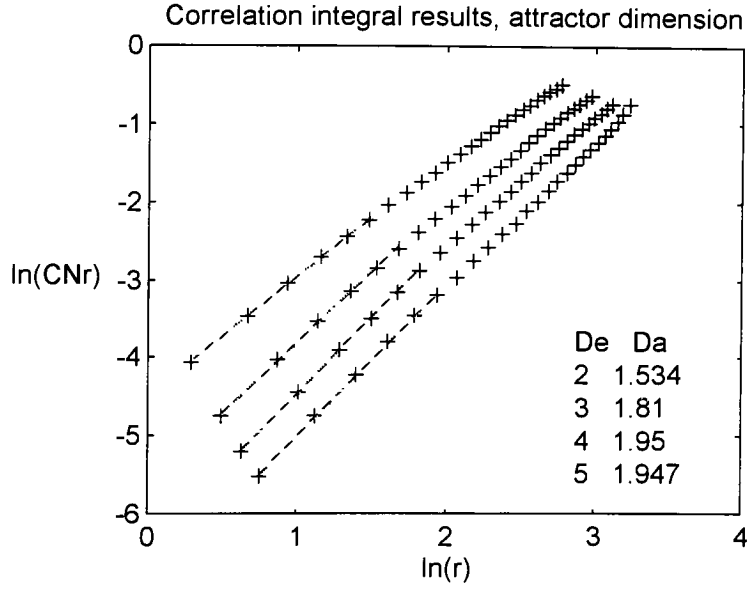


Figure 48. Cases 23-24. Using 5000 points, every 25th point, embedding delay of one fifth and one tenth (duplicate) the first local minimum. Appendix pages A-82 to A-87.

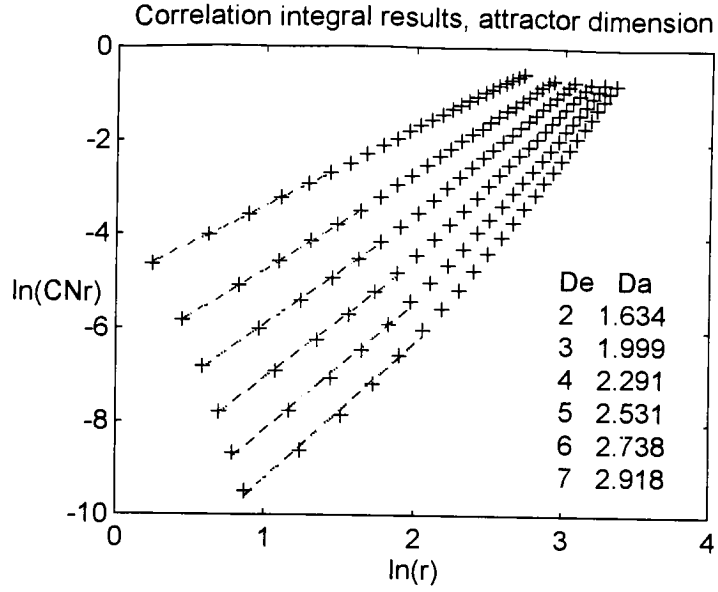


Figure 49. Case 27. Using 5000 points, every 50th point, embedding delay of one fifth of the first zero. Appendix pages A-88 to A-92.

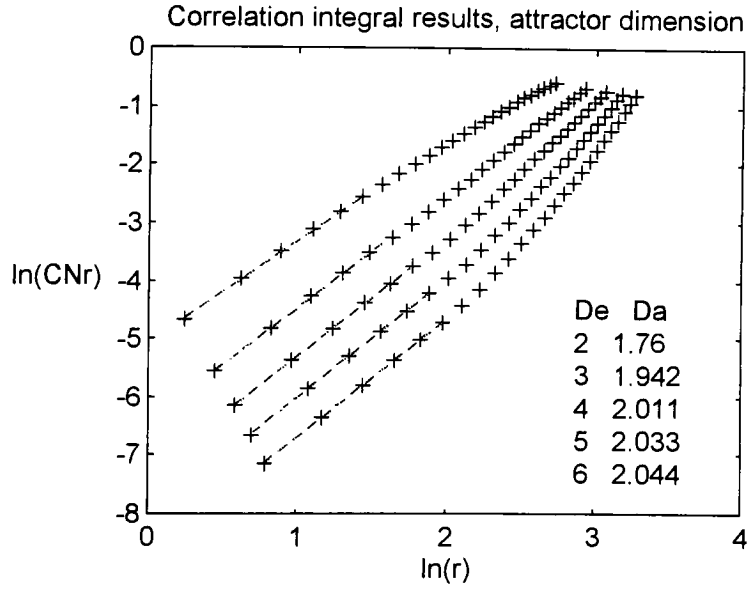


Figure 50. Case 28. Using 5000 points, every 50th point, embedding delay of one tenth of the first zero. Appendix pages A-93 to A-96.

CHAPTER 7

CONCLUSIONS

The most important result of this work was the discovery that the secondary sampling technique may be effectively used to reduce the size of the data set for analysis. The data set size can be reduced without losing information. The results show an increase in the estimation accuracy as the secondary sampling becomes sparser. This is followed by a decrease in estimation accuracy as the secondary sampling became very sparse.

The analysis is however very sensitive to the embedding delay. For embedding delays corresponding to 1/2 and 1/5 of the decorrelation time, the attractor dimension estimates were good, typically falling within 5% of the correct value. At these embedding delays, a very wide range of secondary sampling rates yield acceptable results. The choice of a good embedding delay allows the investigation of the sampling rate's effect on the attractor dimension estimation.

For the embedding delays corresponding to 1 and 1/10 of the decorrelation time only one case provided a reasonable estimate of the attractor dimension. The other cases were very poor estimates indicating that the embedding delay selection was poor.

RECOMMENDATIONS

Throughout this investigation a great many things became clear. First, without the computers and software none of this work would have been possible. Secondly, the present software level (both analysis and word processing) does not match the present hardware level. Lastly, when they have hardware and software that is capable of doing real-time analysis and prediction of complex non-linear systems it will be a truly impressive day.

Probably the most frustrating discovery was that although it was very easy to use, MATLAB was not meant to solve problems this large or so many times. The language is intuitive because it uses matrix notation and because the syntax is very simple. The MATLAB environment has two fundamental problems. The first problem is that it is an interpreted language, that is the commands in the script file must be interpreted each time which takes up valuable time. The second problem is the structure which allows very fast execution of matrix operations is crippled when it is forced to perform a looping operation that can not be vectorized. The suggestion then is to use a language which will create executable files which eliminates both shortcomings. The use of an executable requires more work in the programming department, but should result in greatly reduced run time.

Due to time constraints only one attractor was investigated. With a faster combination of hardware and software it would be desirable to investigate other

attractors to see how widely the secondary sampling technique is applicable. While there is nothing to suggest that the method is not extensible, neither is there anything to prove that it is extensible to other systems.

Additional reading

A great deal of information has been published in the last two years since this work was begun. The amount of information in print may even have doubled in this time with so many people working on different aspects of system analysis. The following list is from research on the subject beginning two years ago. Some of the entries are references in published works, while others are the result of specific subject searches.

No attempt has been made to update the list from two years ago, it is merely provided as a starting point for someone with further interest. The items on the list cover a wide variety of topics associated with non-linear dynamics, some which are applicable to this work and some which are not.

Abarbanel, H.D.I., Brown, R., Kadtke, J.B., Prediction in chaotic nonlinear systems: Methods for time series with broadband Fourier spectra, *Physical Review A*, 41, 1782 (1990)

Aleksik, Z., Estimating the embedding dimension, *Physica D*, 52, 362 (1991)

Asano, A., Itoh, K., Ichioka, Y., The nearest-neighbor median filter; deterministic properties and implementations, *Pattern recognition*, 23, 10, 1059 (1990)

Atmanspacher, H., Scheingraber, H., Voges, W., Global scaling properties of a chaotic attractor reconstructed from experimental data, *Physical Review A*, 1988, 37, 1314 (1988)

Badii, R., Politi, A., Title unknown, *Journal of Statistical Physics*, 40, 725

Ben-Mizrachi, A., Procaccia, I., Grassberger, P., Characterization of experimental (noisy) strange attractors, *Physical Review A*, 29, 975 (1984)

Broomhead, D.S., Indik, R., Newell, A.C., Local adaptive Galerkin bases for large-dimensional dynamical systems, *Nonlinearity*, 4, 159
Broomhead, D.S., Jones, R., Time series analysis, *Proceedings of the Royal Society of London*, 423, 1864, 103
Broomhead, D.S., King, G.P., Title unknown, *Physica D*, 20, 217
Buzug, Th., Pfister, G., Optimal delay time and embedding dimension for delay-time coordinates by analysis of the global static and local dynamic behavior of strange attractors, *Physical Review A*, 45, 7073 (1992)
Casdagli, M., Chaos and Deterministic versus Stochastic Non-linear Modelling, *Journal of the Royal Statistical Society B*, 54, 303 (1992)
Casdagli, M., Nonlinear prediction of chaotic time series, *Physica D*, 35, 335 (1989)
Cawley, R., Hsu, G.H., SNR performance of a noise reduction algorith applied to coarsely sampled chaotic data, *Physics Letters A*, 166, 188 (1992)
Chennaoui, A., Liebler, J., Schuster, H.G., The mechanism of the increase of the generalized dimension of a filtered chaotic time series, *Journal of Statistical Physics*, 59, 1311 (1990)
Chennaoui, A., Pawelzik, K., Liebert, W., Attractor reconstruction from filtered chaotic time series, *Physical Review A*, 41, 4151 (1990)
Corless, R.M., Essex, C., Nerenberg, M.A.H., Numerical methods can suppress chaos, *Physics Letters A*, 157, 27
Crutchfield, Inferring statistical complexity, *Physical Review Letters*, 63, 105 (1989)
Crutchfield, J.P., Farmer, J.D., Packard, N.H., & Shaw, R.S., Chaos, *Scientific American*, Dec. '86, 46 (1986)
Cutler, C.D., Dawson, D.A., Nearest-neighbor analysis of a family of fractal distributions, *The annals of probability*, 18, 1, 256 (1990)
Ding, M., Grebogi, C., Ott, E., Dimensions of strange nonchaotic attractors, *Physics Letters A*, 137, 167
Douglas, J., Seeking Order in Chaos, *EPRI Journal*, Jun '92, 5 (1992)
Dvorak, I., Klanschka, J., Modification of the Grassberger-Procaccia algorith ..., *Physics Letters A*, 145, 225 (1990)
Eckmann, J.-P., Procaccia, I., Fluctuation of dynamical scaling indices in non-Linear systems, *Physical Review A*, 34, 659 (1986)
Eckmann, J.-P., Ruelle, D., Ergodic theory of chaos and strange attractors, *Reviews of Modern Physics*, 57, 617 (1985)
Eckmann, J.-P., Ruelle, D., Fundamental limitations for estimating dimensions and Lyapunov exponents in dynamical sytems, *Physica D*, 56, 185
Eden, A., Local estimates for the Hausdorff dimension of an attractor, *Journal of mathematical analysis and application*, 150, 100 (1990)
Elsner, J.B., Predicting time series using a neural network as a method of distinguishing chaos from noise, *Journal of Physics A*, 25, 843
Elsner, J.B., Tsonis A.A., Nonlinear prediction, chaos, and noise, *Bulletin*, 73, 49
Essex, C., Lookman, T., Nerenberg, M.A.H., The climate attractor over short timescales, *Letters to Nature*, 326, 64 (1987)

Essex, C., Nerenberg, M.A.H., Comments on 'Deterministic chaos: the science and the fiction' by D. Ruell, *Procedings of the Royal Society of London A*, 435, 287 (1991)
Essex, C., Nerenberg, M.A.H., Fractal dimension: Limit capacity or Hausdorff dimension?, *American Journal of Physics*, 58, 986 (1990)
Farmer, J.D., A Rosetta Stone for connectionist, *Physica D*, 42, 153
Farmer, J.D., An analytic approach to practical state space reconstruction, *Physica D*, 57, 1 (1992)
Farmer, J.D., Chaotic attractors of an Infinite Dimensional System, *Physica D*, 4, 366
Farmer, J.D., Deterministic noise amplifiers, *Physica D*, 55, 155
Farmer, J.D., Optimal shadowing and noise reduction, *Physica D*, 47, 373 (1991)
Farmer, J.D., Ott, E., Yorke, J.A., The dimension of chaotic attractors, *Physica D*, 7, 153 (1983)
Farmer, J.D., State space reconstruction with noise, *Physica D*, 51, 52
Farrell, M.E., Passamante, A., Hediger, T., Comparing nearest-neighbor estimator of local attractor dimensions for noisy data to the correlation dimension, *Physical Review A*, 41, 6591 (1990)
Frank, G.W., Lookman, T., Nerenberg, M.A.H., Chaotic time series analyses of epileptic seizures, *Physica D*, 46, 427
Frank, G.W., Lookman, T., Nerenberg, M.A.H., Recovering the attractor: A review of chaotic time-series analysis, *Canadian Journal of Physics*, 68, 711 (1990)
Fraser, A.M., Information and Entropy in Strange Attractors, *IEEE Transactions on Information Theory*, 35, 245 (1989)
Fraser, A.M., Reconstructing attractors from scalar time series: a comparison of singular system and redundancy criteria, *Physica D*, 34, 391
Fraser, A.M., Swinney, H.L., Independent coordinates for strange attractors from mutual information, *Physical Review A*, 33, 1134 (1986)
Fryska, S.T., Zohdy, M.A., Computer dynamics and shadowing of chaotic orbits, *Physics Letters A*, 166, 340 (1992)
Gil, M.A., A note on the choice of sample size in estimating the mutual information, *Applied mathematics and computation*, 30, 125 (1989)
Giona, M., Lentini, F , Cimagalli, V., Functional reconstruction and local prediction of chaotic time series, *Physical Review A*, 44, 3496 (1991)
Grassberger, P., An optimized box-assisted algorithm for fractal dimensions, *Physics Letters A*, 148, 63
Grassberger, P., Do Climatic attractors exist?, *Nature*, 323, 609 (1986)
Grassberger, P., Information content and predictability of lumped and distributed dynamical systems, *Physica Scripta*, 40, 346
Grassberger, P., Noise induced escape from attractors, *Journal of Physics A*, 22, 3283.
Grassberger, P., On the symbolic dynamics of the Henon map, *Journal of Physics A*, 22, 5217 (1989)

Grassberger, P., Procaccia, I., Characterization of Strange Attractors, *Physical Review Letters*, 50, 346 (1983)
Grassberger, P., Procaccia, I., Dimensions and entropies of strange attractors from a fluctuating dynamics approach, *Physica D*, 1984, 13, 34
Grassberger, P., Procaccia, I., Estimation of the Kolmogorov entropy from a chaotic signal, *Physical Review A*, 28, 2591 (1983)
Grassberger, P., Procaccia, I., Measuring the strangeness of strange attractors, *Physica D*, 1983, 9, 189
Greenside, H.S., Wolf, A., Swift, J., Pignataro, T., Impracticality of a box-counting algorith for calculating the dimensionality of strange attractors, *Physical Review A*, 25, 3453 (1982)
Gudowska-Nowak, E., Stochastic versus chaotic dynamics in a deterministic system, *Journal of Statistical Physics*, 54, 539 (1989)
Halsey, T.C., Jensen, M.H., Kadanoff, L.P., Procaccia, I., Shraiman, B.I., Fractal measures and their singularities: The characterization of strange sets, *Physical Review A*, 33, 1141 (1986)
Hammel, S.M., A noise reduction method for chaotic systems, *Physics Letters A*, 148, 421 (1990)
Havstad, J.W., Ehlers, C.L., Attractor dimension of nonstationary dynamical sytems from small data sets, *Physical Review A*, 39, 845 (1989)
Huang, J.Z., Tsang, K.Y., Hanson, J.D., Dimension of strange attractors in four-dimensional maps, *Physics Letters A*, 147, 269
Jimenez, J., Moreno, J.A., Ruggeri, G.J., Forcasting on chaotic time series: A local optimal linear-reconstruction method, *Physical Review A*, 45, 3553 (1992)
Kennel, M.B., Brown, R., Abarbanel, H.D.I., Determining embedding dimension for phase-space reconstruction using a geometrical construction, *Physical Review A*, 45, 3403 (1992)
Kostelich, E.J., Yorke, J.A., Noise reduction: finding the simplest dynamical system consistent with the data, *Physica D*, 41, 183
Landa, P.S., Rosenblyum, M.G., A comparison of methods of constructing a phase space ..., *Soviet physics*, 34, 1229 (1989)
Landa, P.S., Rosenblyum, M.G., Method for evaluating the embedding dimension ... experimental results, *Soviet Physics*, 34, 6 (1989)
Lee, S.J., Halsey, T.C., Some results on multifractal correlations, *Physica A*, 164, 575 (1990)
Li, W., Mutual information functions versus correlation functions, *Journal of Statistical Physics*, 60, 823 (1990)
Liebert, W., Pawelzik, K., Schuster, H.G., Optimal embeddings of chaotic attractors from topological considerations, *Europhysics letters*, 14, 521 (1991)
Liebert, W., Schuster, H.G., Proper choice of the time for the analysis of chaotic time series, *Physics Letters A*, 142, 107
Lindsay, P.S., An efficient method of forecasting chaotic time series using linear interpolation, *Physics Letters A*, 153, 353 (1991)

Mackey, M.C., Glass, L., Oscillation and Chaos in Physiological Control Systems, *Science*, 197, 287 (1977)
Marteau, P.F., Abarbanel, H.D.I., Noise reduction in chaotic time series... scaled probablistic methods, *Journal of nonlinear science*, 1, 313 (1991)
Martinerie, J.M., Albano, A.M., Mees, A.I., Rapp, P.E., Mutual information, strange attractors, and the optimal estimation of dimension, *Physical Review A*, 45, 7058 (1992)
Nerenberg, M.A.H., Essex, C., Correlation dimension and systematic geometric effects, *Physical Review A*, 42, 7065 (1990)
Nicolis, C., Nicolis, G., Chaotic Dynamics, Markov partitions and Zipf's Law, *Journal of Statistical Physics*, 54, 915
Nicolis, C., Nicolis, G., Evidence for climatic attractors, *Nature*, 326, 523 (1987)
Nicolis, C., Nicolis, G., Is there a climatic attractor?, *Nature*, 311, 529 (1984)
Nicolis, C., Probablistic aspects of error growth in atmospheric dynamics, *Quarterly journal of the Royal Meteorological Society*, 118, 553
Packard, N.H., Crutchfield, J.P., Farmer, J.D., & Shaw, R.S., Geometry from a Time Series, *Physical Review Letters*, 45, 712 (1980)
Pawelzik, K., Schuster, H.G., Generalized dimensions and entropies from a measured time series, *Physical Review A*, 35, 481 (1987)
Peeters, P., Nicolis, G., Intrinsic fluctuations in chaotic dynamics, *Physica A*, 188, 426
Peterson, I., Ribbon of Chaos, *Science News*, 139, 60 (1991)
Ruelle, D., Strange Attractors, *Mathematical Intelligencer*, 2, 126 (1980)
Russel, D.A., Hanson, J.D., Ott, E., Dimension of Strange Attractors, *Physical Review Letters*, 45, 1175 (1980)
Sawada, Y., Matsushita, M., Yamazaki, M., Kondo, H., Title unknown, *Physics Scripta*, T9, 130
Schnabl, W., Stadler, P.F., Forst, C., Full characterization of a strange attractor, *Physica D*, 48, 65 (1991)
Schuster, H.G., Information content of chaotic signals, *Physica Scripta*, 40, 367
Shan, L., Goertz, C.K., Smith, R.A., On the Embedding-Dimension Analysis of AE and AL Time Series, *Geophysical Research Letters*, 18, 1647 (1991)
Shaw, R., Title unknown, *Z. Naturforsch*, 36, 80 (1981)
Shreiber, T., Grassberger, P., A simple noise-reduction method for real data, *Physics Letters A*, 160, 411
Smith, R.L., Estimating Dimension in Noisy Chaotic Time Series, *Journal of the Royal Statistical Society B*, 54, 329 (1992)
Takens, F., Title unknown, *Lecture Notes in Mathematics*, 894
Termonia, Y., Alexandrovitch, Z., Title unknown, *Physical Review Letters*, 45, 1265
Theiler, J., Estimating the fractal dimension of chaotic time series, *The Lincoln Laboratory Journal*, 3, 63 (1990)
Tsonis, A.A., & Elsner, J.B., Chaos, strange attractors, and weather, *Bulletin of the American meteorological society*, 70, 14 (1989)

Tsonis, A.A., & Elsner, J.B., Multiple attractors, Fractal basins and longterm climate dynamics, *Contributions to atmospheric physics; Beitrage*, 63, 171

Tsonis, A.A., & Elsner, J.B., Nonlinear prediction as a way of distinguishing chaos from random fractal sequences, *Nature*, 358, 217 (1992)

Tsonis, A.A., & Elsner, J.B., The weather attractor over very short timescales, *Nature*, 333, 545 (1988)

Tsonis, A.A., The effect of truncation and round-off on computer simulated chaotic trajectories, *Computers & mathematics with applications*, 21, 93

Wales, D.J., Calculating the loss of information from chaotic time series by forcasting, *Nature*, 350, 485 (1991)

Wolff, R.C., A note on the behavior of the correlation integral in the presence of a time series, *Biometrika*, 77, 689 (1990)

Wolfram, S., Title unknown, *Reviews of Modern Physics*, 55, 601

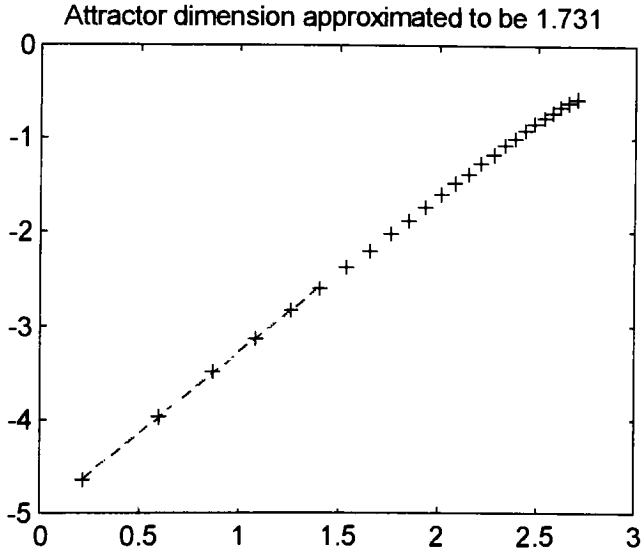
Zardecki, A., Title unknown, *Physics Letters A*, 90, 274 (1982)

Zhao, D., Lookman, T., Critical exponents for c-animals with nearest neighbor interaction, *Journal of Physics A*, 25, 2181

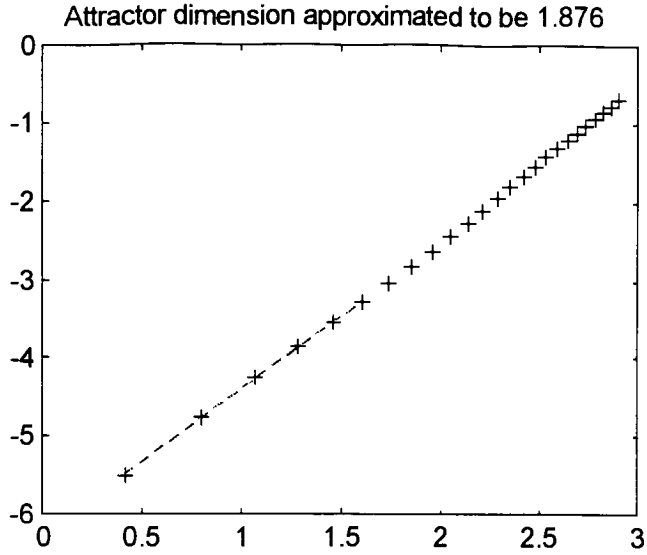
Appendix A -- Experimental Documentation

Time series analysis program by William Robertson.
18-Jun-94
12:20:47.79
Loading time series.
Read 300001 points from time series.
Using last 5,000 points of time series.
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 53.
12:21:43.71
Elapsed time 56.52 seconds

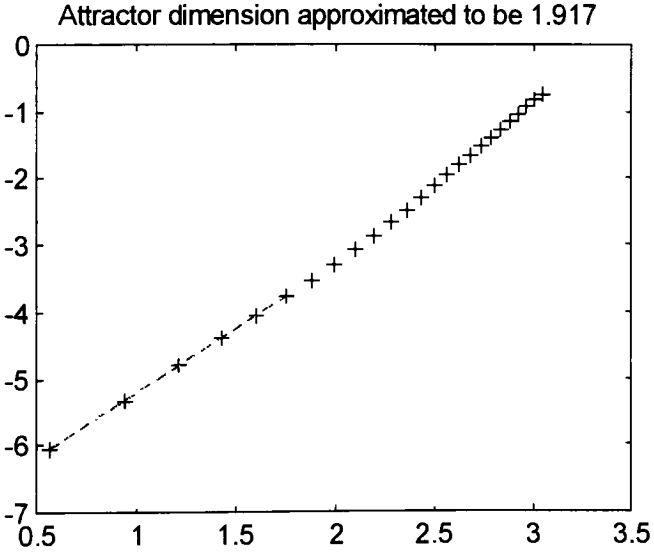
2-Dimensional embedding.
Using embedding delay of 53.
Number of embedded points = 4943
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:35:41.76
Elapsed time 1.169e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 53.
Number of embedded points = 4890
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:52:6.35
Elapsed time 2.348e+004 seconds.

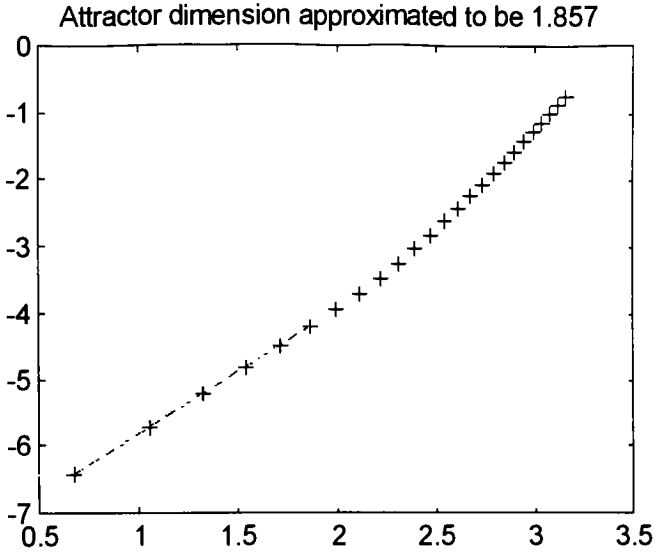


4-Dimensional embedding.
Using embedding delay of 53.
Number of embedded points = 4837
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
22:11:26.93
Elapsed time 3.544e+004 seconds.



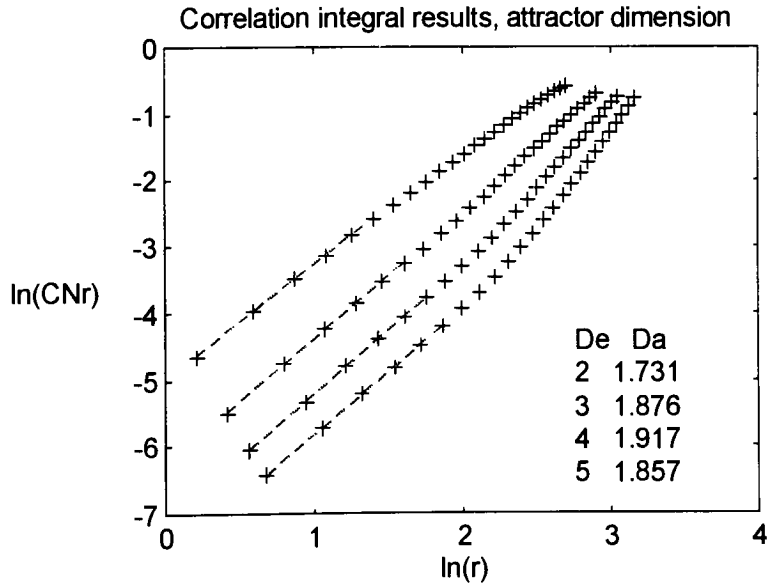
5-Dimensional embedding.
Using embedding delay of 53.
Number of embedded points = 4784
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
1:32:46.36

Elapsed time 4.752e+004 seconds.



Embedding procedure complete.

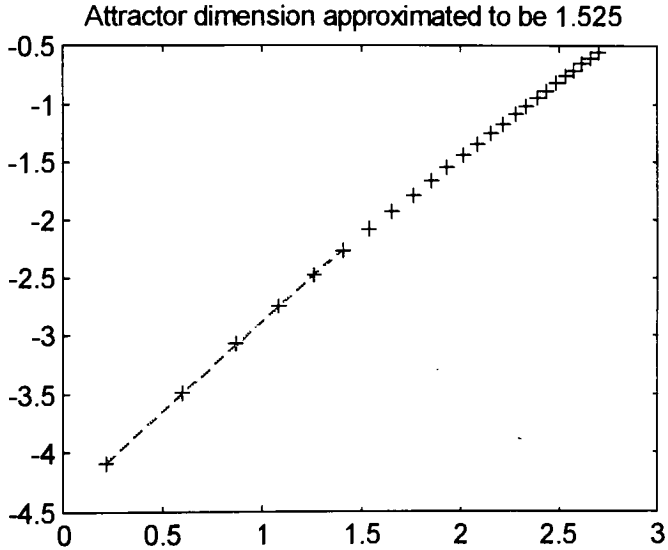
Embedding Dimension	Attractor Dimension
2	1.731
3	1.876
4	1.917
5	1.857



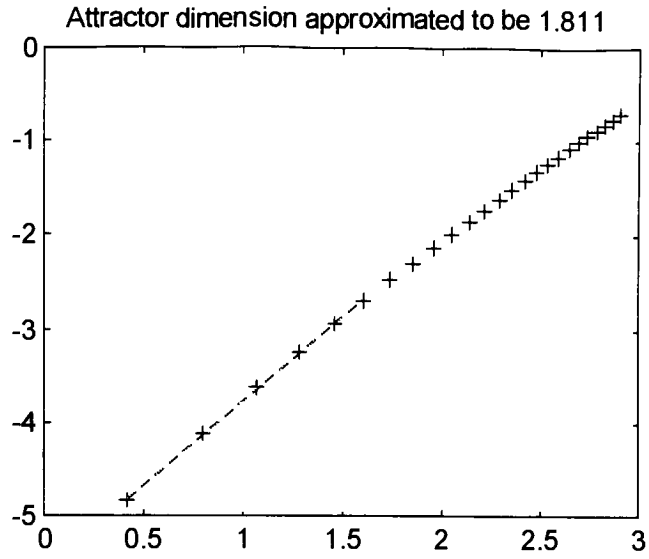
1:32:51.47
Elapsed time 4.752e+004 seconds.
Floating point operations = 3.681e+009.
normal termination

Time series analysis program by William Robertson.
19-Jun-94
7:32:21.48
Loading time series.
Read 300001 points from time series.
Using last 5.000 points of time series.
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 53.
7:33:14.76
Elapsed time 53.61 seconds

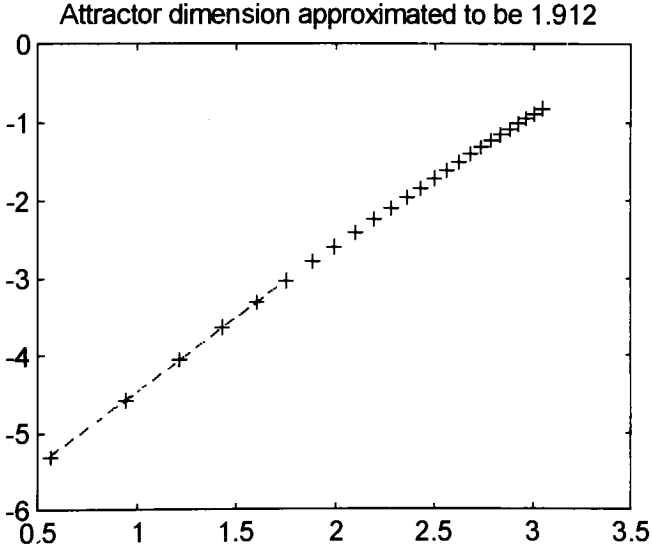
2-Dimensional embedding.
Using embedding delay of 27.
Number of embedded points = 4969
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
10:47:35.33
Elapsed time 1.171e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 27.
Number of embedded points = 4942
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
14:7:39.63
Elapsed time 2.372e+004 seconds.

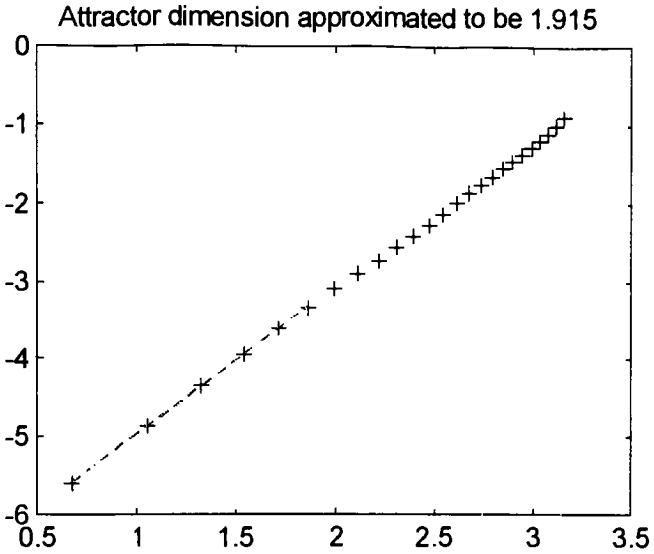


4-Dimensional embedding.
Using embedding delay of 27.
Number of embedded points = 4915
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
17:32:56.18
Elapsed time 3.603e+004 seconds.

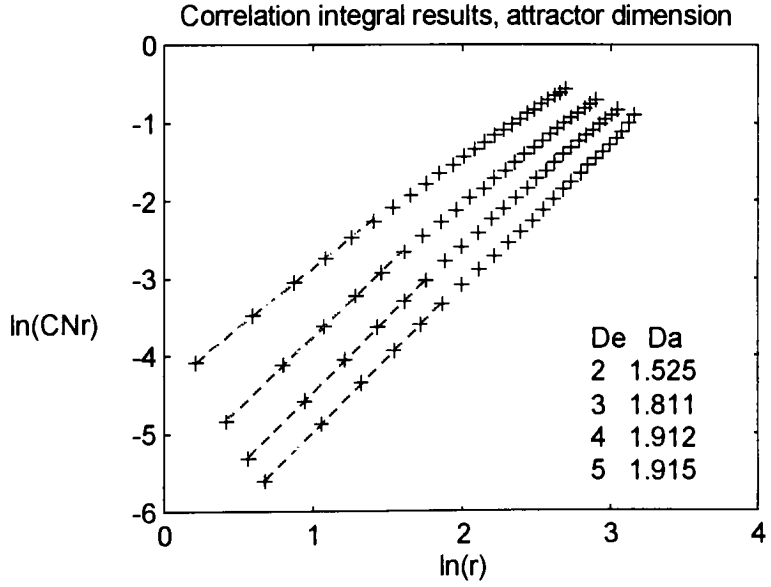


5-Dimensional embedding.
Using embedding delay of 27.
Number of embedded points = 4888
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
21:1:53.74

Elapsed time 4.857e+004 seconds.
Embedding procedure complete.



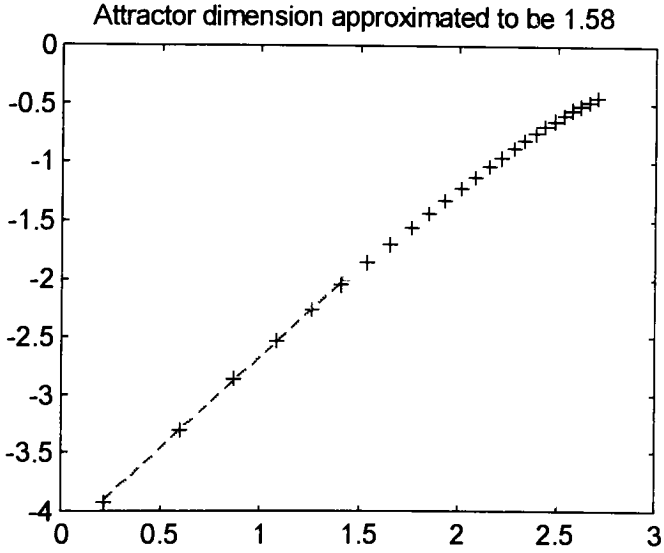
Embedding Dimension	Attractor Dimension
2	1.525
3	1.811
4	1.912
5	1.915



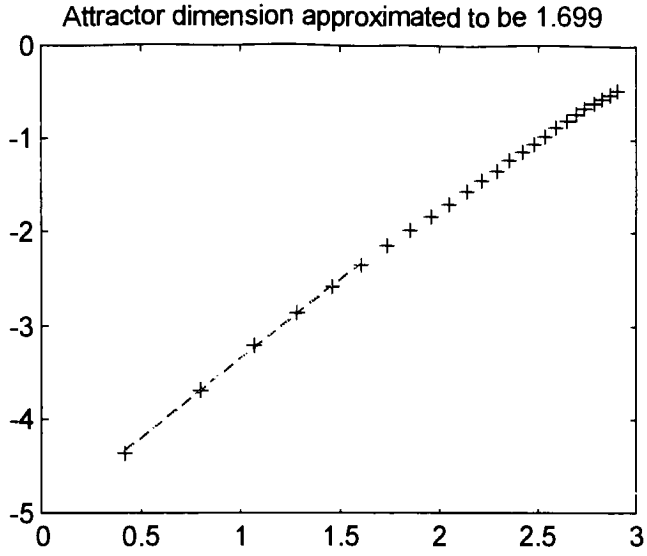
21:1:58.36
Elapsed time 4.858e+004 seconds.
Floating point operations = 3.785e+009.
normal termination

Time series analysis program by William Robertson.
19-Jun-94
21:57:25.74
Loading time series.
Read 300001 points from time series.
Using last 5.000 points of time series.
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 53.
21:58:6.33
Elapsed time 41.03 seconds

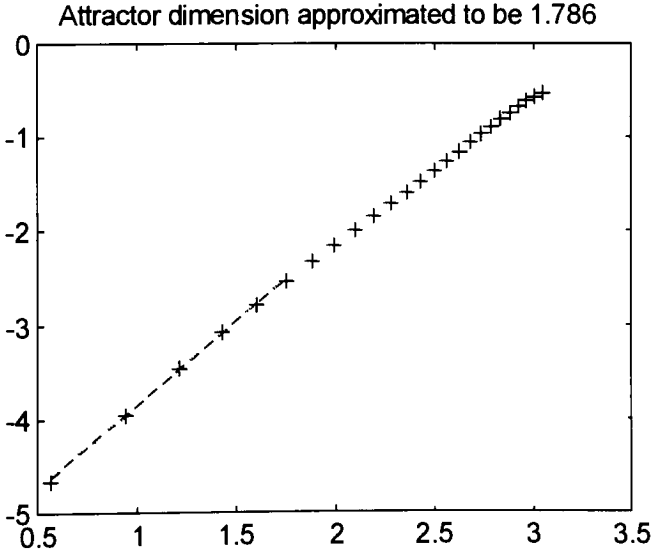
2-Dimensional embedding.
Using embedding delay of 11.
Number of embedded points = 4985
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
1:14:39.72
Elapsed time 1.183e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 11.
Number of embedded points = 4974
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
4:37:10.17
Elapsed time 2.398e+004 seconds.

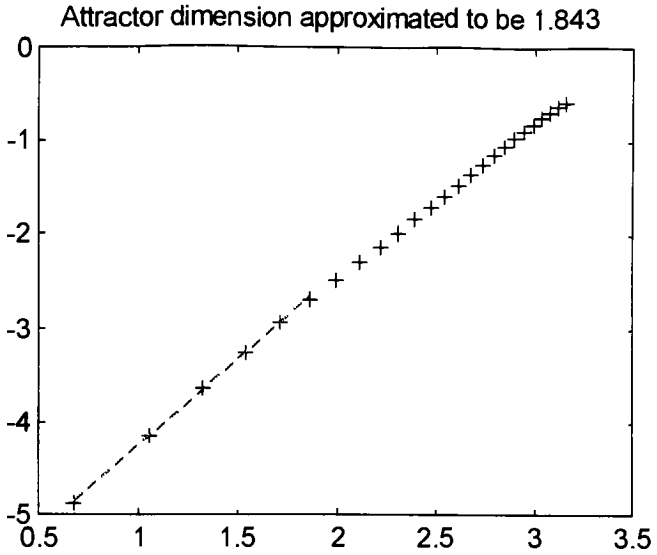


4-Dimensional embedding.
Using embedding delay of 11.
Number of embedded points = 4963
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
8:5:33.68
Elapsed time 3.649e+004 seconds.



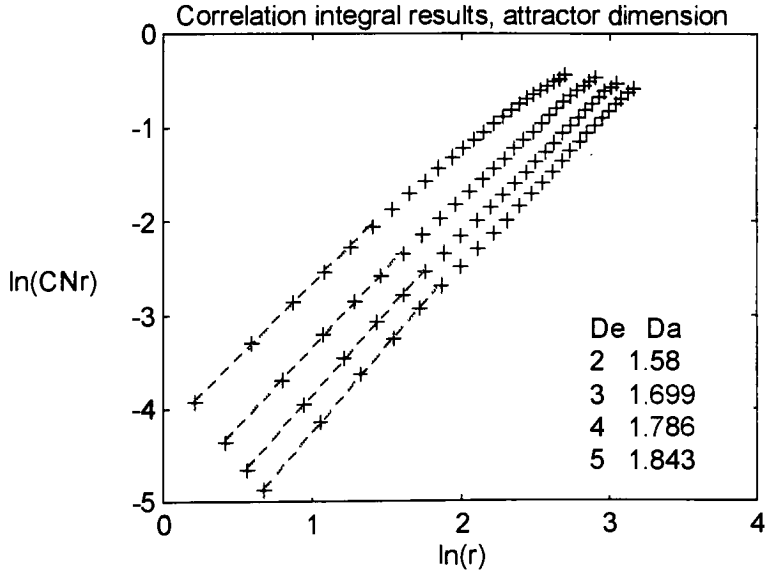
5-Dimensional embedding.
Using embedding delay of 11.
Number of embedded points = 4952
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
11:42:34.82

Elapsed time 4.951e+004 seconds.



Embedding procedure complete.

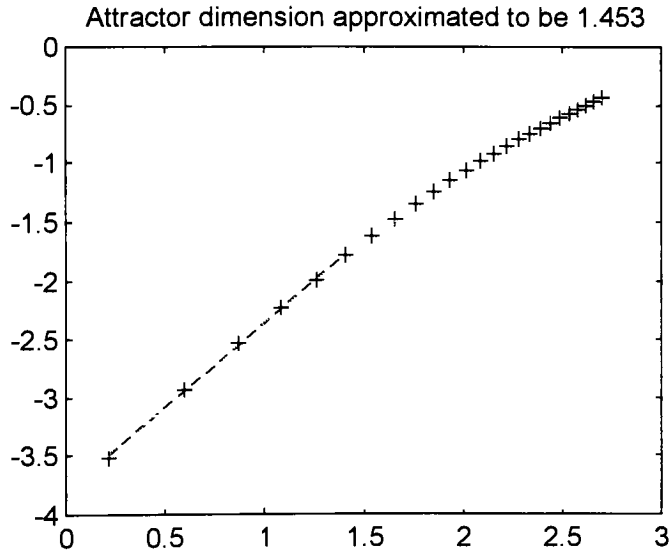
Embedding Dimension	Attractor Dimension
2	1.58
3	1.699
4	1.786
5	1.843



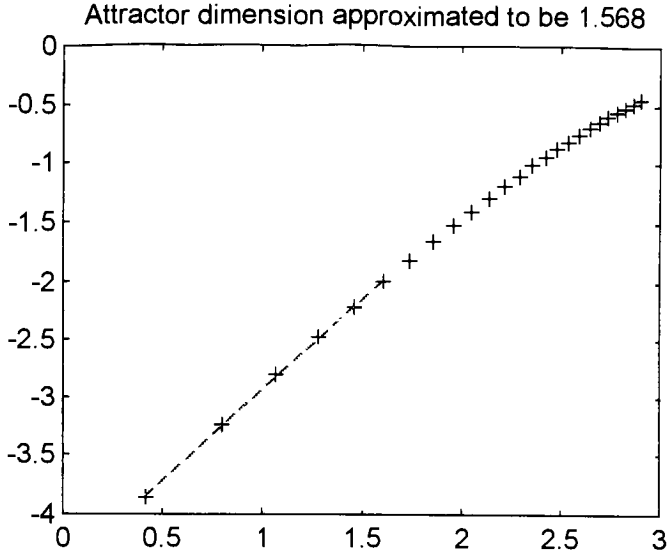
11:42.39.32
Elapsed time 4.951e+004 seconds.
Floating point operations = 3.85e+009.
normal termination

Time series analysis program by William Robertson.
20-Jun-94
12:17:29.02
Loading time series.
Read 300001 points from time series.
Using last 5.000 points of time series.
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 53.
12:18:18.39
Elapsed time 49.93 seconds

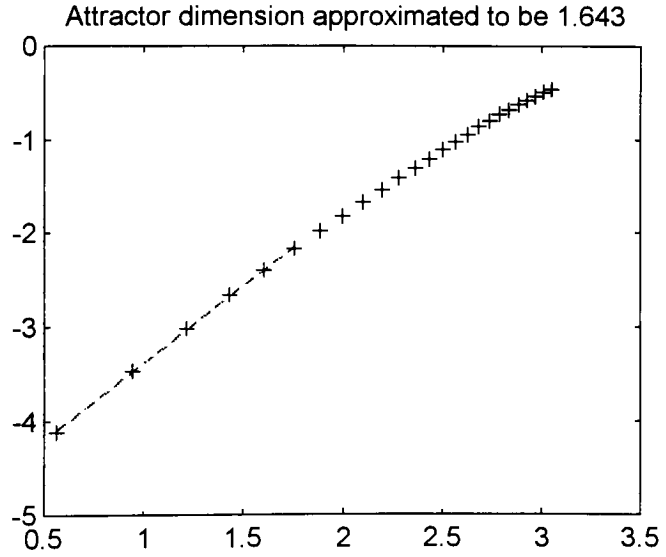
2-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:33:34.66
Elapsed time 1.177e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4984
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:56:58.83
Elapsed time 2.397e+004 seconds.

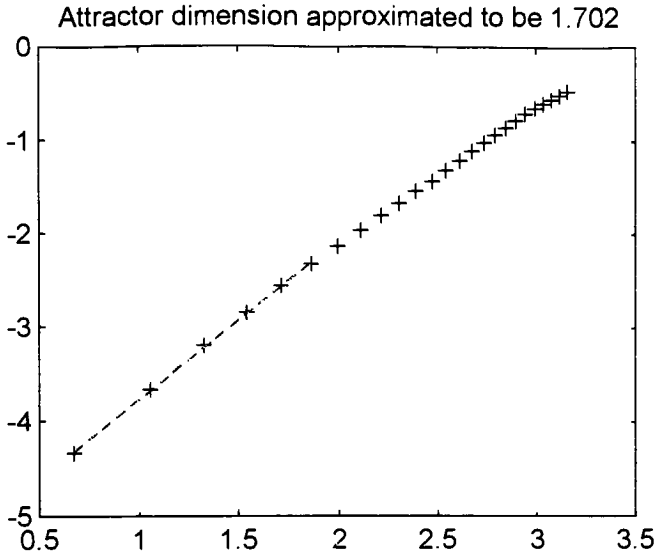


4-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4978
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
22:37:27.03
Elapsed time 3.72e+004 seconds.



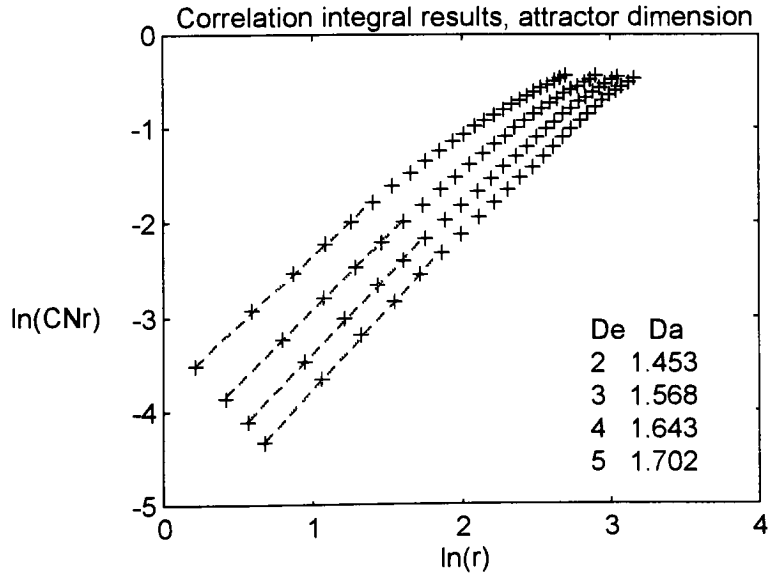
5-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4972
Using W factor of 4.
Finished correlation integral.
Finished LINREG.

2:11:12.85
Elapsed time 5.002e+004 seconds.



Embedding procedure complete.

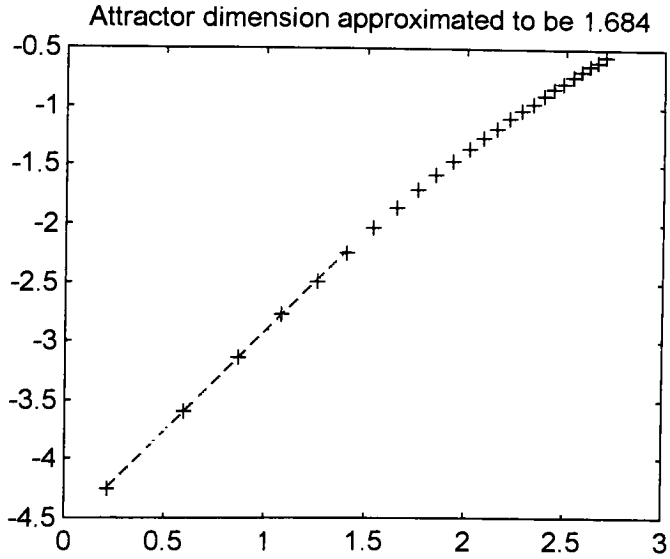
Embedding Dimension	Attractor Dimension
2	1.453
3	1.568
4	1.643
5	1.702



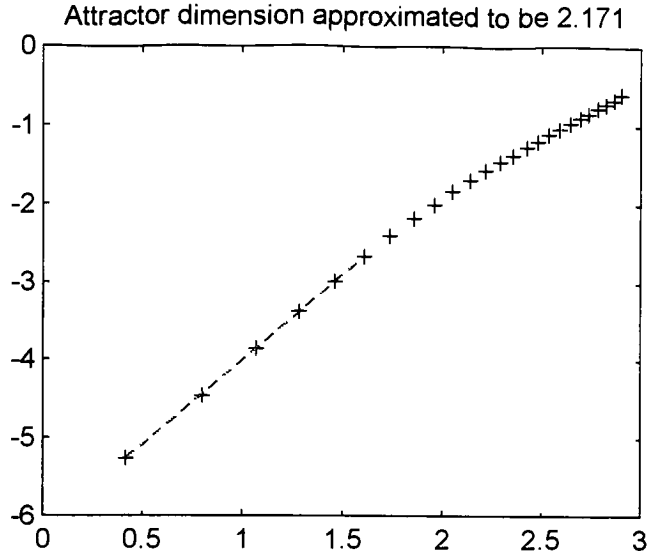
2:11:17.02
Elapsed time 5.003e+004 seconds.
Floating point operations = 3.87e+009.
normal termination

Time series analysis program by William Robertson.
29-Jun-94
19:45:3.63
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 2 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 38.
19:45:41.69
Elapsed time 38.51 seconds

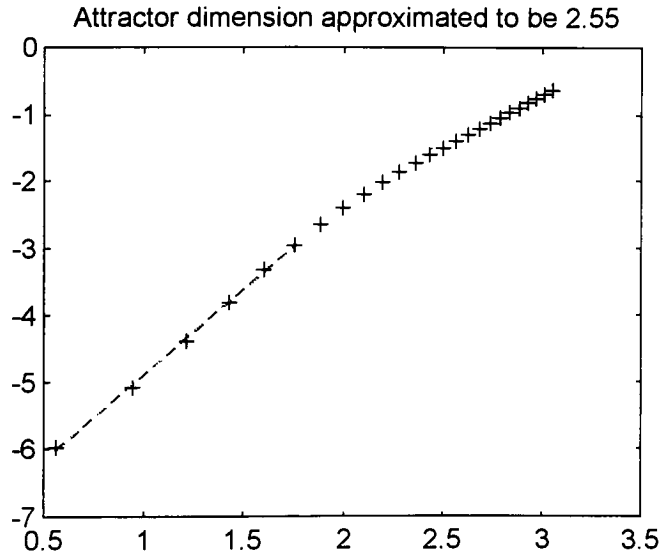
2-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4958
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
23:0:14.62
Elapsed time 1.171e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4920
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
2:17:22.17
Elapsed time 2.354e+004 seconds.

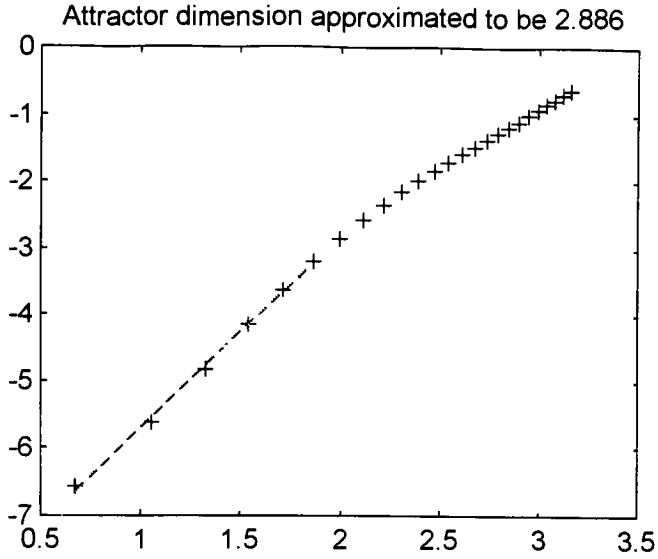


4-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4882
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
5:38:4.03
Elapsed time 3.558e+004 seconds.

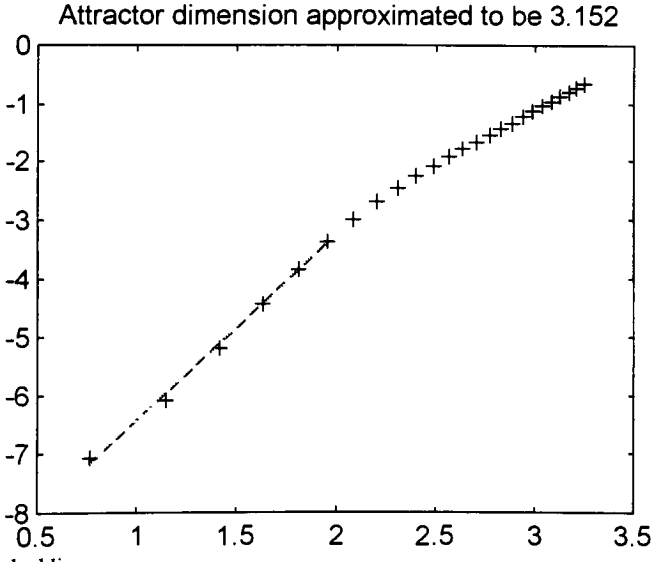


5-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4844
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

9:4:56.7
Elapsed time 4.799e+004 seconds.

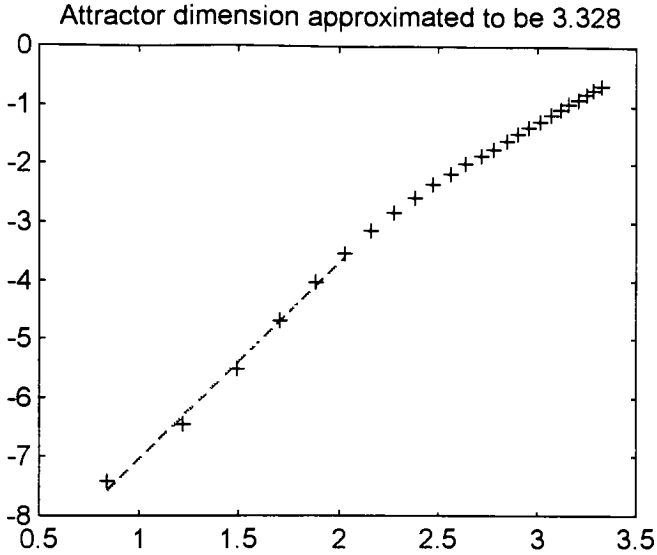


6-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4806
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
12:33:0.72
Elapsed time 6.048e+004 seconds.

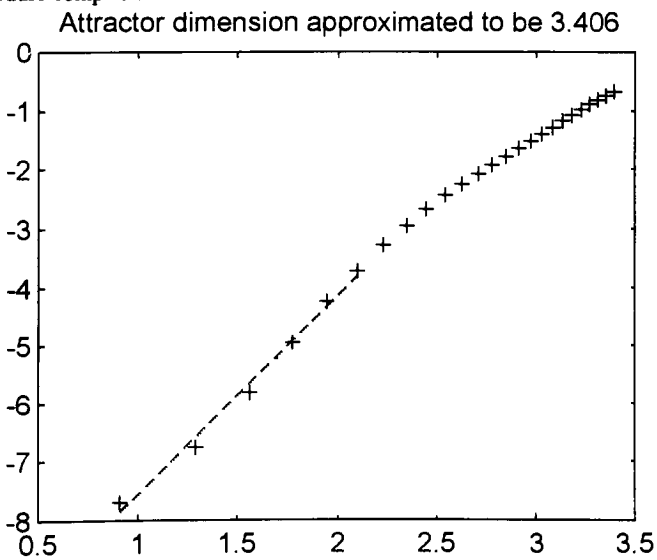


7-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4768
Using W factor of 6.

Finished correlation integral.
Finished LINREG.
16:5:6.9
Elapsed time 7.32e+004 seconds.

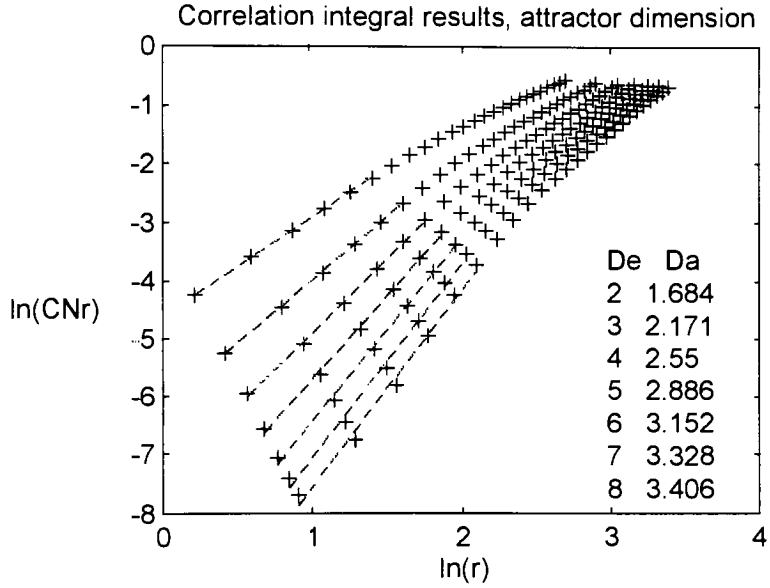


8-Dimensional embedding.
Using embedding delay of 38.
Number of embedded points = 4730
Using W factor of 7.
Finished correlation integral.
Finished LINREG.
19:39:37.7
Elapsed time 8.607e+004 seconds.
Embedding procedure complete.



Embedding Attractor

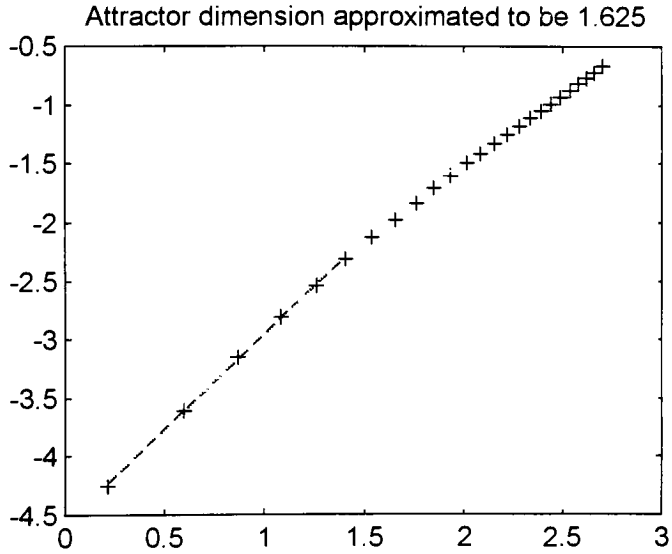
Dimension	Dimension
2	1.684
3	2.171
4	2.55
5	2.886
6	3.152
7	3.328
8	3.406



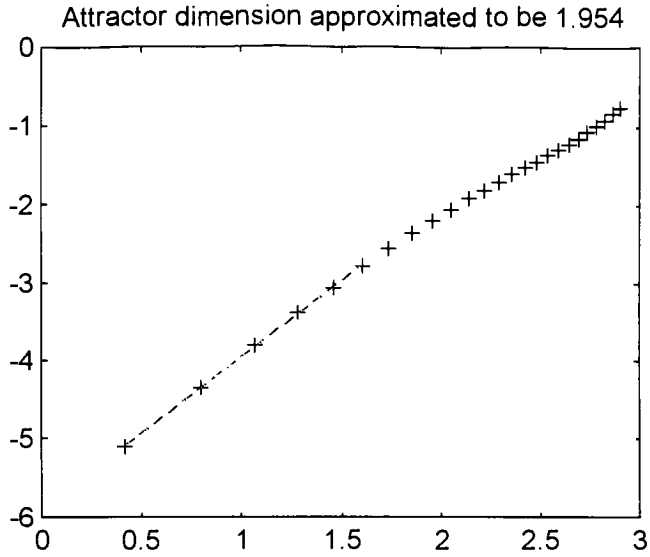
19:39:42.97
Elapsed time 8.608e+004 seconds.
Floating point operations = 7.352e+009.
normal termination

Time series analysis program by William Robertson.
29-Jun-94
19:38:33.22
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 2 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 38.
19:38:53.1
Elapsed time 20.15 seconds

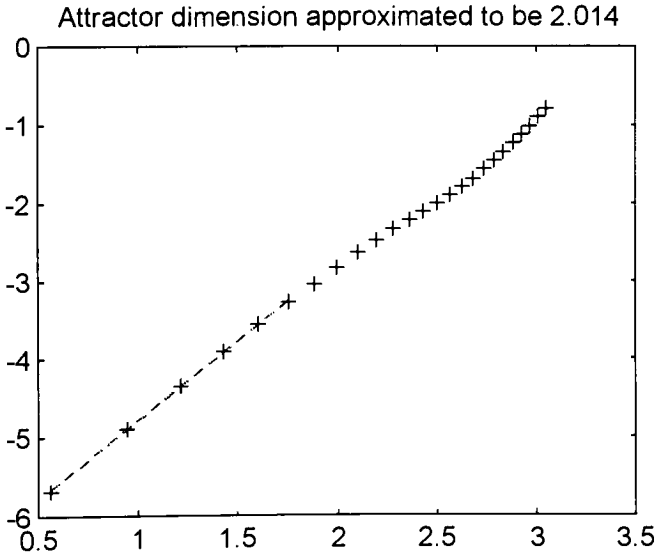
2-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4977
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
20:59:15.89
Elapsed time 4843 seconds.



3-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4958
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
22:22:46.52
Elapsed time 9853 seconds.

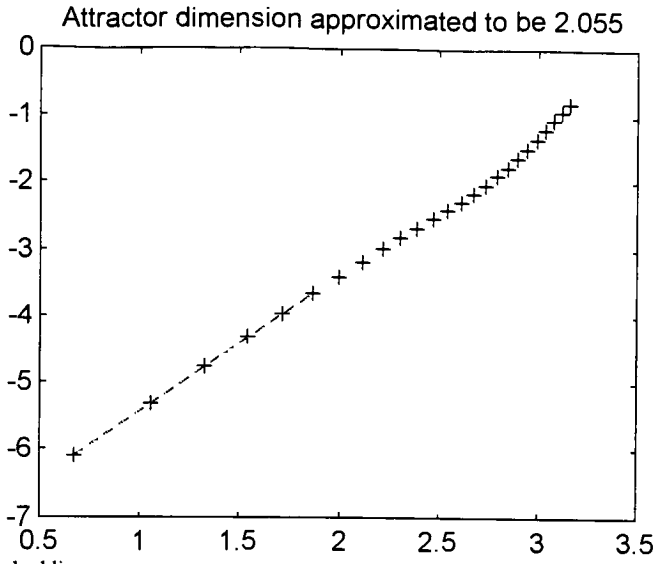


4-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4939
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
23:49:18.85
Elapsed time 1.505e+004 seconds.

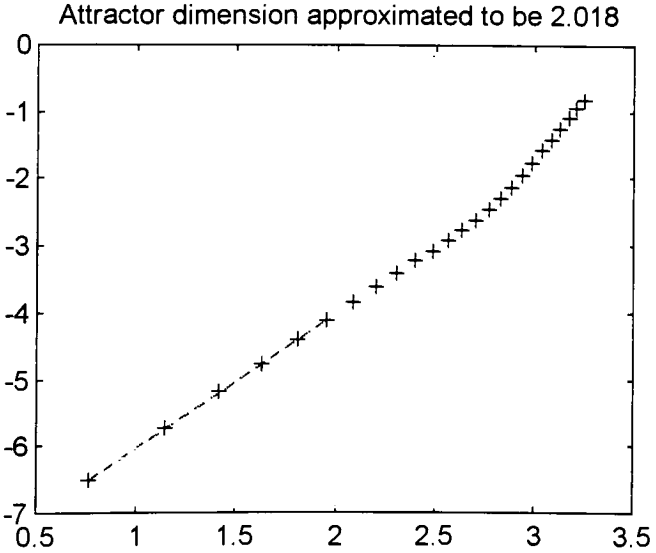


5-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4920
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

1:19:18.25
Elapsed time 2.045e+004 seconds.

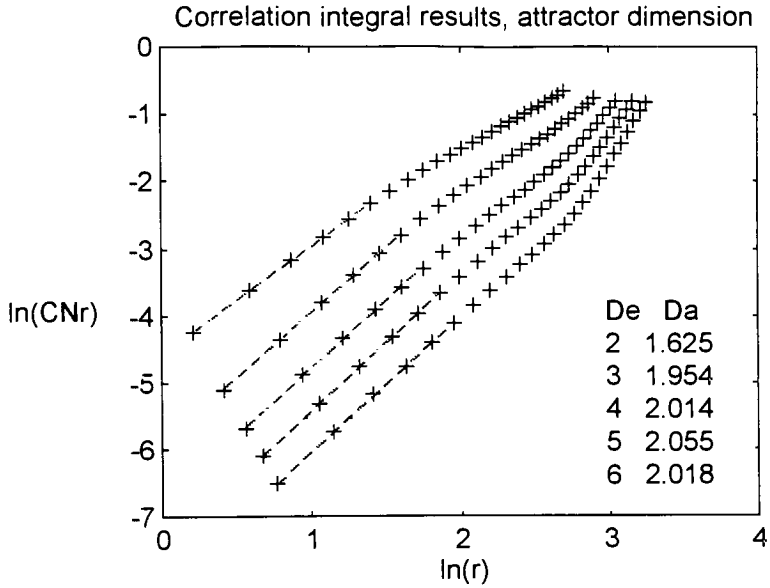


6-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4901
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
2:52:1.76
Elapsed time 2.601e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.625

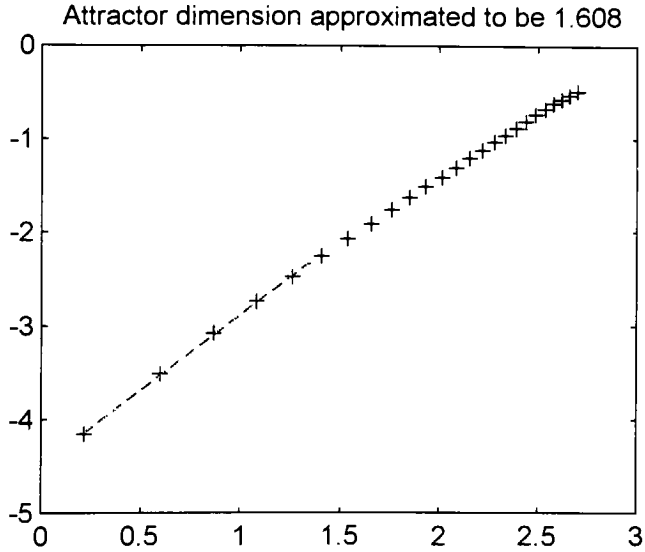
3 1.954
4 2.014
5 2.055
6 2.018



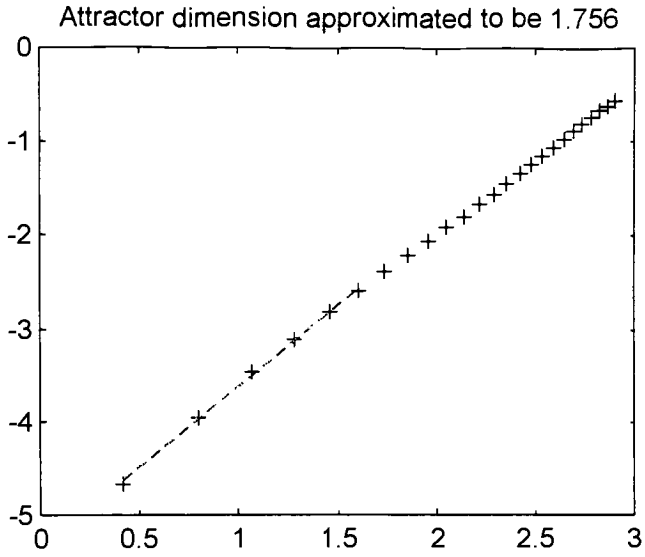
2:52:3.9
Elapsed time 2.601e+004 seconds.
Floating point operations = 4.994e+009.
normal termination

Time series analysis program by William Robertson.
30-Jun-94
6:29:35.74
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 2 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 38.
6:29:57.71
Elapsed time 22.19 seconds

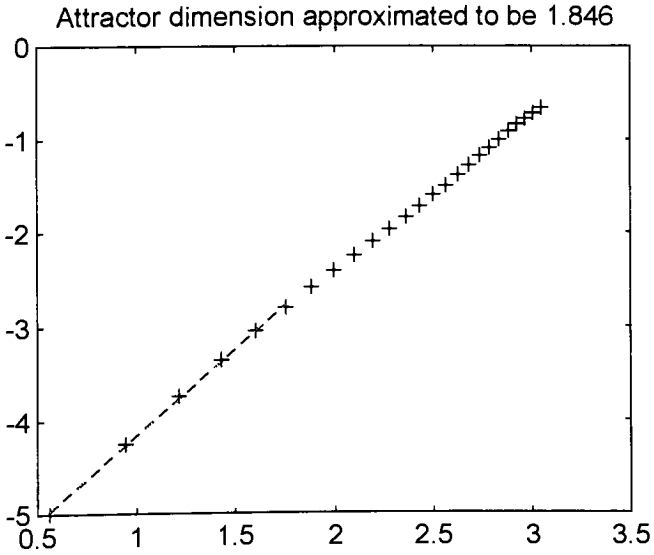
2-Dimensional embedding.
Using embedding delay of 8.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
7:50:31.09
Elapsed time 4855 seconds.



3-Dimensional embedding.
Using embedding delay of 8.
Number of embedded points = 4980
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
9:15:47.9
Elapsed time 9972 seconds.

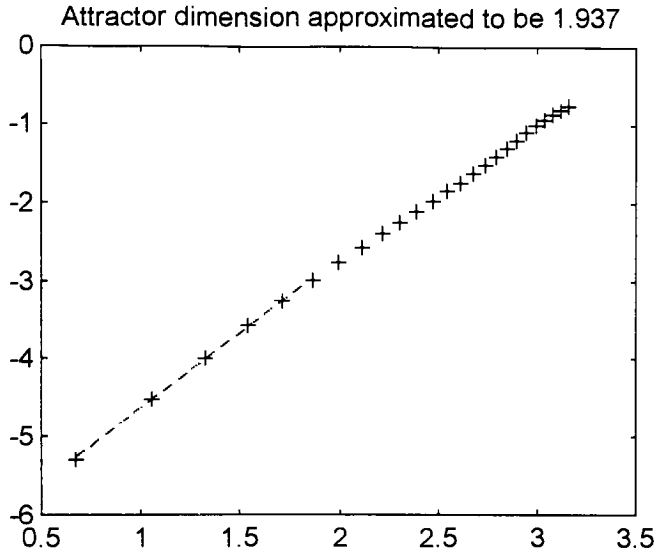


4-Dimensional embedding.
Using embedding delay of 8.
Number of embedded points = 4972
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
10:42:36.7
Elapsed time 1.518e+004 seconds.

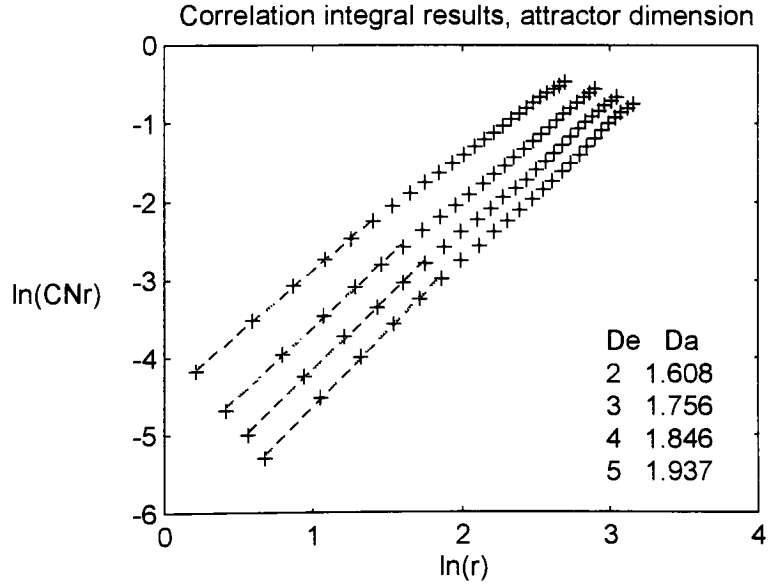


5-Dimensional embedding.
Using embedding delay of 8.
Number of embedded points = 4964
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

12:13:29.21
Elapsed time 2.063e+004 seconds.
Embedding procedure complete.



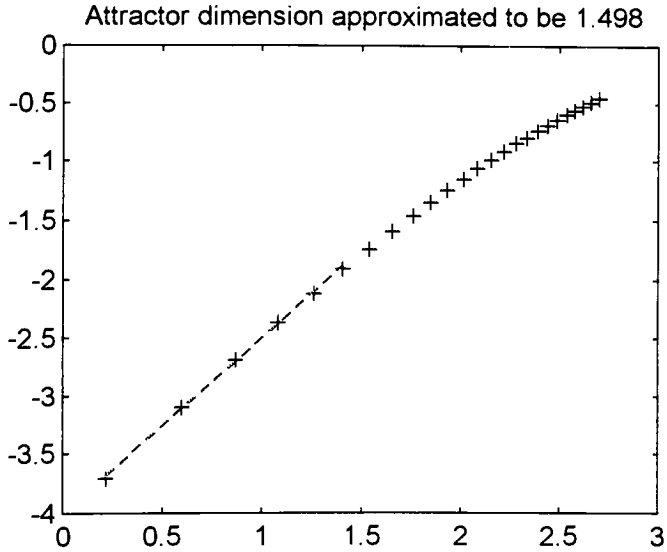
Embedding Dimension	Attractor Dimension
2	1.608
3	1.756
4	1.846
5	1.937



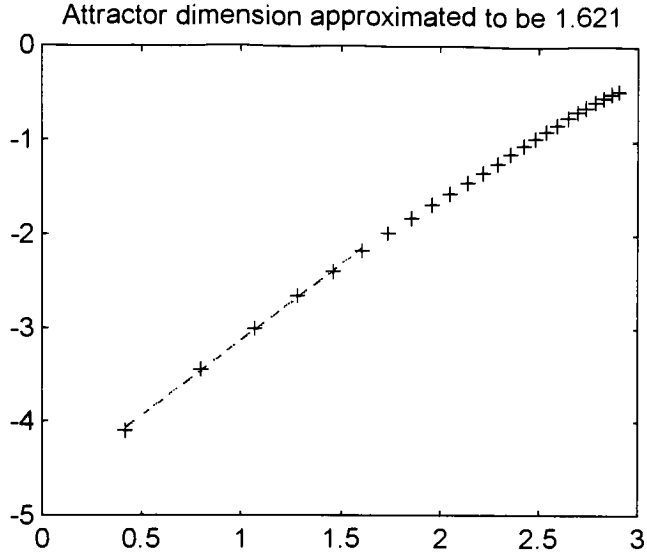
12:13:31.13
Elapsed time 2.064e+004 seconds.
Floating point operations = 3.862e+009.
normal termination

Time series analysis program by William Robertson.
30-Jun-94
16:45:57.67
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 2 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 38.
16:46:20.03
Elapsed time 22.57 seconds

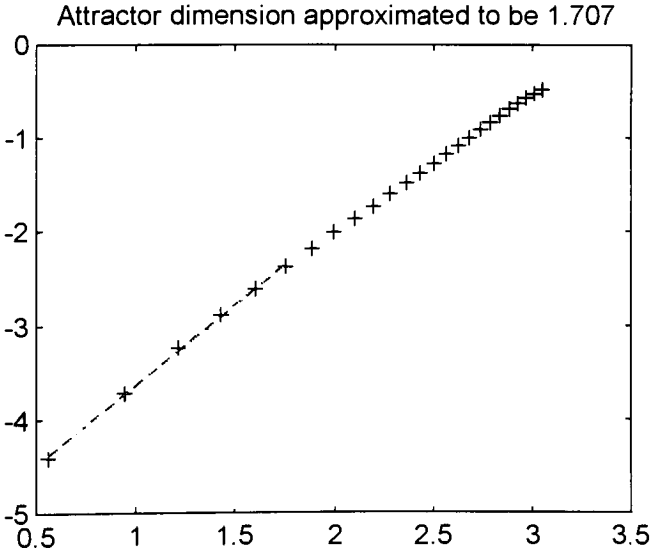
2-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:6:35.4
Elapsed time 4838 seconds.



3-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
19:32:42.79
Elapsed time 1.001e+004 seconds.



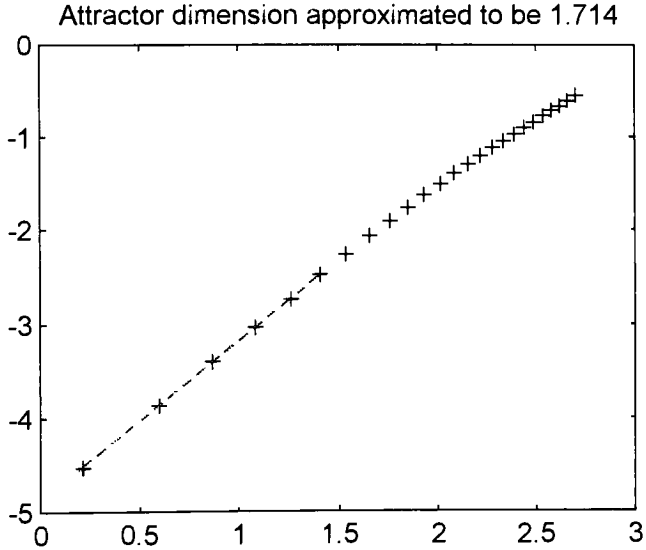
4-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4984
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
20:59:46.48
Elapsed time 1.523e+004 seconds.



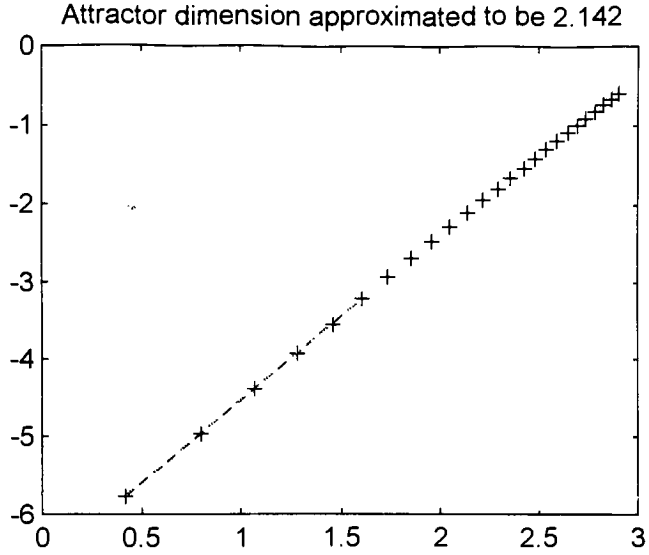
5-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4980
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

Time series analysis program by William Robertson.
30-Jun-94
21:24:38.81
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 3 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 29.
21:25:9.51
Elapsed time 31.2 seconds

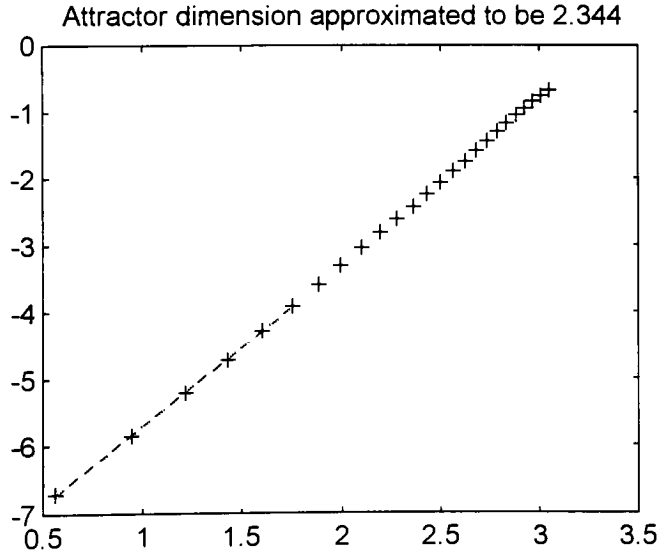
2-Dimensional embedding.
Using embedding delay of 29.
Number of embedded points = 4967
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
0:40:3.7
Elapsed time -2.58e+006 seconds.



3-Dimensional embedding.
Using embedding delay of 29.
Number of embedded points = 4938
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
3:59:5.27
Elapsed time -2.568e+006 seconds.

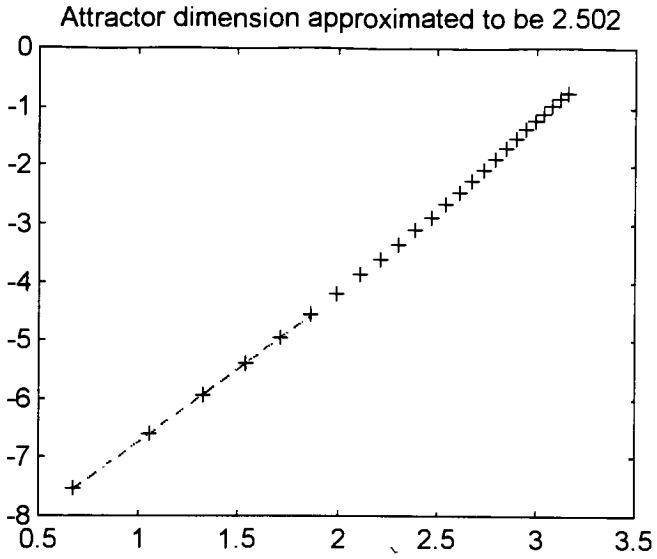


4-Dimensional embedding.
Using embedding delay of 29.
Number of embedded points = 4909
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
7:22:45.7
Elapsed time -2.556e+006 seconds.

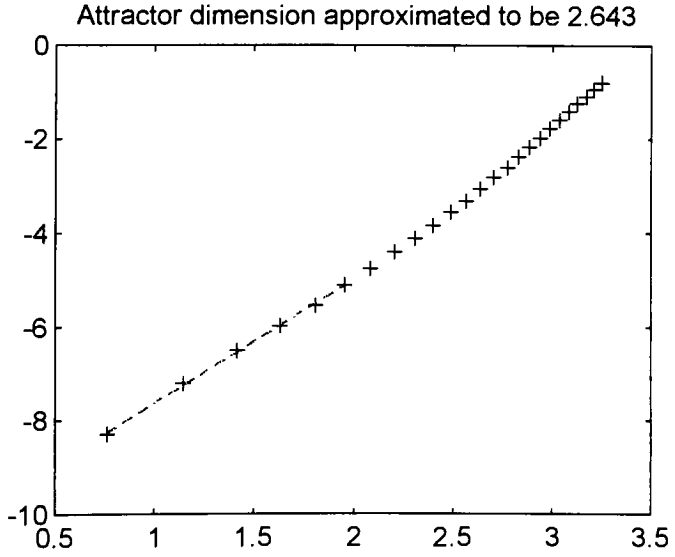


5-Dimensional embedding.
Using embedding delay of 29.
Number of embedded points = 4880
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

10:51:31.07
Elapsed time -2.544e+006 seconds.

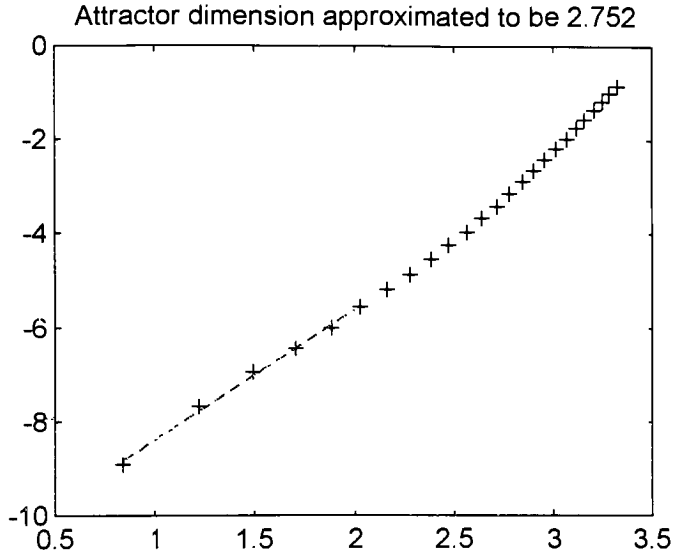


6-Dimensional embedding.
Using embedding delay of 29.
Number of embedded points = 4851
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
14:25:39.41
Elapsed time -2.531e+006 seconds.

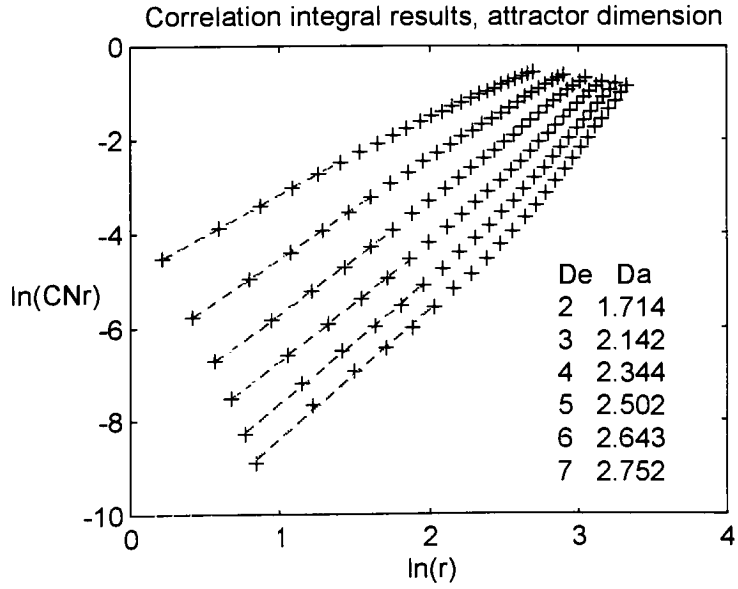


7-Dimensional embedding.
Using embedding delay of 29.
Number of embedded points = 4822
Using W factor of 5.

Finished correlation integral.
Finished LINREG.
18:4:48.08
Elapsed time -2.518e+006 seconds.
Embedding procedure complete.



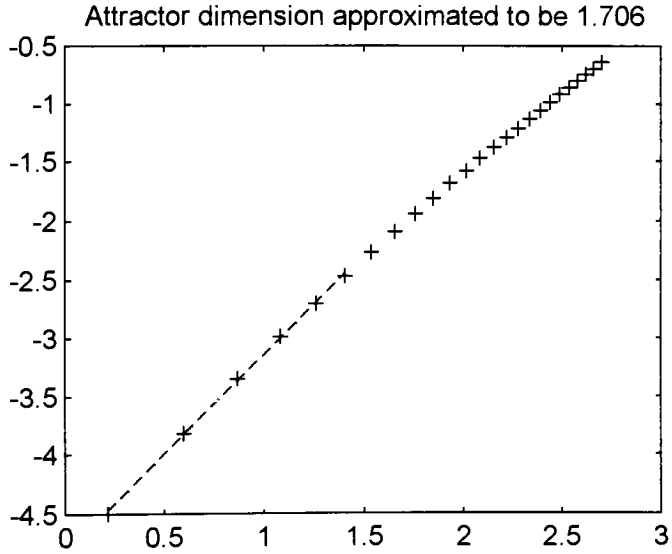
Embedding Dimension	Attractor Dimension
2	1.714
3	2.142
4	2.344
5	2.502
6	2.643
7	2.752



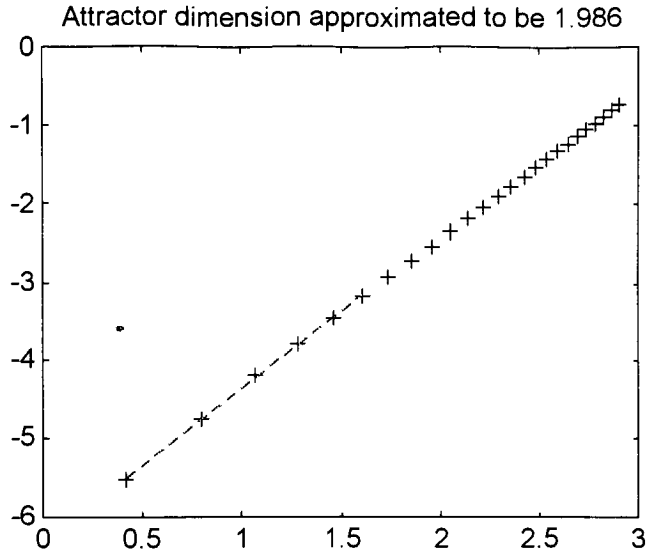
18:4:52.58
Elapsed time -2.518e+006 seconds.
Floating point operations = 6.162e+009.
normal termination

Time series analysis program by William Robertson.
30-Jun-94
22:38:37.28
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 3 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 29.
22:38:57.88
Elapsed time 24.88 seconds

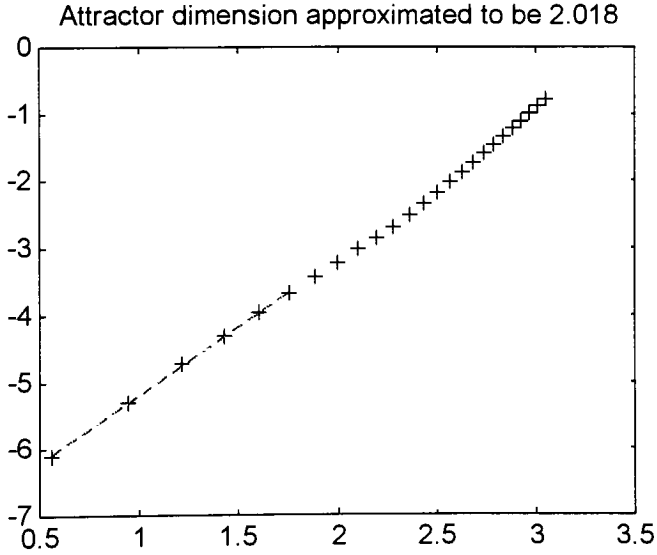
2-Dimensional embedding.
Using embedding delay of 15.
Number of cmbedded points = 4981
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
23:59:32.86
Elapsed time 4856 seconds.



3-Dimensional embedding.
Using embedding delay of 15.
Number of embedded points = 4966
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
1:24:47.52
Elapsed time -2.582e+006 seconds.

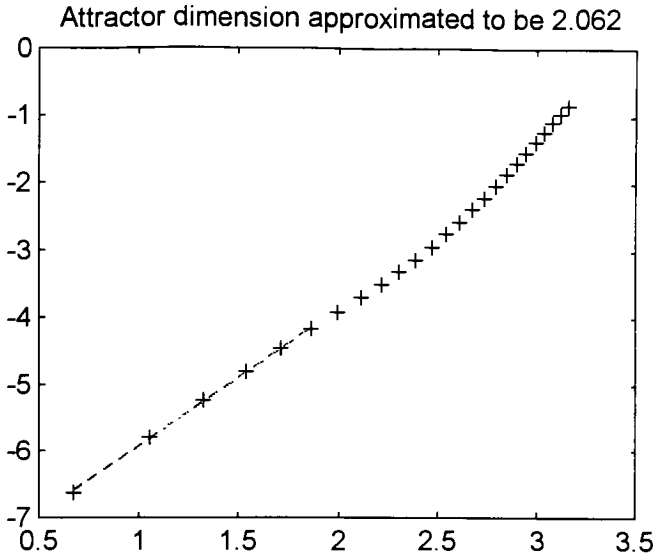


4-Dimensional embedding.
Using embedding delay of 15.
Number of embedded points = 4951
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
2:52:51.03
Elapsed time -2.577e+006 seconds.

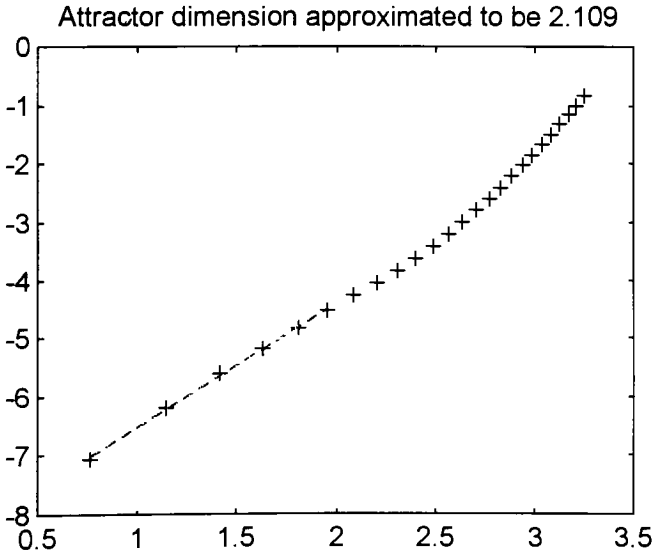


5-Dimensional embedding.
Using embedding delay of 15.
Number of embedded points = 4936
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

4:22:49.76
Elapsed time -2.571e+006 seconds.

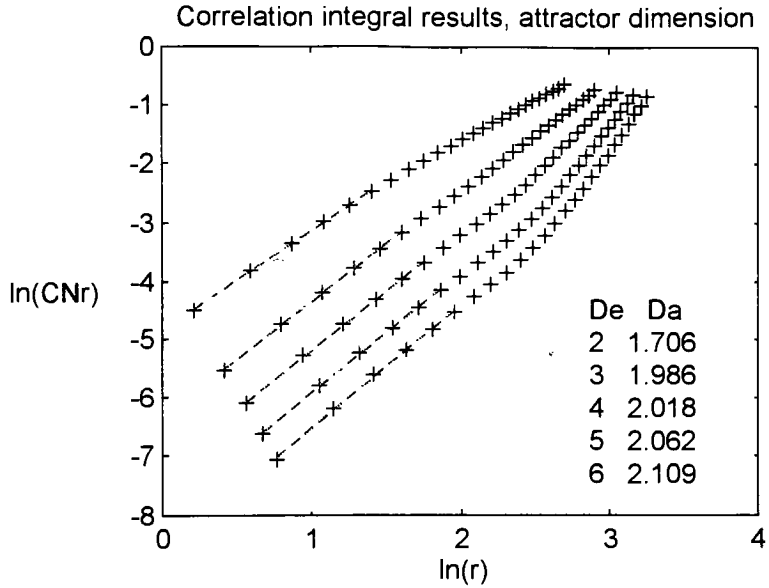


6-Dimensional embedding.
Using embedding delay of 15.
Number of embedded points = 4921
Using W factor of 4.
Finished correlation integral.
Finished LINREG.
5:57:7.25
Elapsed time -2.566e+006 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.706

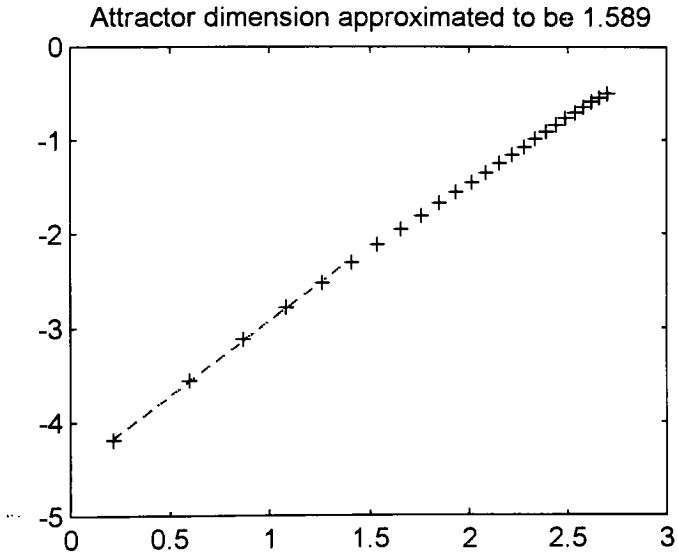
3	1.986
4	2.018
5	2.062
6	2.109



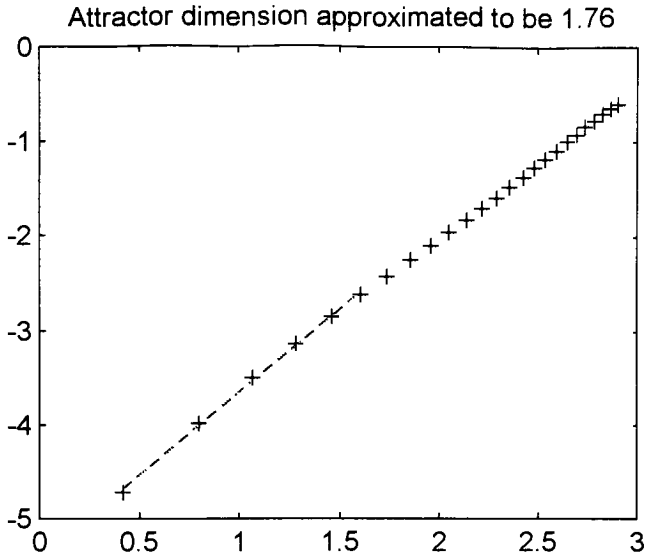
5:57:9.34
Elapsed time -2.566e+006 seconds.
Floating point operations = 5.02e+009.
normal termination

Time series analysis program by William Robertson.
1-Jul-94
6:26:35.42
Loading time series.
Read 300001 points from time series.
Using 5.000 points of time series, every 3 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 29.
6:26:55.74
Elapsed time 20.59 seconds

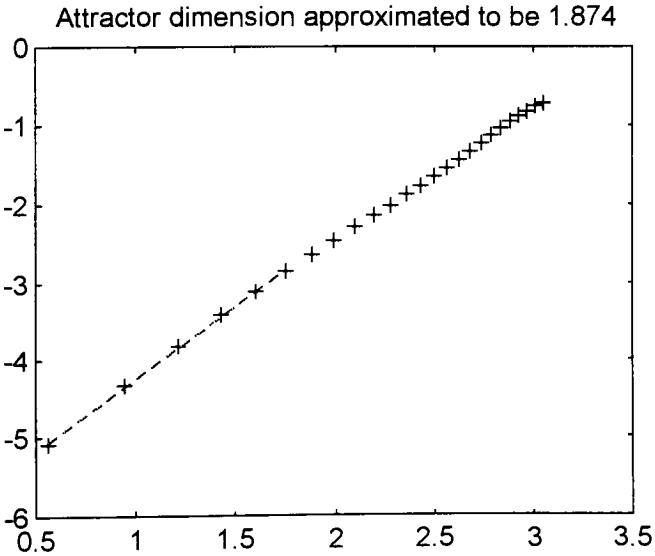
2-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
7:47:29.02
Elapsed time 4854 seconds.



3-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4984
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
9:11:25.35
Elapsed time 9890 seconds.

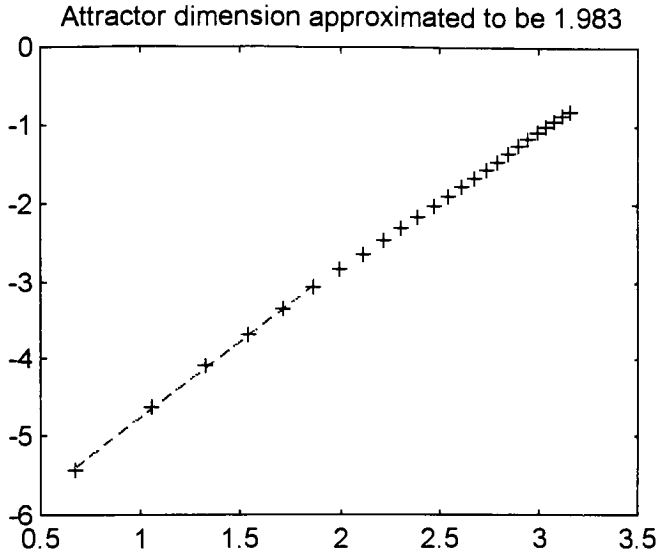


4-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4978
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
10:39:58.46
Elapsed time 1.52e+004 seconds.

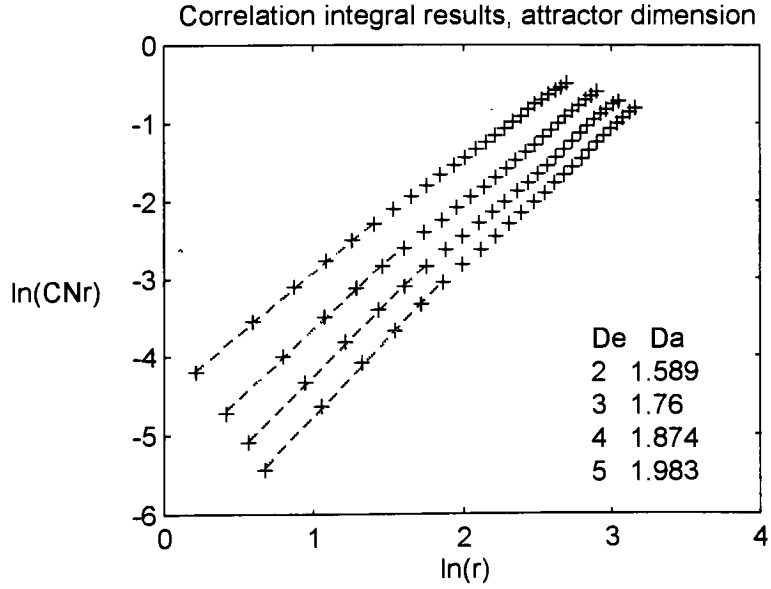


5-Dimensional embedding.
Using embedding delay of 6.
Number of embedded points = 4972
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

12:11:13.22
Elapsed time 2.068e+004 seconds.
Embedding procedure complete.



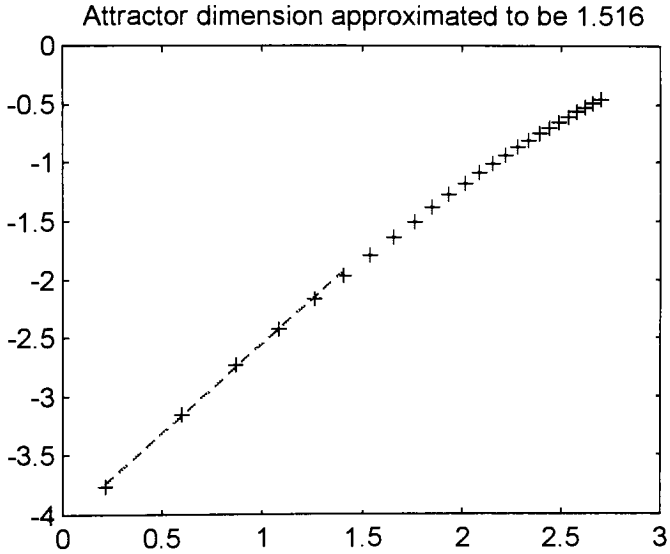
Embedding Dimension	Attractor Dimension
2	1.589
3	1.76
4	1.874
5	1.983



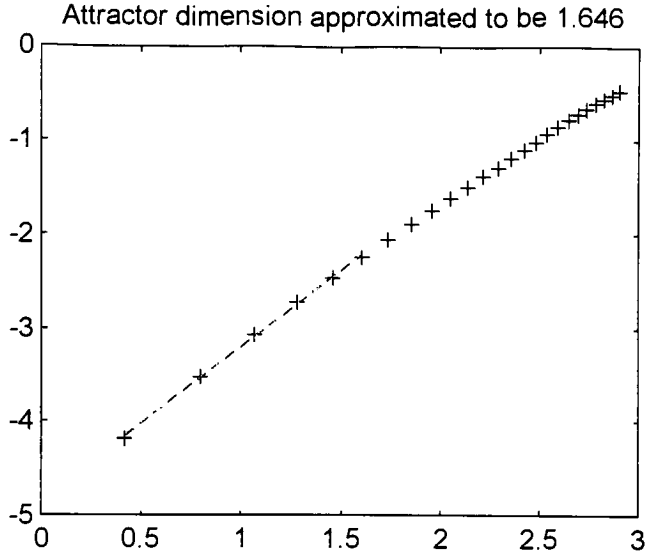
12:11:15.03
Elapsed time 2.068e+004 seconds.
Floating point operations = 3.87e+009.
normal termination

Time series analysis program by William Robertson.
1-Jul-94
19:54:59.84
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 3 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 29.
19:55:40.32
Elapsed time 40.97 seconds

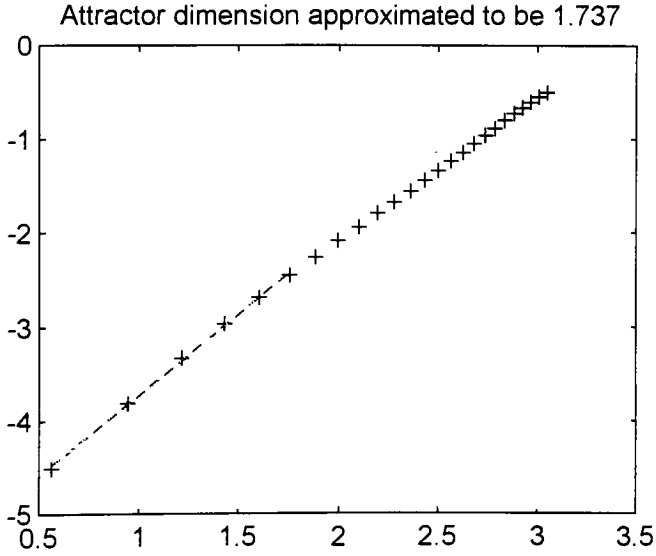
2-Dimensional embedding.
Using embedding delay of 3.
Number of embedded points = 4993
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
23:12:4.53
Elapsed time 1.182e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 3.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
2:34:23.78
Elapsed time 2.396e+004 seconds.

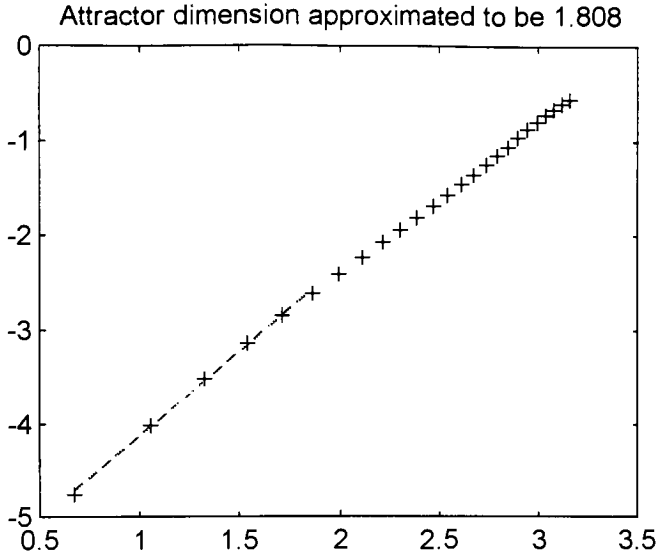


4-Dimensional embedding.
Using embedding delay of 3.
Number of embedded points = 4987
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
6:3:11.19
Elapsed time 3.649e+004 seconds.

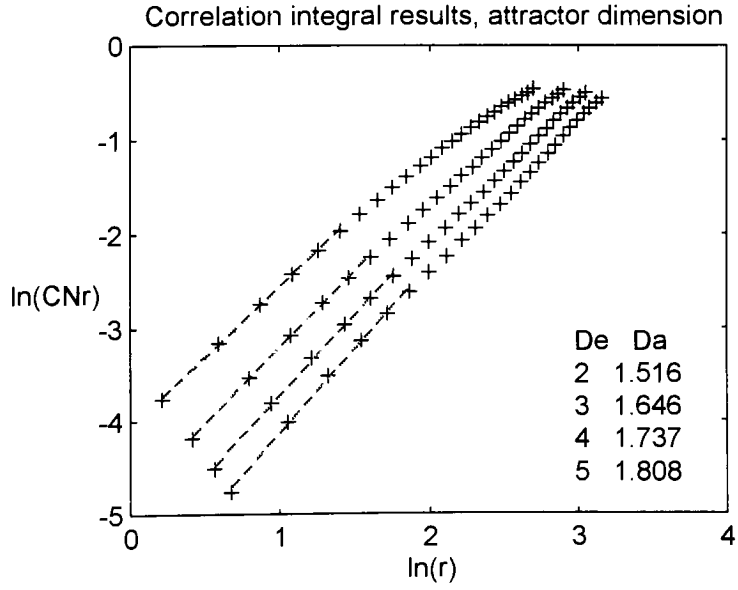


5-Dimensional embedding.
Using embedding delay of 3.
Number of embedded points = 4984
Using W factor of 3.
Finished correlation integral.
Finished LINREG.

9:39:35.63
Elapsed time 4.948e+004 seconds.
Embedding procedure complete.



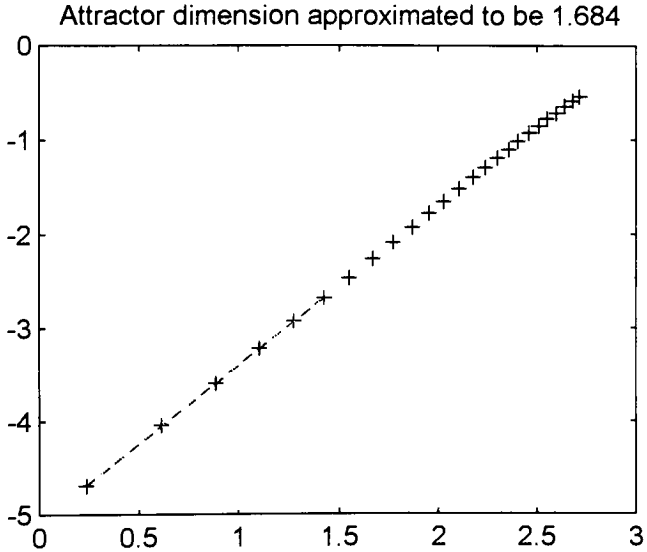
Embedding Dimension	Attractor Dimension
2	1.516
3	1.646
4	1.737
5	1.808



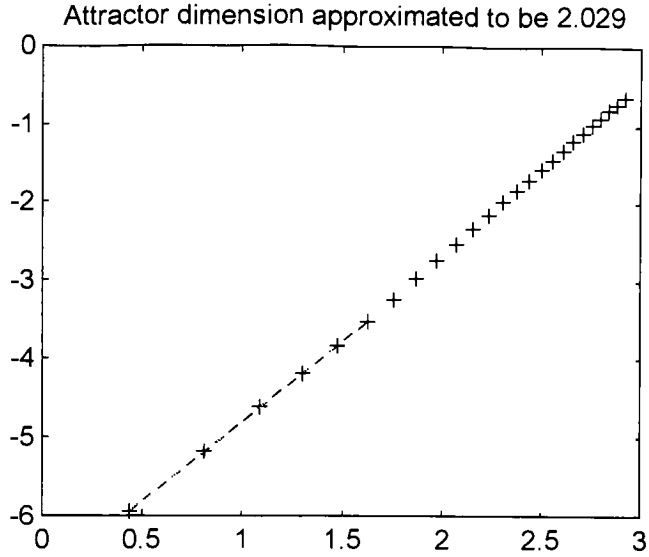
9:39:39.48
Elapsed time 4.948e+004 seconds.
Floating point operations = 3.883e+009.
normal termination

Time series analysis program by William Robertson.
2-Jul-94
17:12:30.73
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 5 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 19.
17:12:51.17
Elapsed time 20.71 seconds

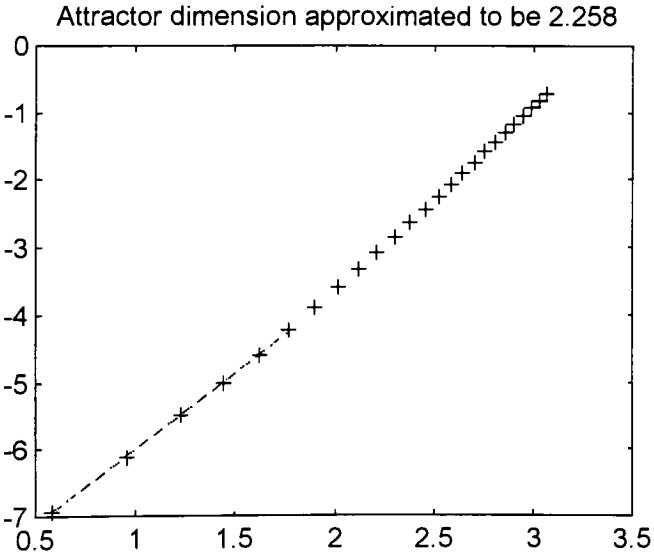
2-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4977
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:32:50.61
Elapsed time 4820 seconds.



3-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4958
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
19:56:17.01
Elapsed time 9826 seconds.

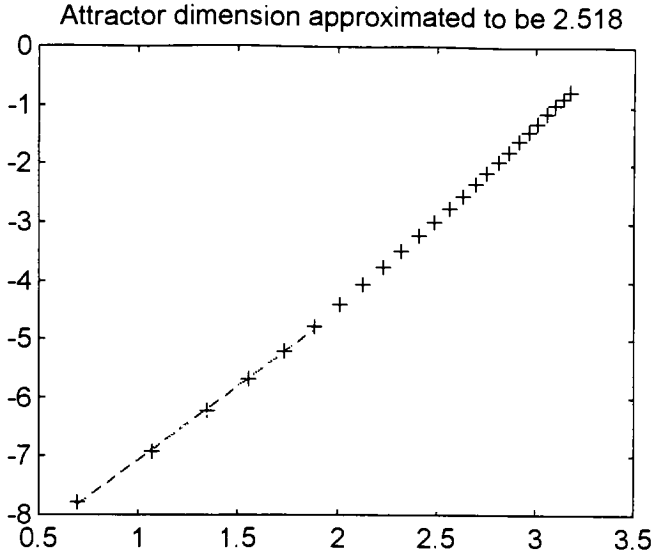


4-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4939
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
21:22:39.07
Elapsed time 1.501e+004 seconds.

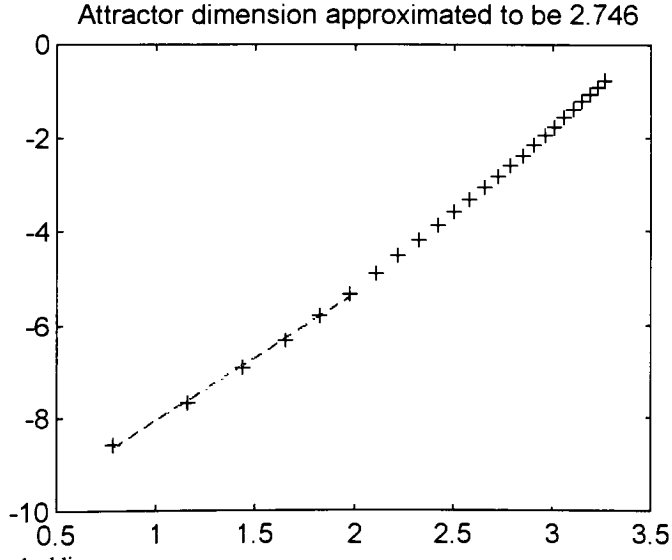


5-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4920
Using W factor of 2.
Finished correlation integral.
Finished LINREG.

22:54:33.75
Elapsed time 2.052e+004 seconds.

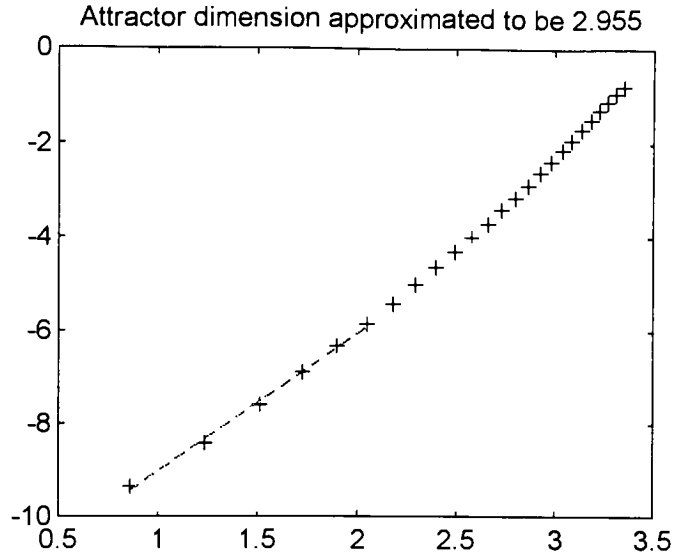


6-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4901
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
0:27:28.14
Elapsed time 2.61e+004 seconds.

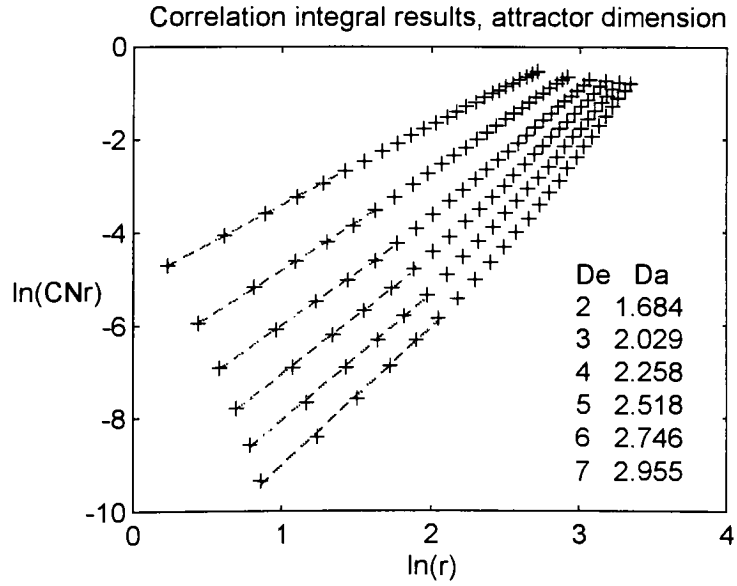


7-Dimensional embedding.
Using embedding delay of 19.
Number of embedded points = 4882
Using W factor of 4.
Finished correlation integral.

Finished LINREG.
2.3:2.64
Elapsed time 3.183e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.684
3	2.029
4	2.258
5	2.518
6	2.746
7	2.955

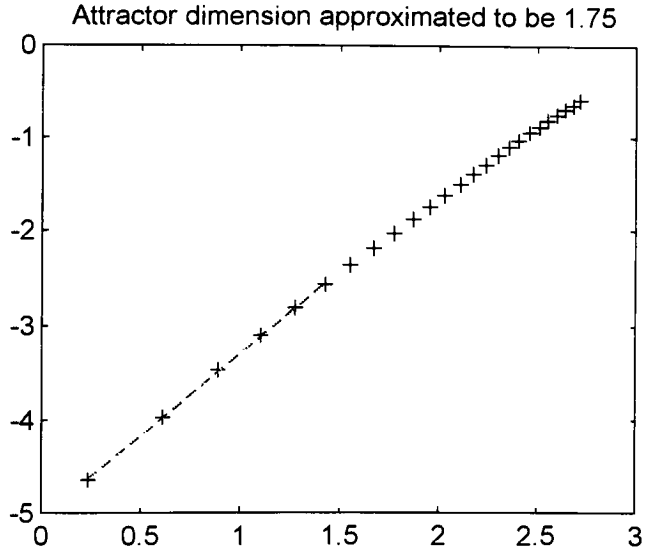


2:3:4.84

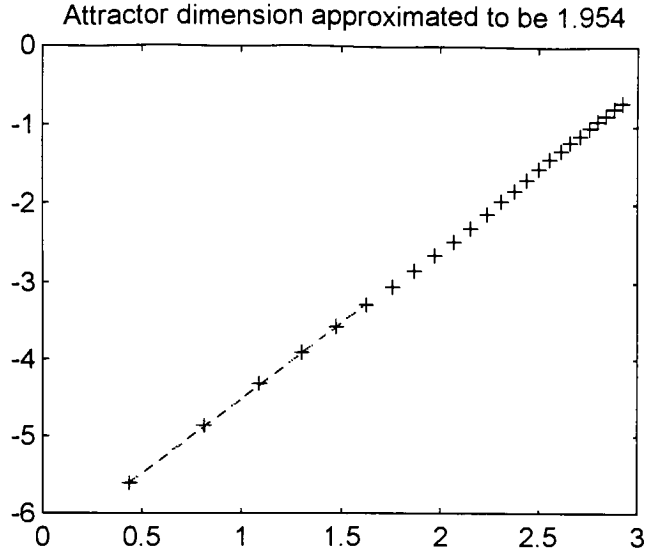
Elapsed time 3.183e+004 seconds.
Floating point operations = 6.257e+009.
normal termination

Time series analysis program by William Robertson.
3-Jul-94
9:39:48.21
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 5 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 19.
9:40:8.59
Elapsed time 20.6 seconds

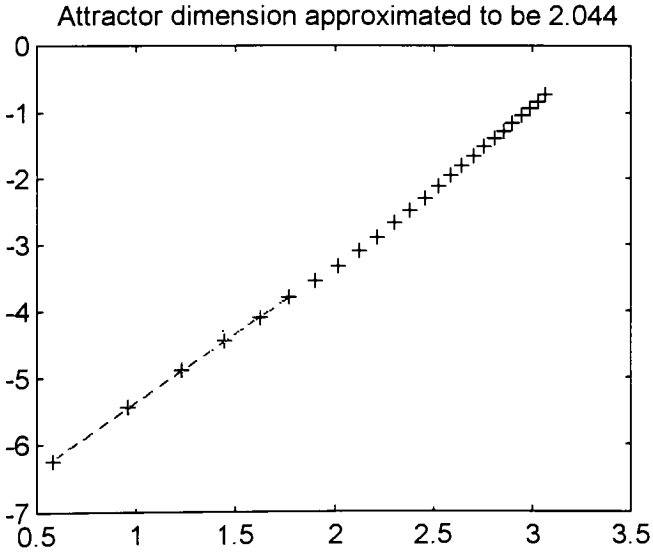
2-Dimensional embedding.
Using embedding delay of 10.
Number of embedded points = 4986
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
11:0:35.11
Elapsed time 4847 seconds.



3-Dimensional embedding.
Using embedding delay of 10.
Number of embedded points = 4976
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:24:36.72
Elapsed time 9889 seconds.

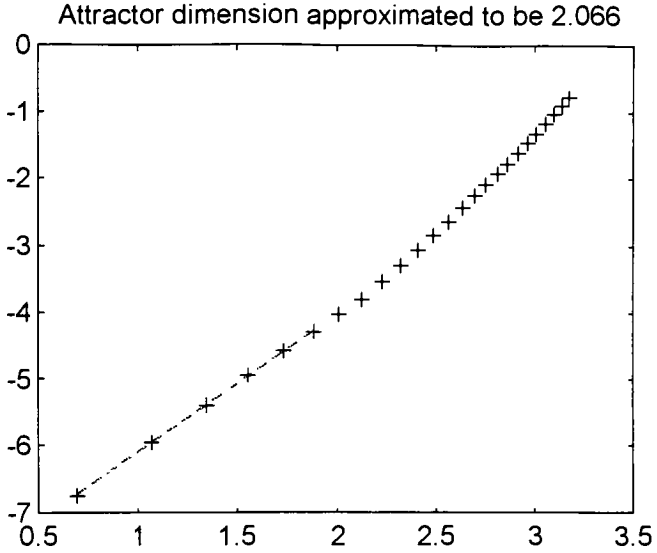


4-Dimensional embedding.
Using embedding delay of 10.
Number of embedded points = 4966
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
13:51:52.38
Elapsed time 1.512e+004 seconds.

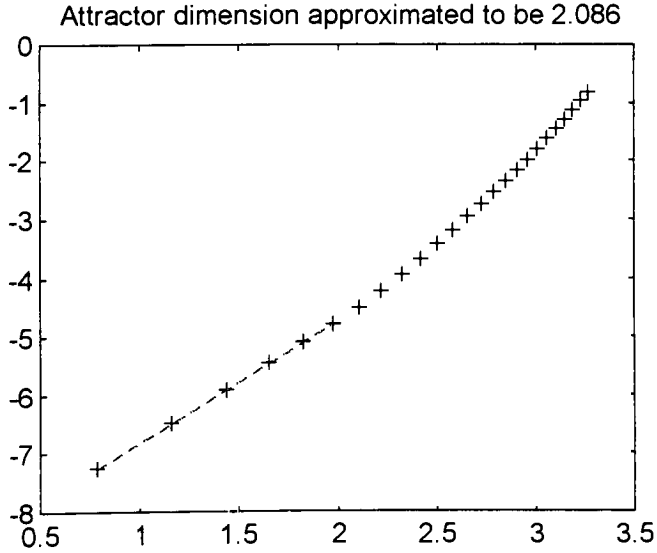


5-Dimensional embedding.
Using embedding delay of 10.
Number of embedded points = 4956
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:24:52.1

Elapsed time 2.07e+004 seconds.

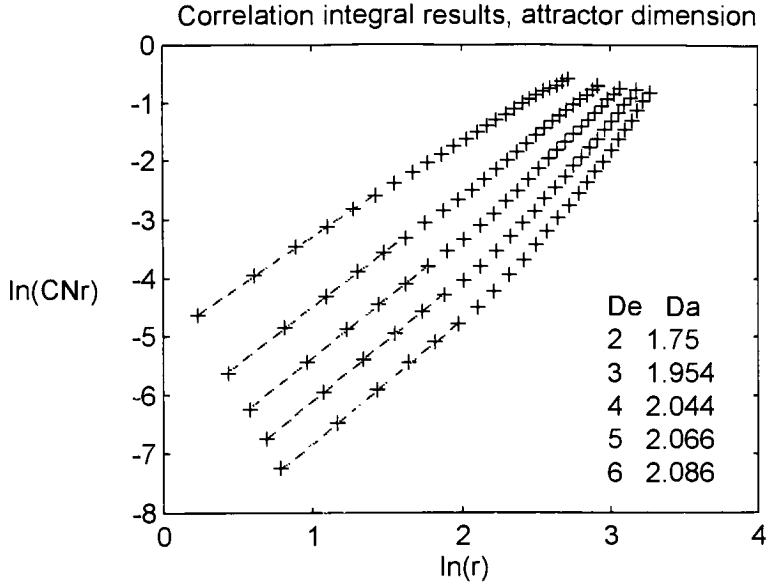


6-Dimensional embedding.
Using embedding delay of 10.
Number of embedded points = 4946
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
16:59:39.14
Elapsed time 2.639e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.75
3	1.954
4	2.044

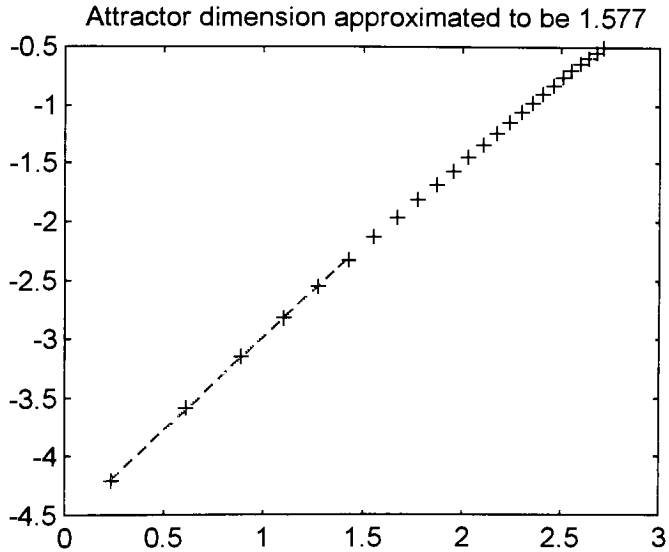
5 2.066
6 2.086



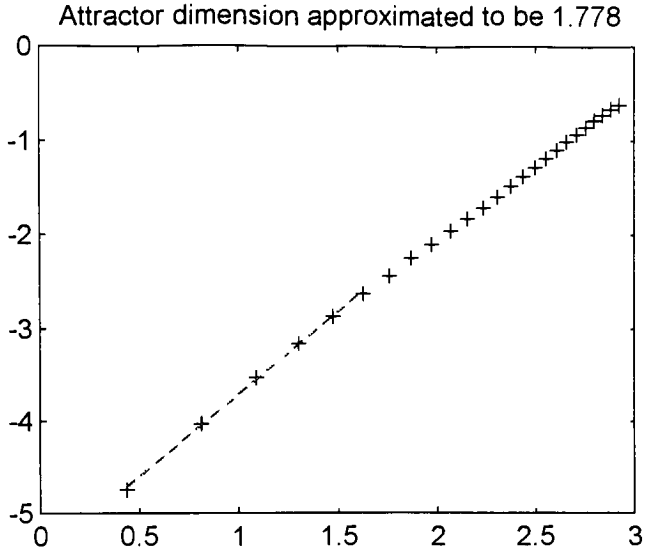
16:59:41.23
Elapsed time 2.639e+004 seconds.
Floating point operations = 5.053e+009.
normal termination

Time series analysis program by William Robertson.
2-Jul-94
9:41:53.11
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 5 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 19.
9:42:13.71
Elapsed time 20.87 seconds

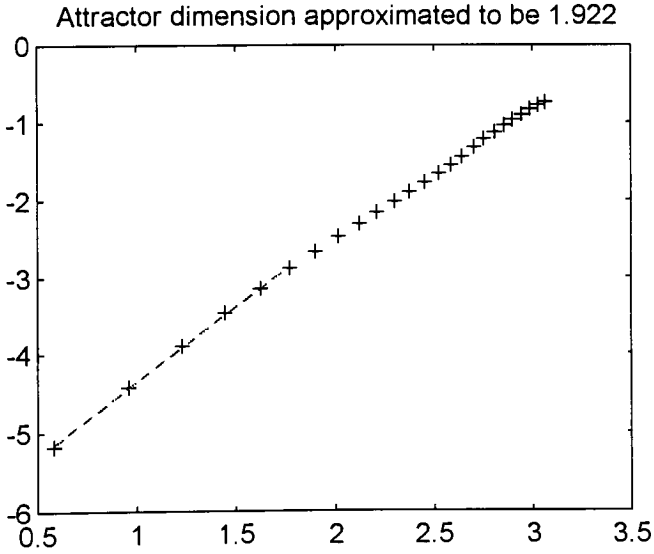
2-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
11:2:37.04
Elapsed time 4844 seconds.



3-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:26:50.85
Elapsed time 9898 seconds.

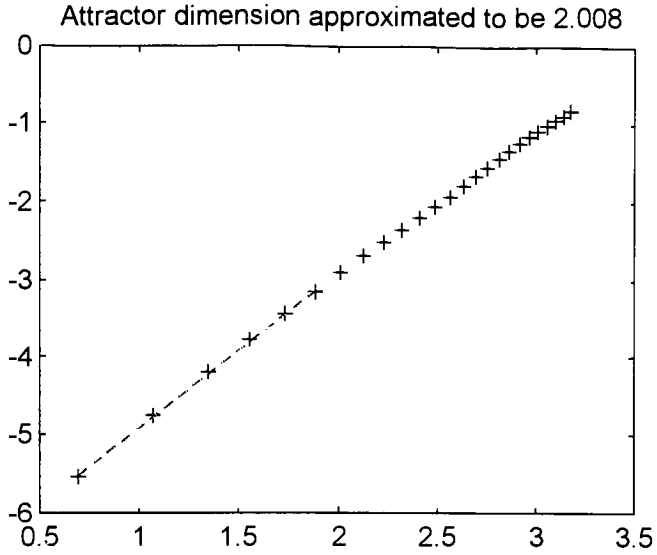


4-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4984
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
13:54:29.14
Elapsed time 1.516e+004 seconds.

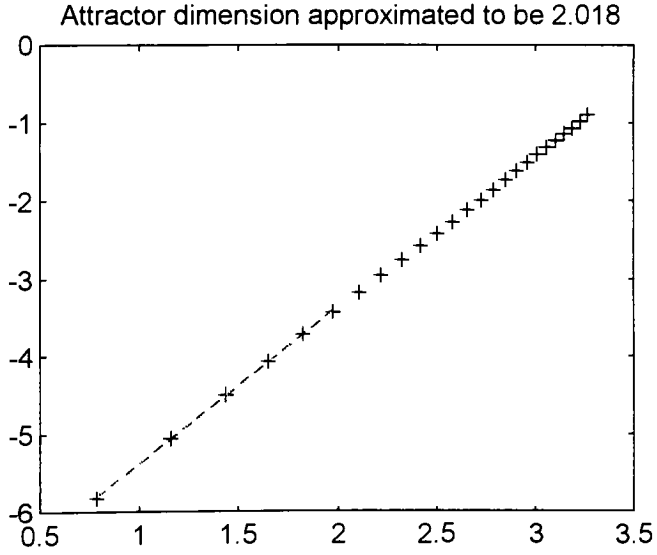


5-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4980
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:27:56.48

Elapsed time 2.076e+004 seconds.

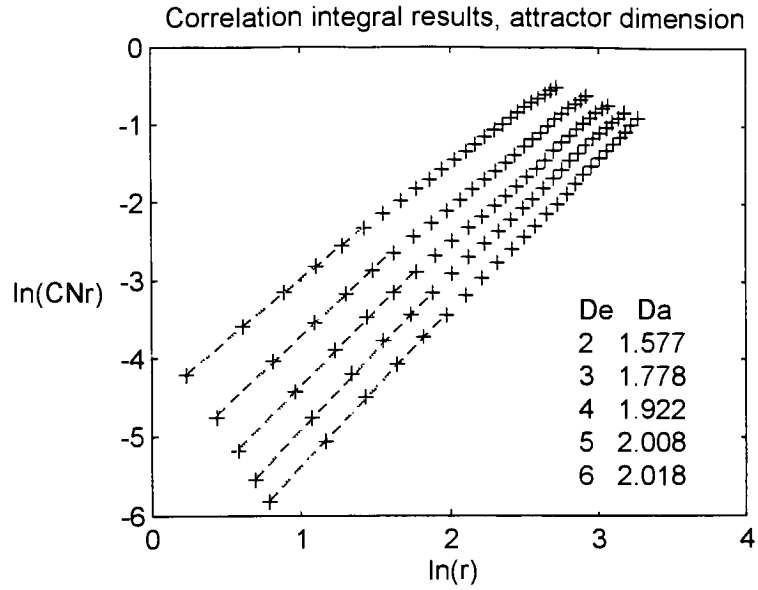


6-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4976
Using W factor of 3.
Finished correlation integral.
Finished LINREG.
17:3:27.41
Elapsed time 2.649e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.577
3	1.778
4	1.922

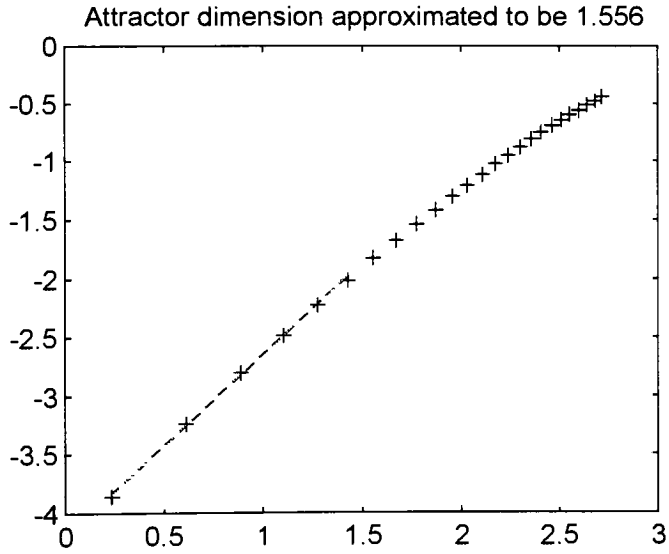
5 2.008
6 2.018



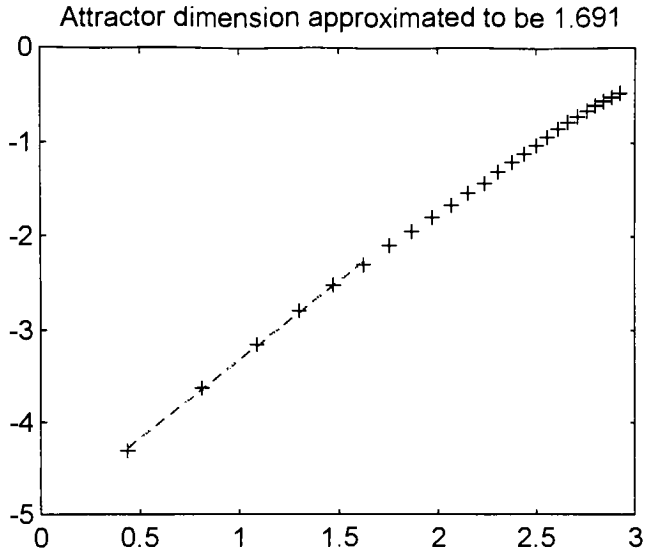
17:3:29.44
Elapsed time 2.65e+004 seconds.
Floating point operations = 5.092e+009.
normal termination

Time series analysis program by William Robertson.
2-Jul-94
10:12:26.08
Loading time series.
Read 300001 points from time series.
Using 5.000 points of time series, every 5 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 19.
10:12:56.51
Elapsed time 30.87 seconds

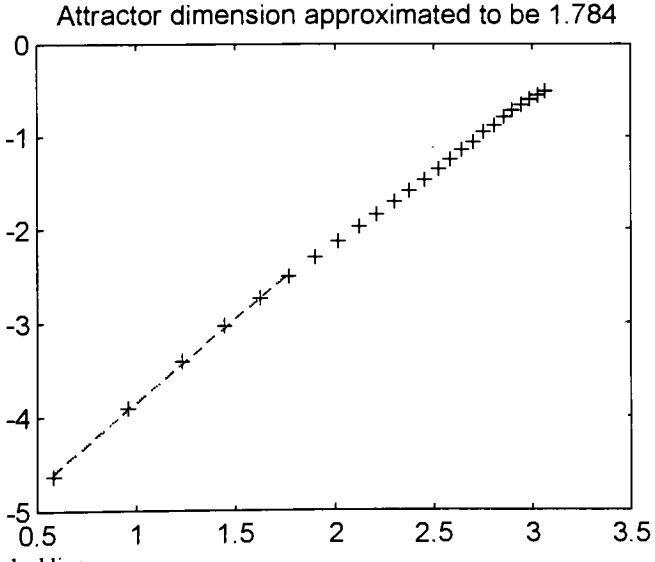
2-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
13:29:24.24
Elapsed time 1.182e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
16:52:11.99
Elapsed time 2.399e+004 seconds.

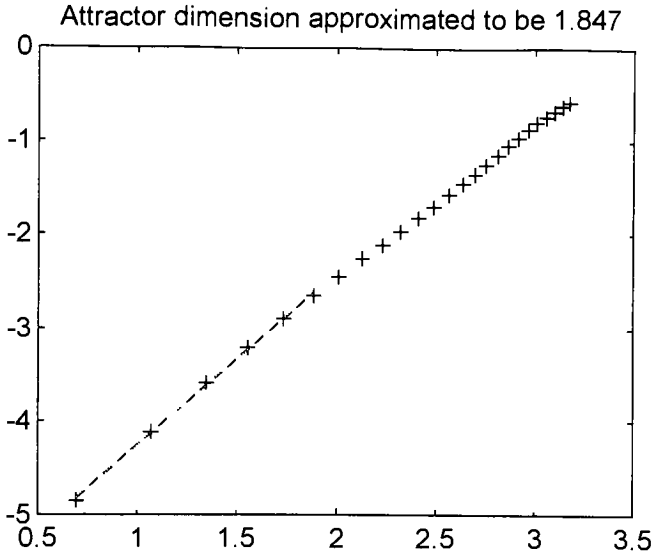


4-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
20:21:17.14
Elapsed time 3.653e+004 seconds.

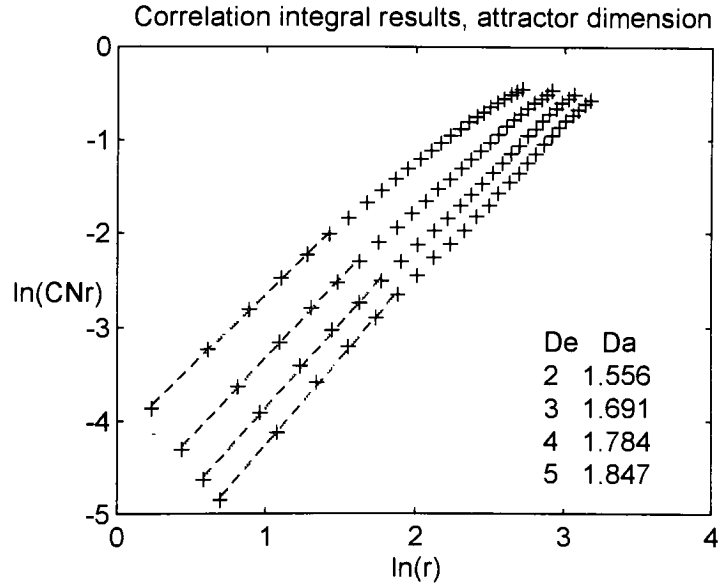


5-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
23:58:13.99

Elapsed time 4.955e+004 seconds.
Embedding procedure complete.



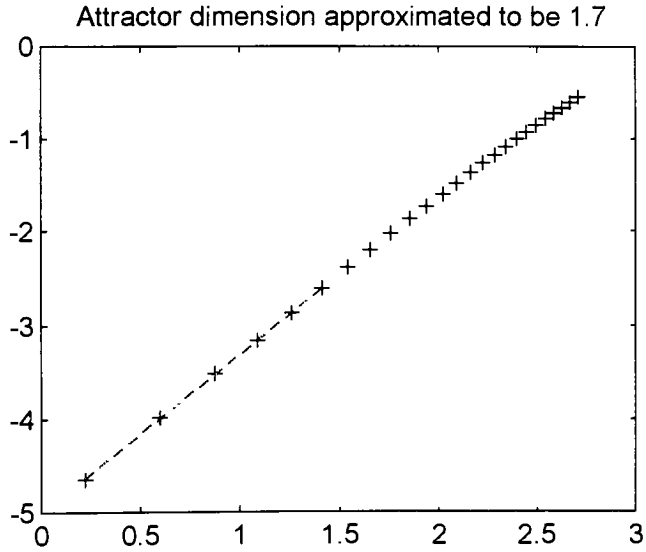
Embedding Dimension	Attractor Dimension
2	1.556
3	1.691
4	1.784
5	1.847



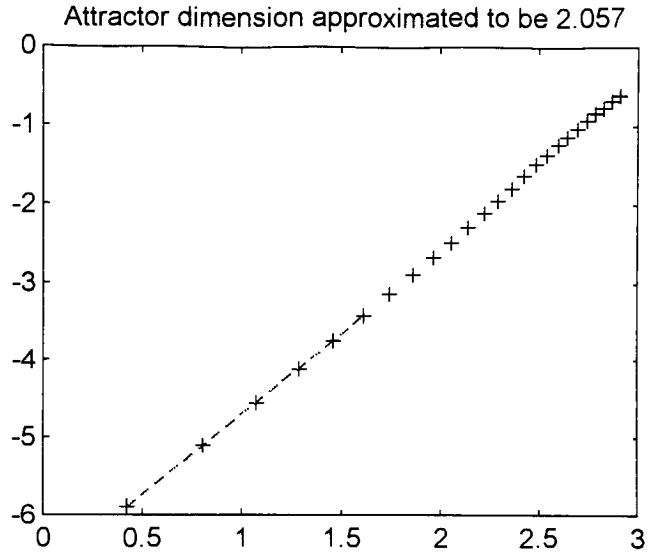
23:58:17.89
Elapsed time 4.955e+004 seconds.
Floating point operations = 3.887e+009.
normal termination

Time series analysis program by William Robertson.
3-Jul-94
17:55:48.6
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 10 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 9.
17:56:8.92
Elapsed time 20.6 seconds

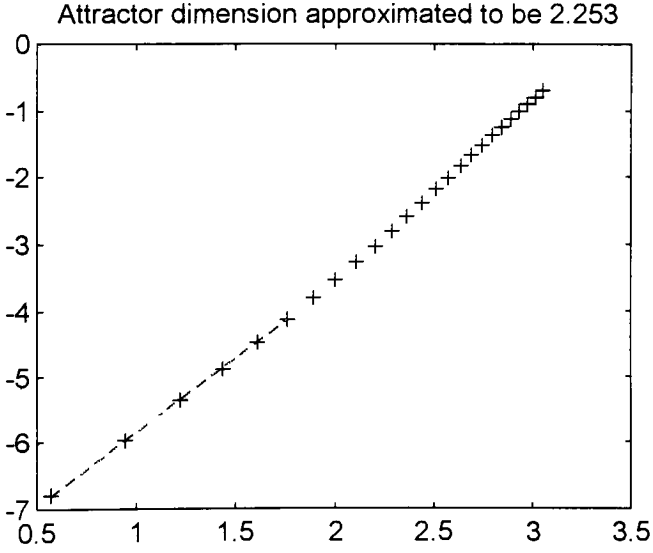
2-Dimensional embedding.
Using embedding delay of 9.
Number of embedded points = 4987
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
19:16:33.41
Elapsed time 4845 seconds.



3-Dimensional embedding.
Using embedding delay of 9.
Number of embedded points = 4978
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
20:40:39.86
Elapsed time 9891 seconds.

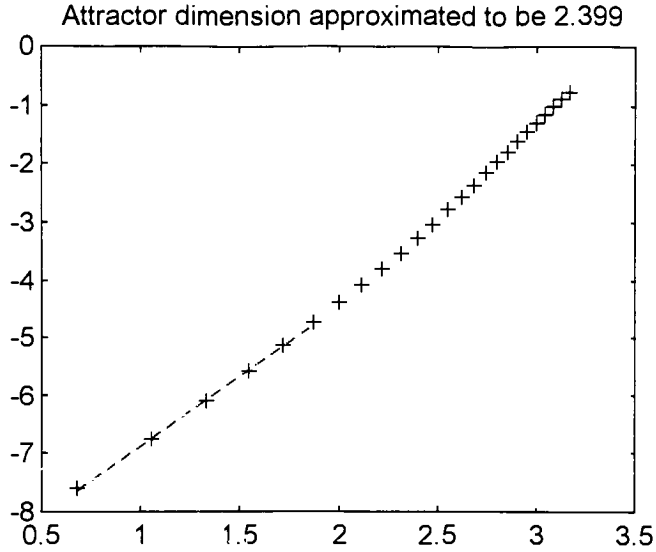


4-Dimensional embedding.
Using embedding delay of 9.
Number of embedded points = 4969
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
22:8:3.98
Elapsed time 1.514e+004 seconds.

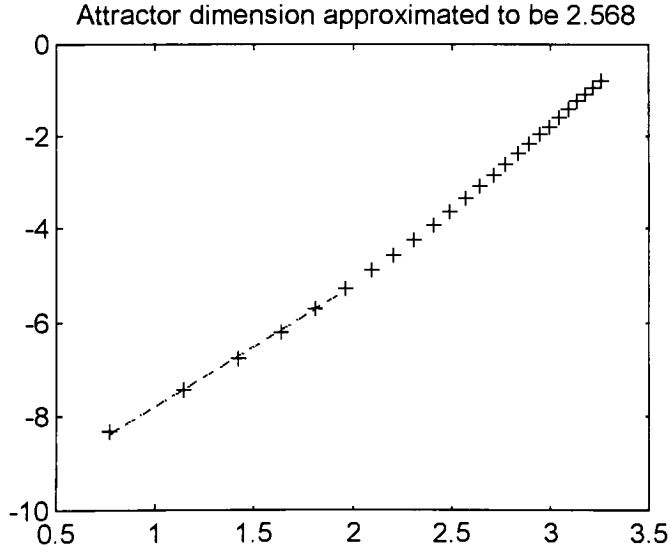


5-Dimensional embedding.
Using embedding delay of 9.
Number of embedded points = 4960
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
23:41:12.26

Elapsed time 2.072e+004 seconds.

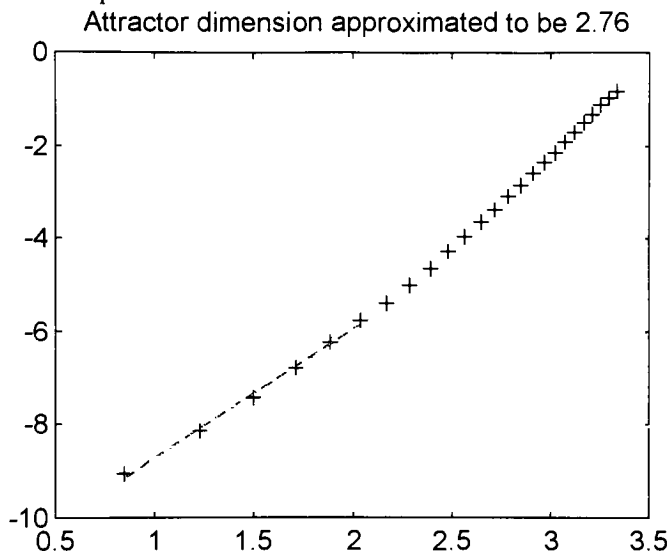


6-Dimensional embedding.
Using embedding delay of 9.
Number of embedded points = 4951
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
1:16:21.61
Elapsed time 2.643e+004 seconds.

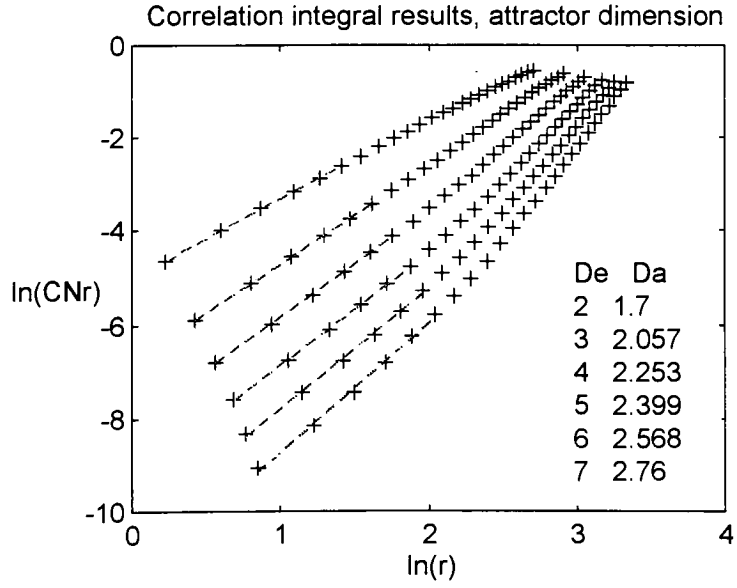


7-Dimensional embedding.
Using embedding delay of 9.
Number of embedded points = 4942
Using W factor of 2.
Finished correlation integral.
Finished LINREG.

2:54:15.23
Elapsed time 3.231e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.7
3	2.057
4	2.253
5	2.399
6	2.568
7	2.76

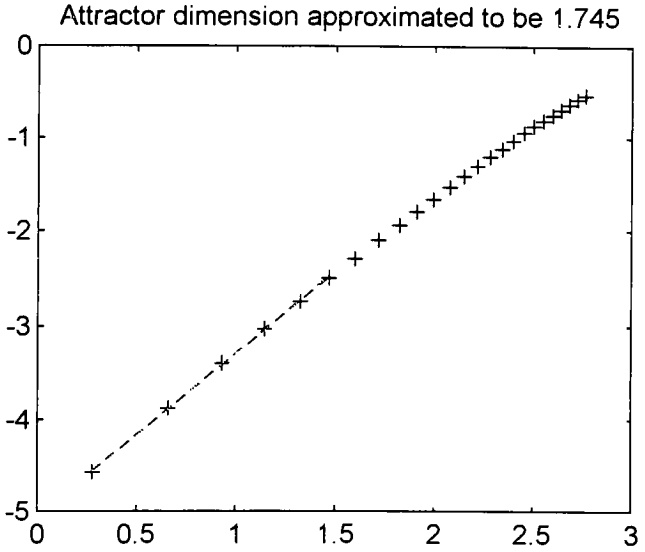


2:54:17.37
Elapsed time 3.231e+004 seconds.

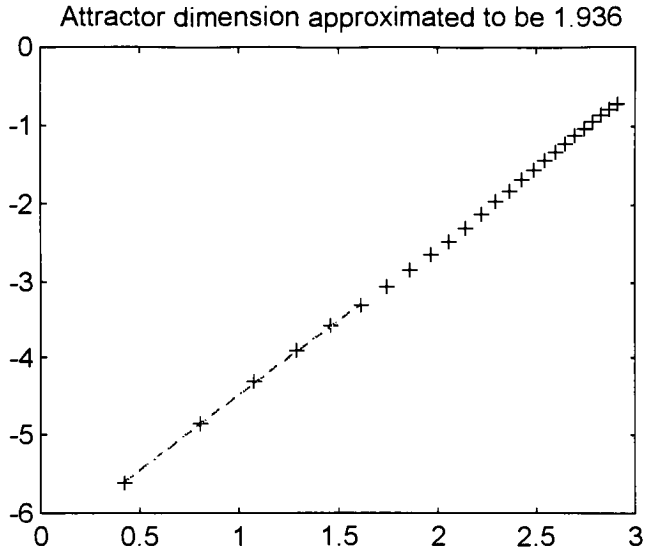
Floating point operations = 6.353e+009.
normal termination

Time series analysis program by William Robertson.
4-Jul-94
8:47:1.15
Loading time series.
Read 300001 points from time series.
Using 5.000 points of time series, every 10 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 9.
8:47:21.53
Elapsed time 20.59 seconds

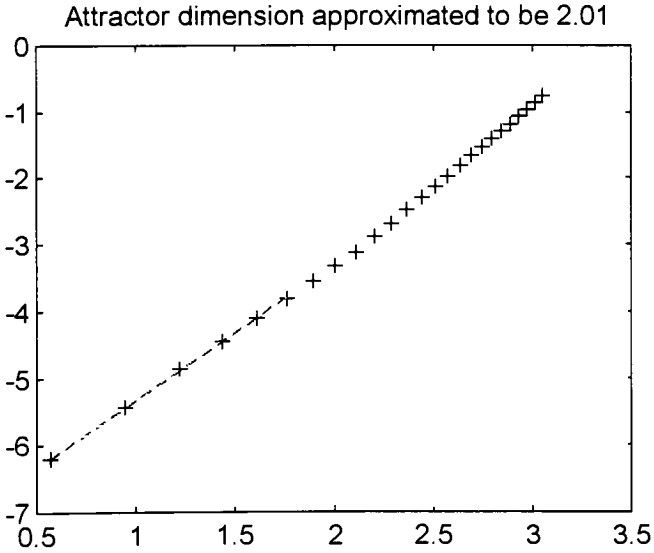
2-Dimensional embedding.
Using embedding delay of 5.
Number of embedded points = 4991
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
10:7:47.67
Elapsed time 4847 seconds.



3-Dimensional embedding.
Using embedding delay of 5.
Number of embedded points = 4986
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
11:32:9.22
Elapsed time 9908 seconds.

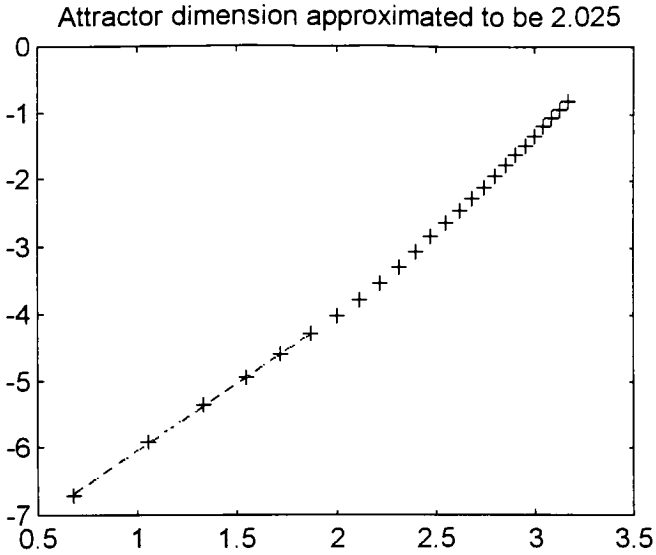


4-Dimensional embedding.
Using embedding delay of 5.
Number of embedded points = 4981
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:59:56.41
Elapsed time 1.518e+004 seconds.

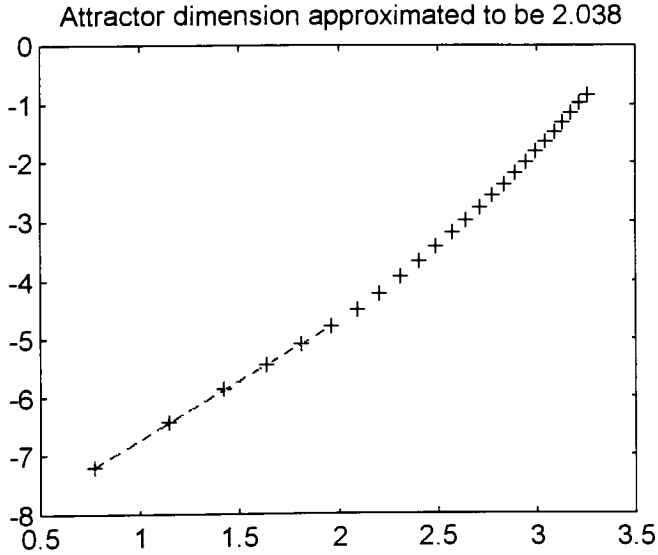


5-Dimensional embedding.
Using embedding delay of 5.
Number of embedded points = 4976
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
14:33:27.65

Elapsed time 2.079e+004 seconds.

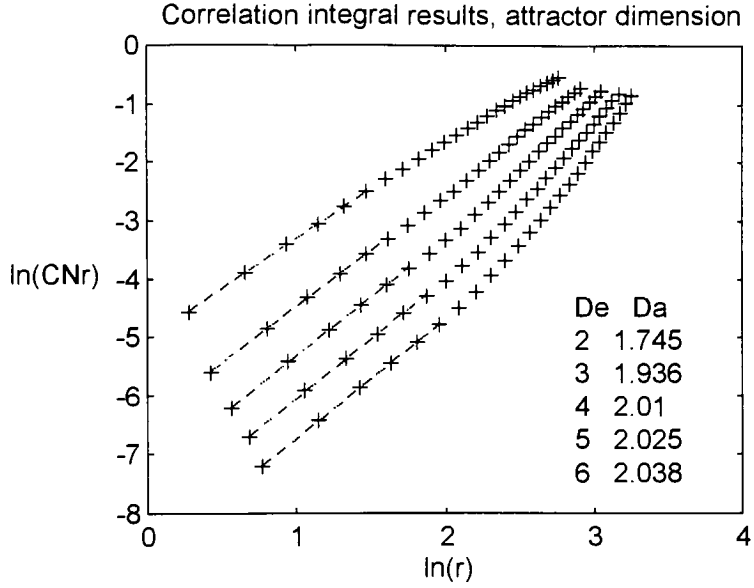


6-Dimensional embedding.
Using embedding delay of 5.
Number of embedded points = 4971
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
16:9:36.03
Elapsed time 2.655e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.745
3	1.936
4	2.01

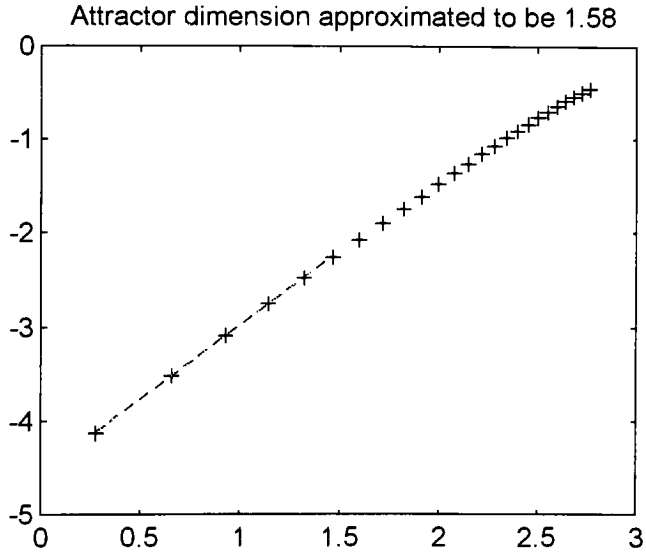
5 2.025
6 2.038



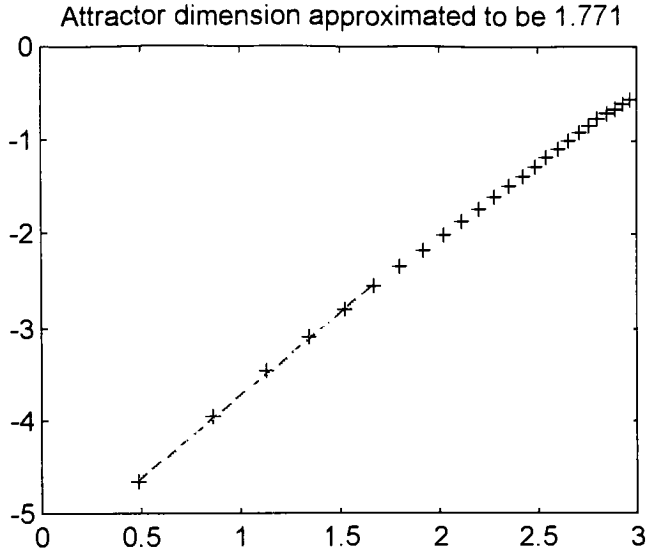
16:9:38.07
Elapsed time 2.656e+004 seconds.
Floating point operations = 5.085e+009.
normal termination

Time series analysis program by William Robertson.
4-Jul-94
16:49:10.35
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 10 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 9.
16:49:30.73
Elapsed time 20.66 seconds

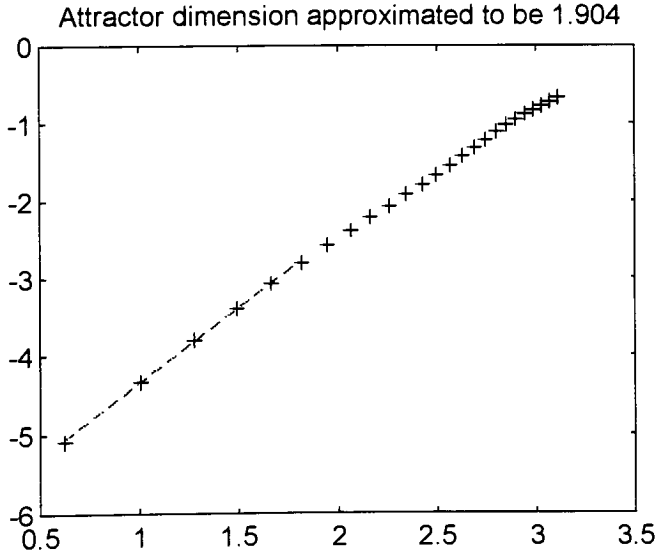
2-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:9:55.22
Elapsed time 4845 seconds.



3-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
19:34:16.55
Elapsed time 9906 seconds.

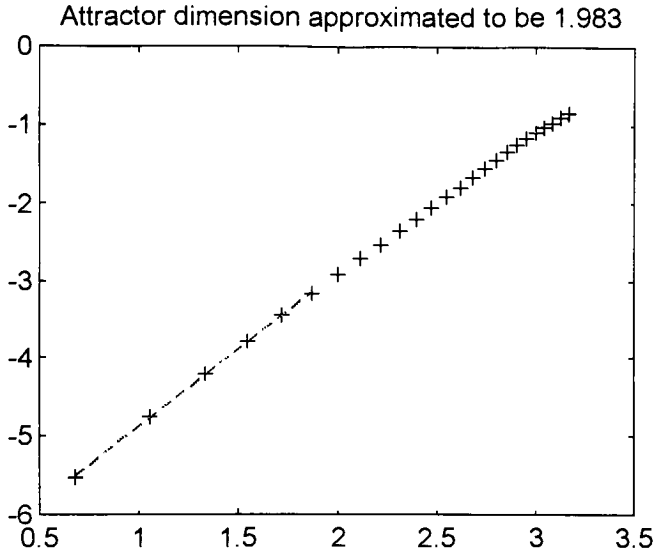


4-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
21:2:11.54
Elapsed time 1.518e+004 seconds.

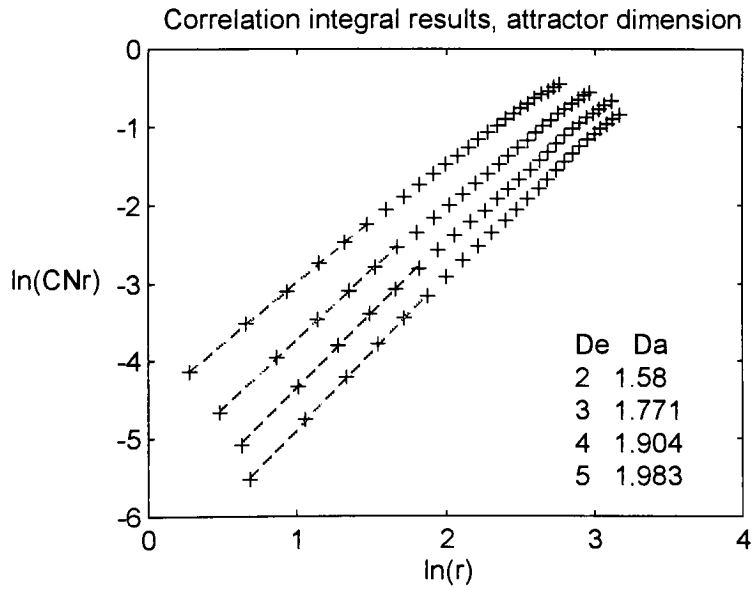


5-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
22:35:56.46

Elapsed time 2.081e+004 seconds.
Embedding procedure complete.



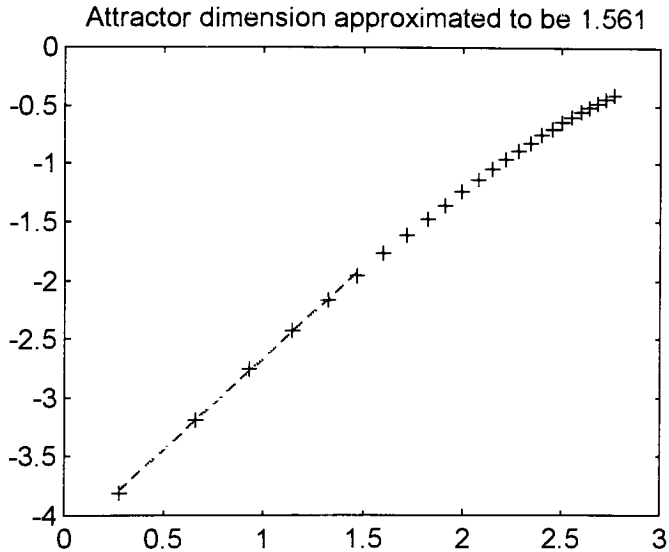
Embedding Dimension	Attractor Dimension
2	1.58
3	1.771
4	1.904
5	1.983



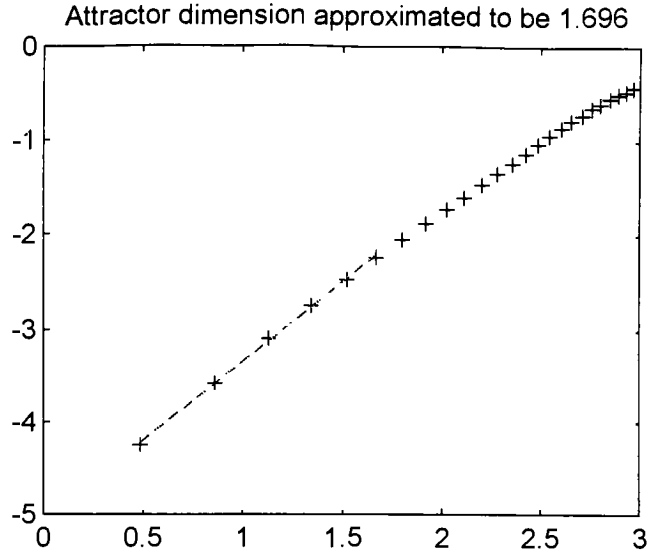
22:35:58.38
Elapsed time 2.081e+004 seconds.
Floating point operations = 3.887e+009.
normal termination

Time series analysis program by William Robertson.
3-Jul-94
9:13:12.84
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 10 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 9.
9:13:43.38
Elapsed time 31.03 seconds

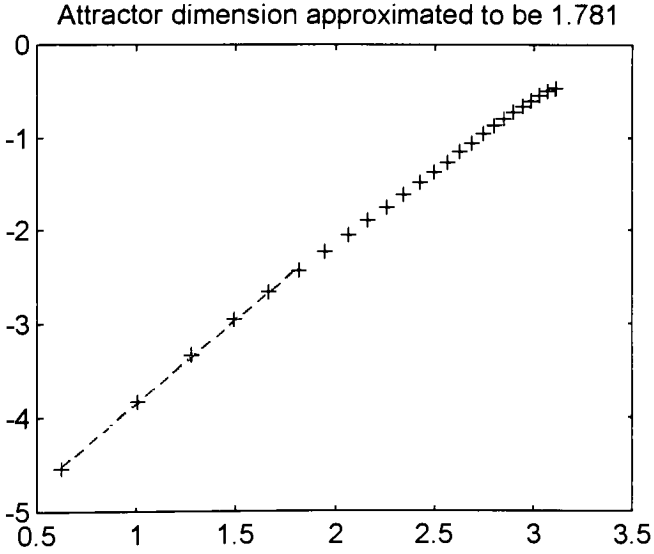
2-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4995
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:29:55.07
Elapsed time 1.18e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:52:39.2
Elapsed time 2.397e+004 seconds.

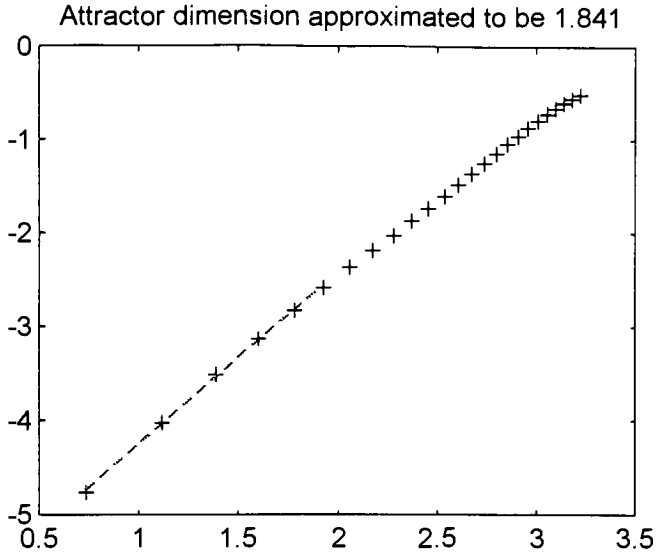


4-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4993
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
19:21:59.12
Elapsed time 3.653e+004 seconds.

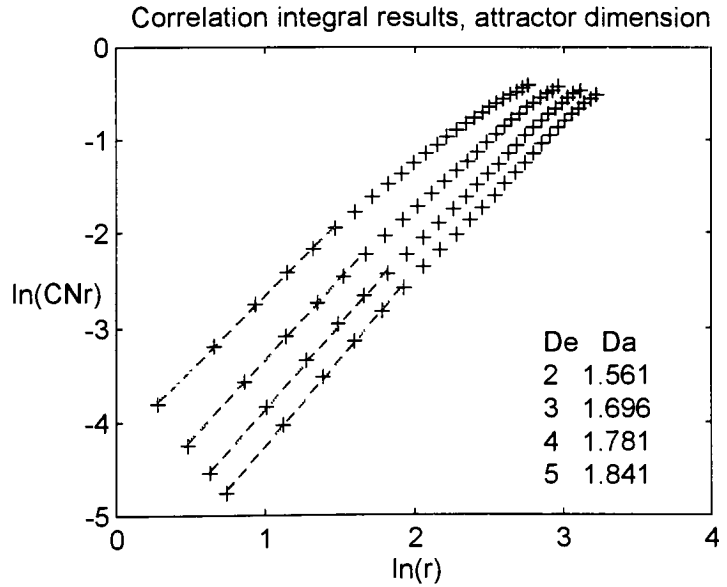


5-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
22:58:19.06

Elapsed time 4.951e+004 seconds.
Embedding procedure complete.



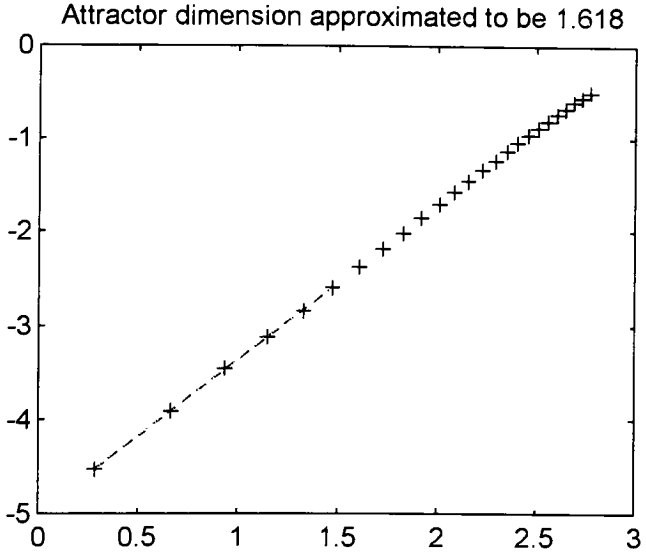
Embedding Dimension	Attractor Dimension
2	1.561
3	1.696
4	1.781
5	1.841



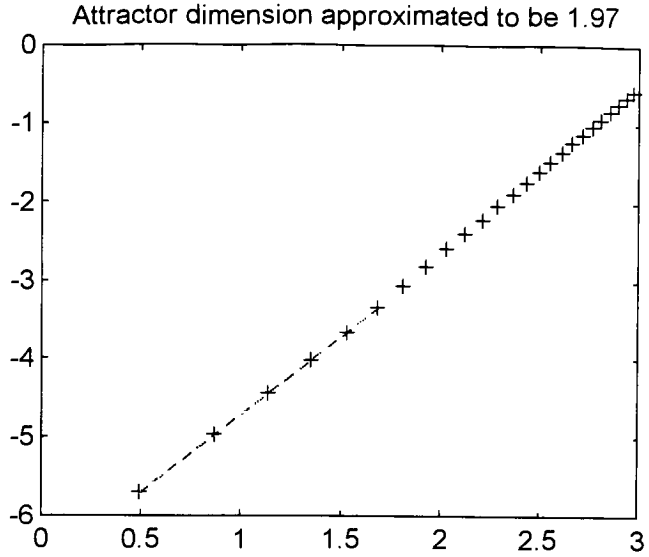
22:58:22.85
Elapsed time 4.951e+004 seconds.
Floating point operations = 3.891e+009.
normal termination

Time series analysis program by William Robertson.
4-Jul-94
22:45:51.03
Loading time series.
Read 300001 points from time series.
Using 5.000 points of time series, every 25 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 4.
22:46:11.68
Elapsed time 20.87 seconds

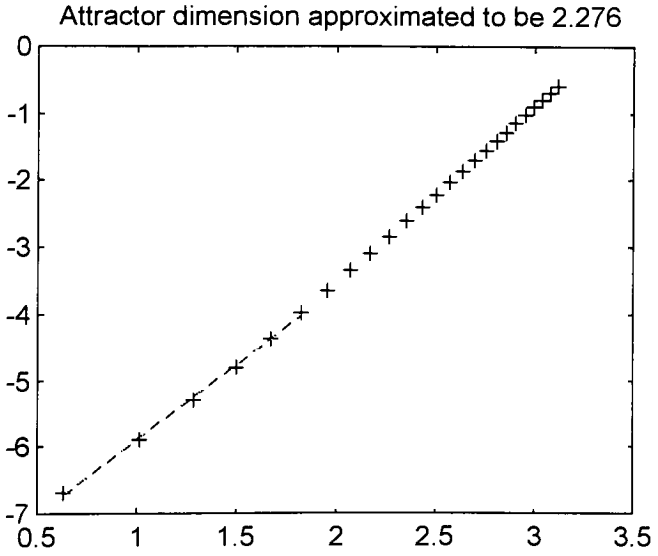
2-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
0:6:58.53
Elapsed time 4868 seconds.



3-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
1:31:40.34
Elapsed time 9949 seconds.

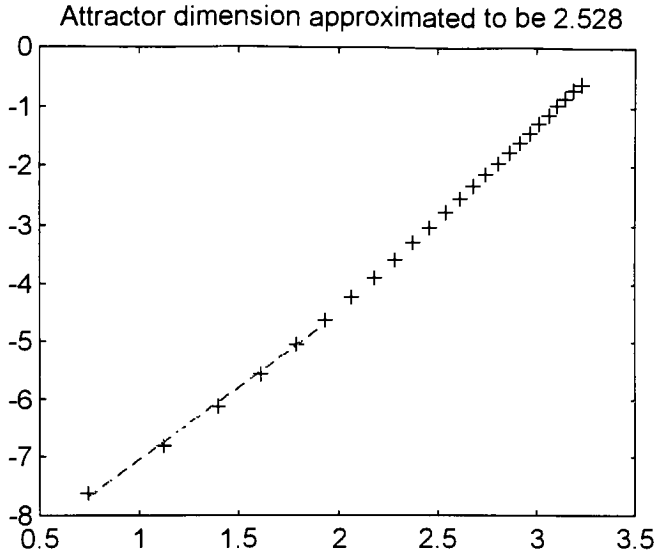


4-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4984
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
2:59:54.56
Elapsed time 1.524e+004 seconds.

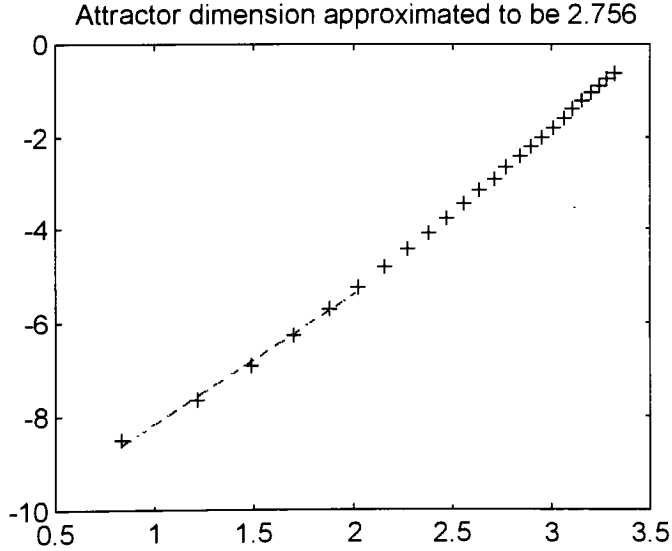


5-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4980
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
4:31:52.43

Elapsed time 2.076e+004 seconds.

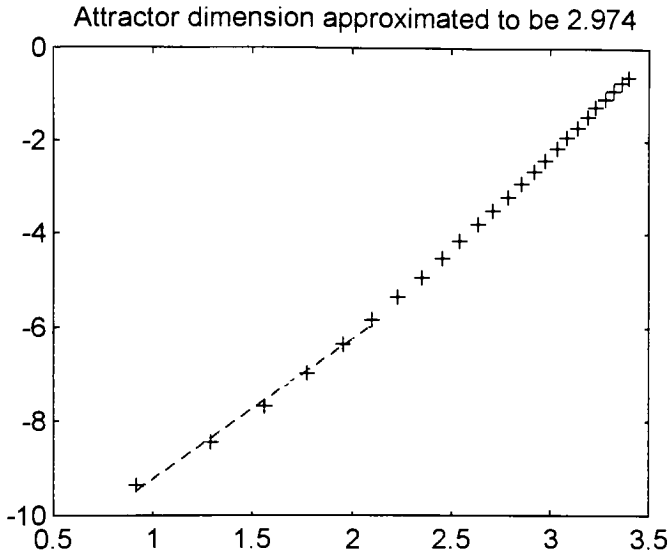


6-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4976
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
6:7:15.11
Elapsed time 2.648e+004 seconds.

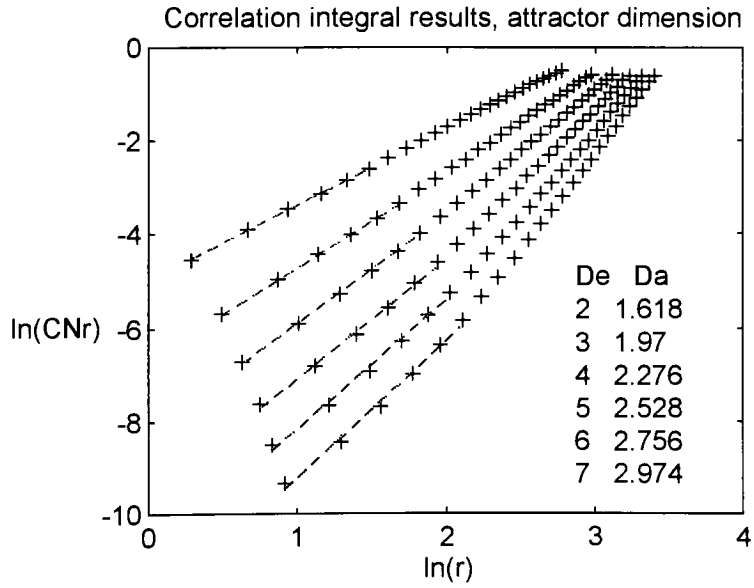


7-Dimensional embedding.
Using embedding delay of 4.
Number of embedded points = 4972
Using W factor of 2.
Finished correlation integral.
Finished LINREG.

7:46:15.69
Elapsed time 3.242e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.618
3	1.97
4	2.276
5	2.528
6	2.756
7	2.974

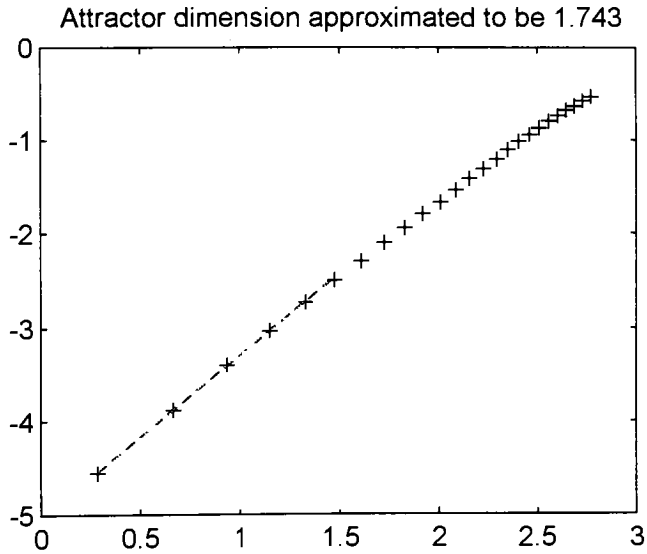


7:46:17.89
Elapsed time 3.243e+004 seconds.

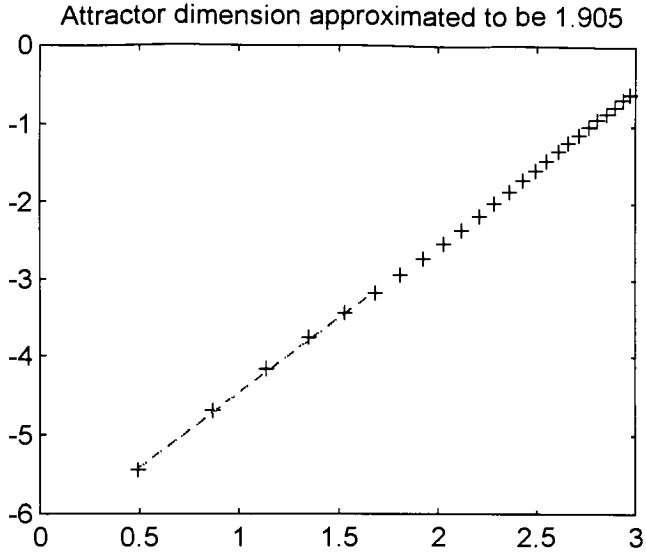
Floating point operations = 6.402e+009.
normal termination

Time series analysis program by William Robertson.
5-Jul-94
8:20:19.03
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 25 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 4.
8:20:39.3
Elapsed time 20.54 seconds

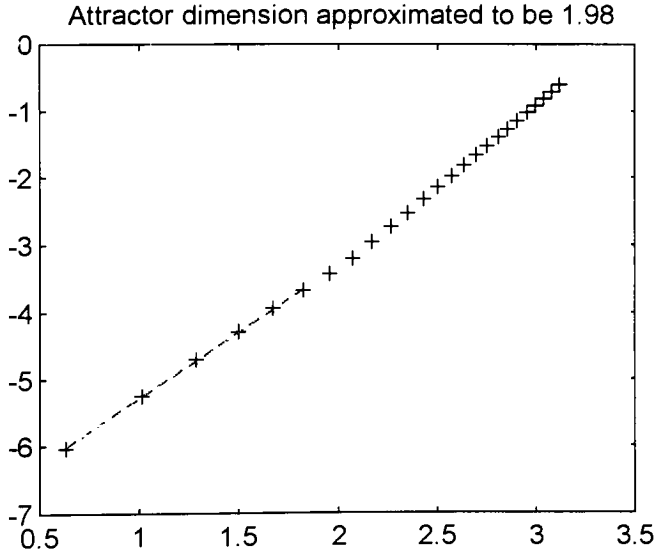
2-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
9:41:22.19
Elapsed time 4863 seconds.



3-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
11:6:11.53
Elapsed time 9953 seconds.

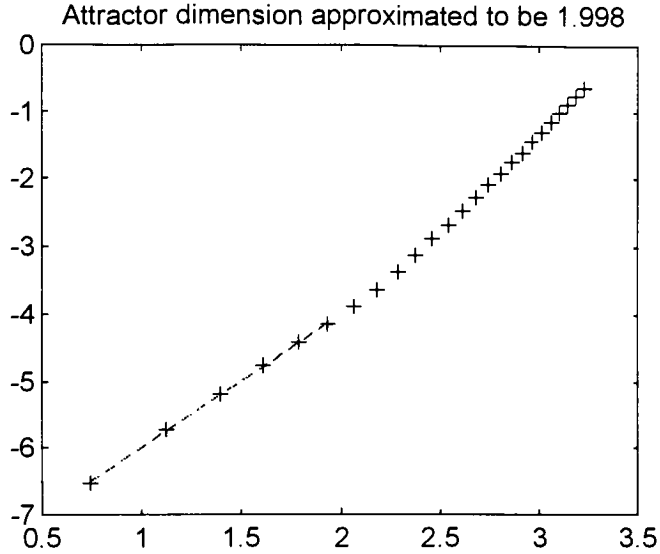


4-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:34:35.41
Elapsed time 1.526e+004 seconds.

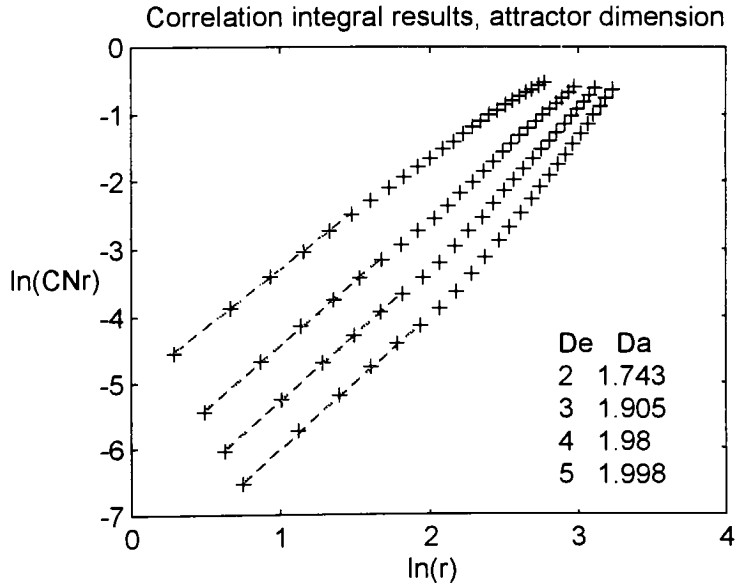


5-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
14:6:49.43

Elapsed time 2.079e+004 seconds.
Embedding procedure complete.



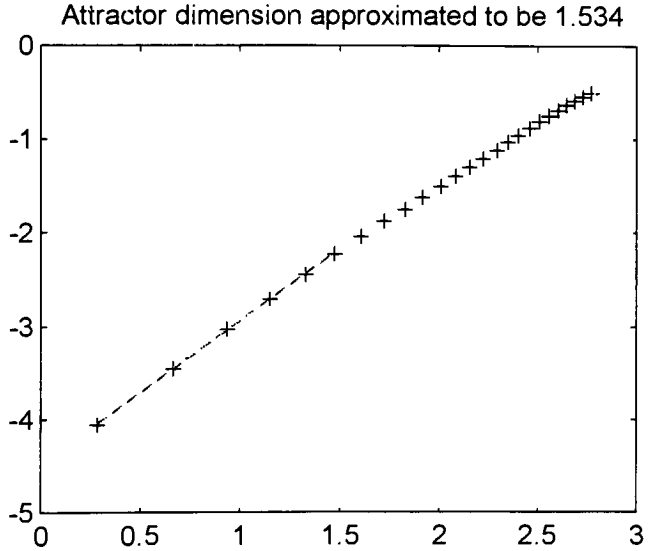
Embedding Dimension	Attractor Dimension
2	1.743
3	1.905
4	1.98
5	1.998



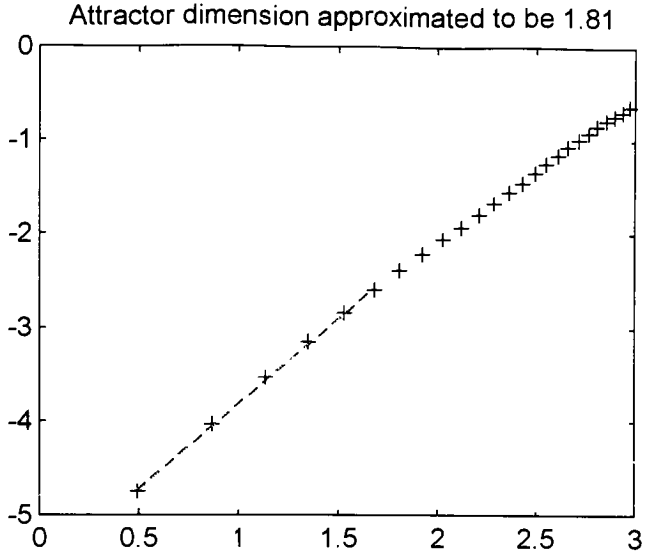
14:6:51.35
Elapsed time 2.079e+004 seconds.
Floating point operations = 3.887e+009.
normal termination

Time series analysis program by William Robertson.
5-Jul-94
14:11:2.03
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 25 point(s)
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 4.
14:11:22.63
Elapsed time 20.87 seconds

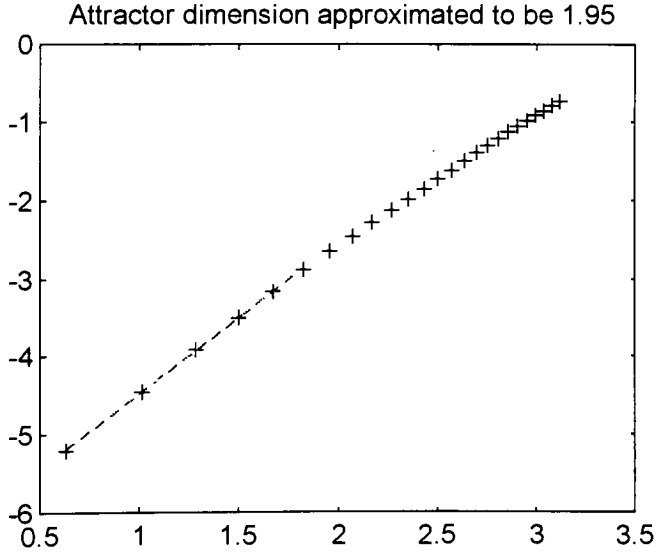
2-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4995
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:32:7.6
Elapsed time 4866 seconds.



3-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
16:56:57.44
Elapsed time 9956 seconds.

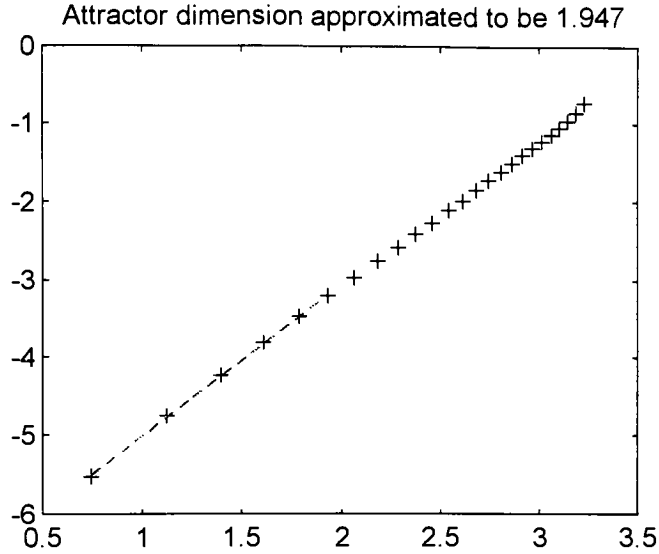


4-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4993
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:25:23.9
Elapsed time 1.526e+004 seconds.

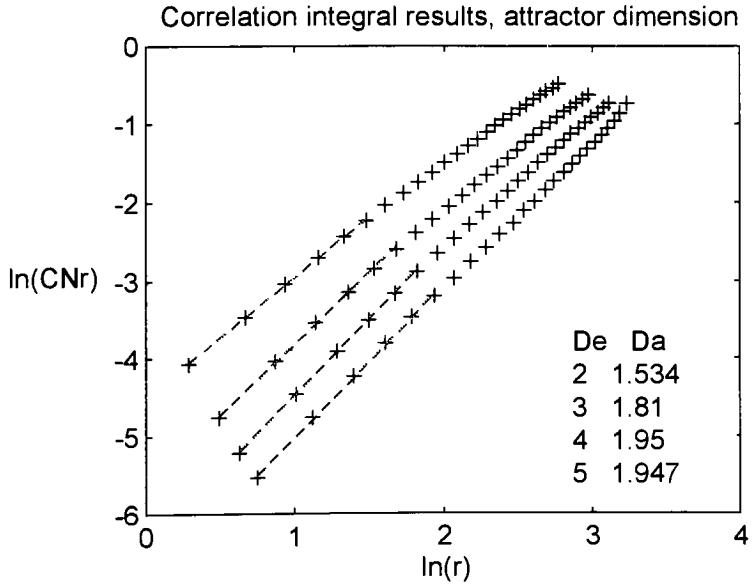


5-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
19:57:39.73

Elapsed time 2.08e+004 seconds.
Embedding procedure complete.



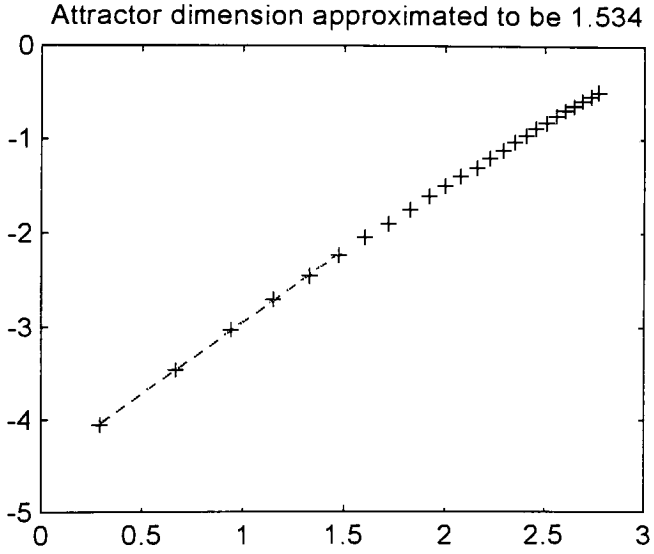
Embedding Dimension	Attractor Dimension
2	1 534
3	1.81
4	1.95
5	1.947



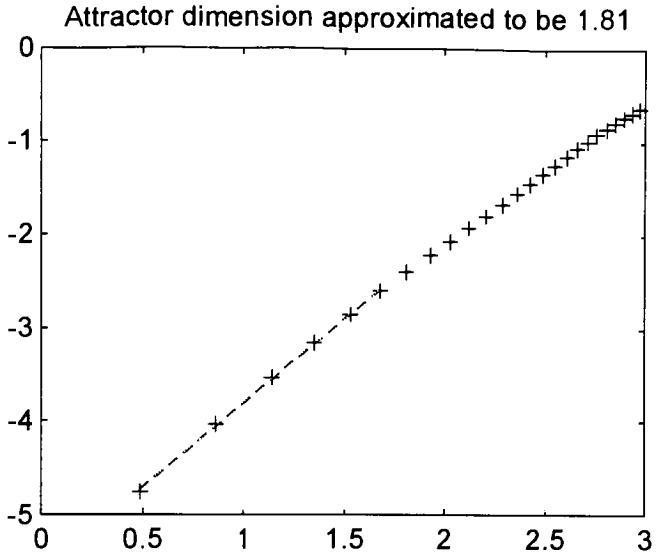
19:57:41.54
Elapsed time 2.08e+004 seconds.
Floating point operations = 3.891e+009.
normal termination

Time series analysis program by William Robertson.
4-Jul-94
8:46:18.91
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 25 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation local minimum at lag 4.
8:46:49.45
Elapsed time 30.93 seconds

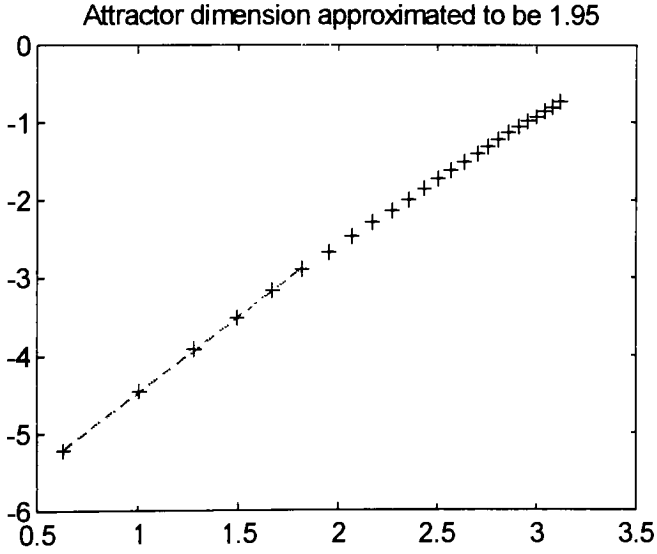
2-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4995
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:3:28.49
Elapsed time 1.183e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:26:39.86
Elapsed time 2.402e+004 seconds.

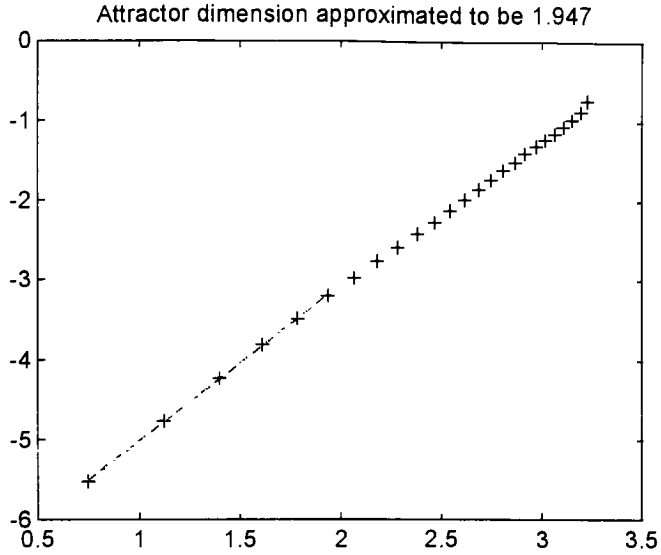


4-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4993
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
18:56:25.16
Elapsed time 3.661e+004 seconds.

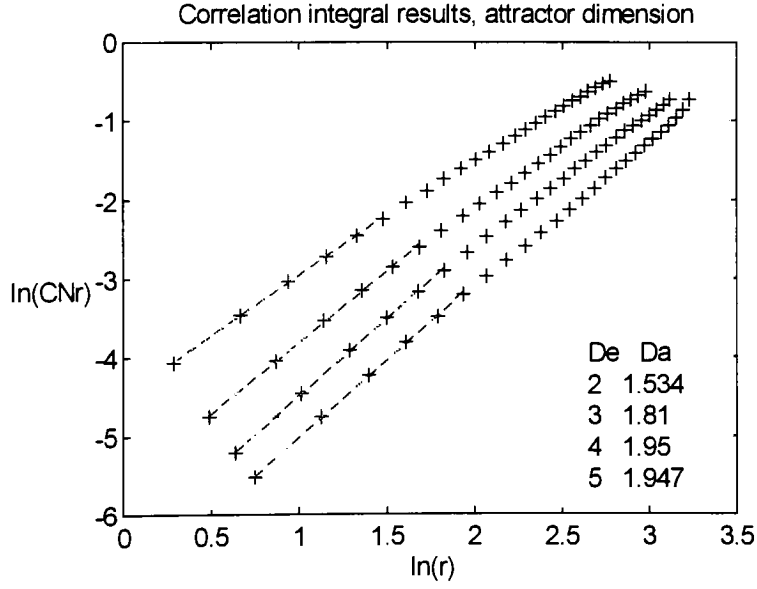


5-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
22:34:7.98

Elapsed time 4.967e+004 seconds.
Embedding procedure complete.



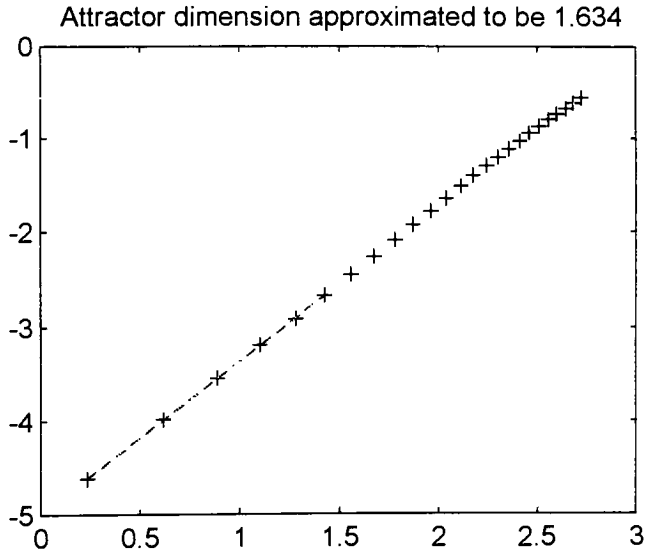
Embedding Dimension	Attractor Dimension
2	1.534
3	1.81
4	1.95
5	1.947



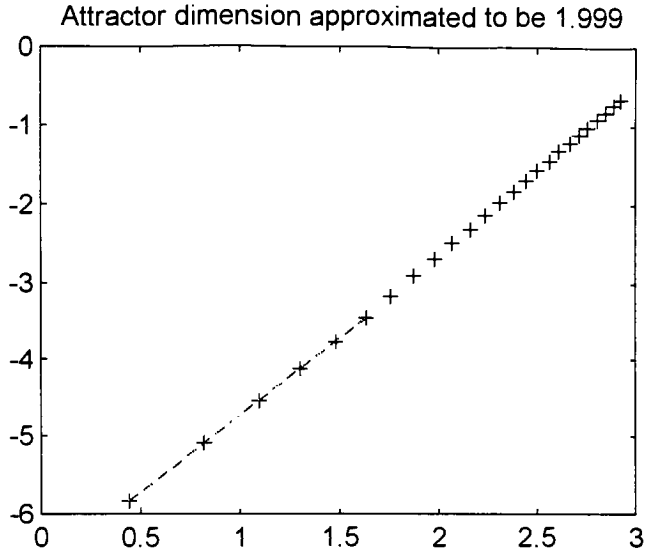
22:34:11.88
Elapsed time 4.967e+004 seconds.
Floating point operations = 3.891e+009.
normal termination

Time series analysis program by William Robertson.
5-Jul-94
21:34:16.95
Loading time series.
Read 300001 points from time series.
Using 5,000 points of time series, every 50 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 6.
21:34:47.49
Elapsed time 30.98 seconds

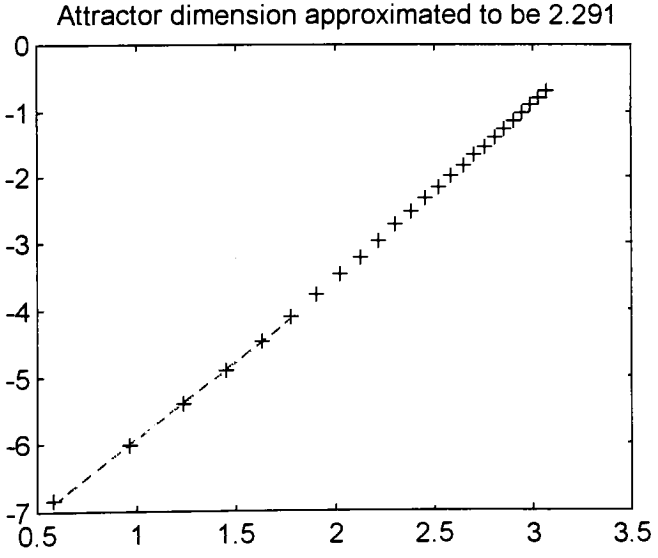
2-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
0:51:49.55
Elapsed time 1.185e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
4:15:33.11
Elapsed time 2.408e+004 seconds.

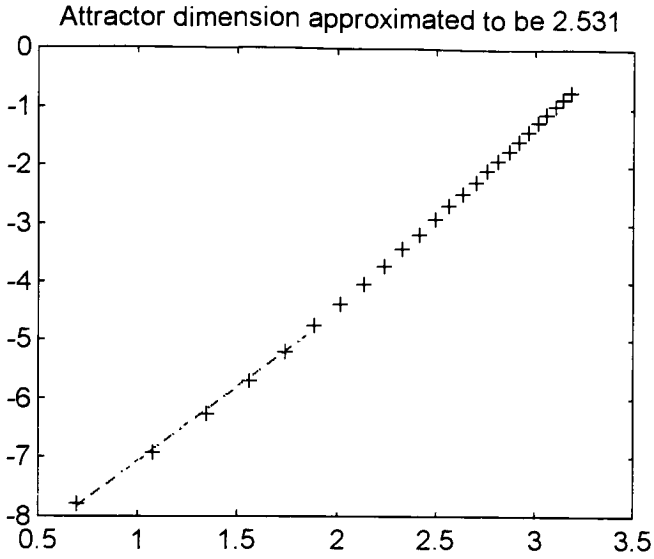


4-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4990
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
7:45:58.44
Elapsed time 3.67e+004 seconds.

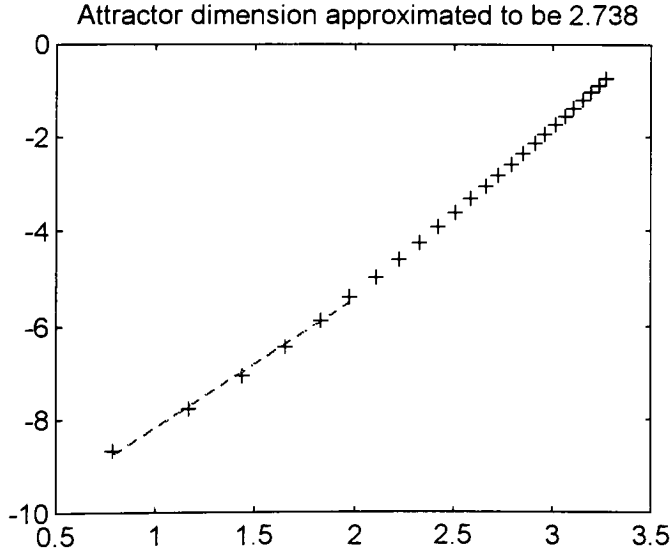


5-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4988
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
11:23:27.86

Elapsed time 4.975e+004 seconds.

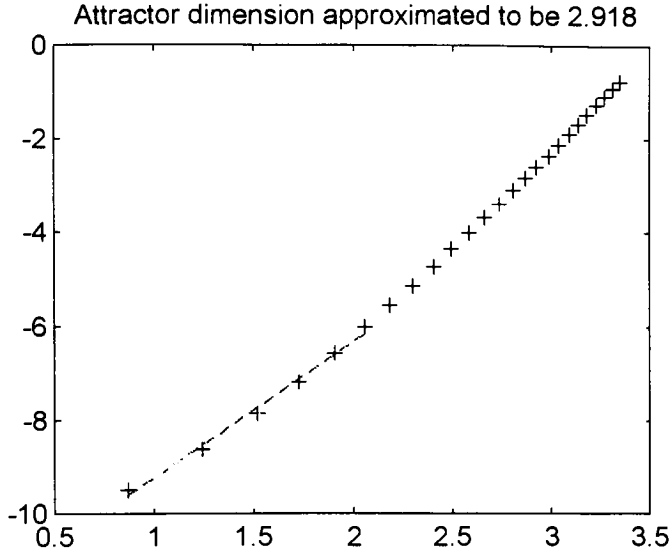


6-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4986
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
15:9:28.58
Elapsed time 6.331e+004 seconds.

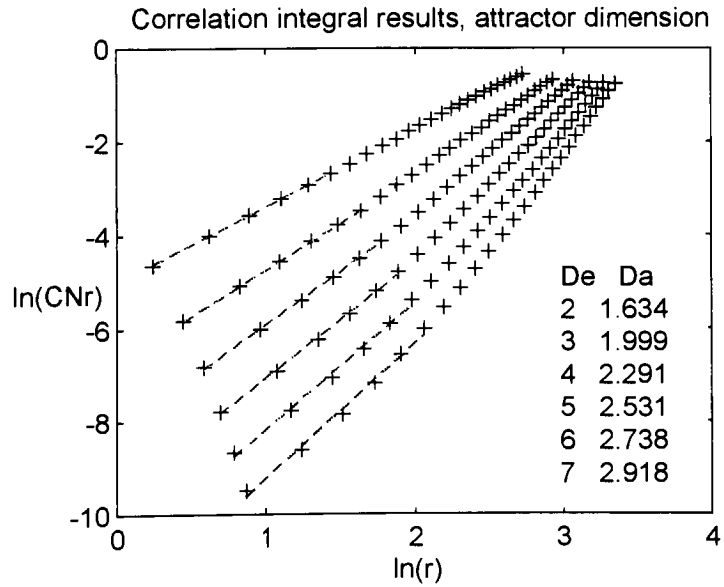


7-Dimensional embedding.
Using embedding delay of 2.
Number of embedded points = 4984
Using W factor of 2.
Finished correlation integral.
Finished LINREG.

19:1:54.82
Elapsed time 7.726e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.634
3	1.999
4	2.291
5	2.531
6	2.738
7	2.918

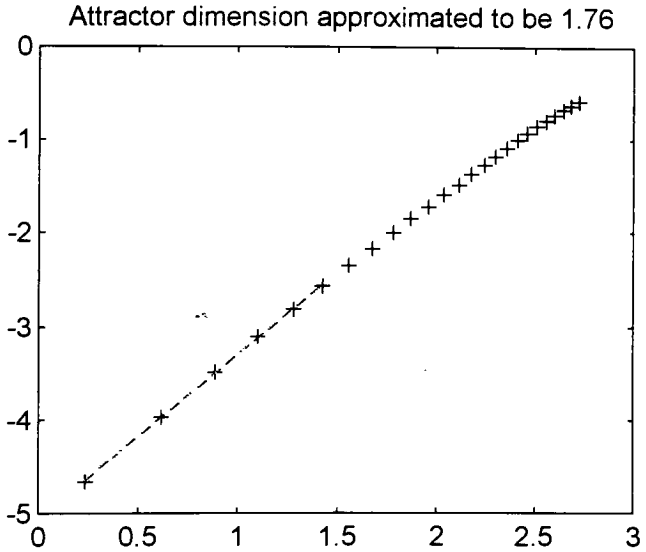


19:1:59.49
Elapsed time 7.726e+004 seconds.

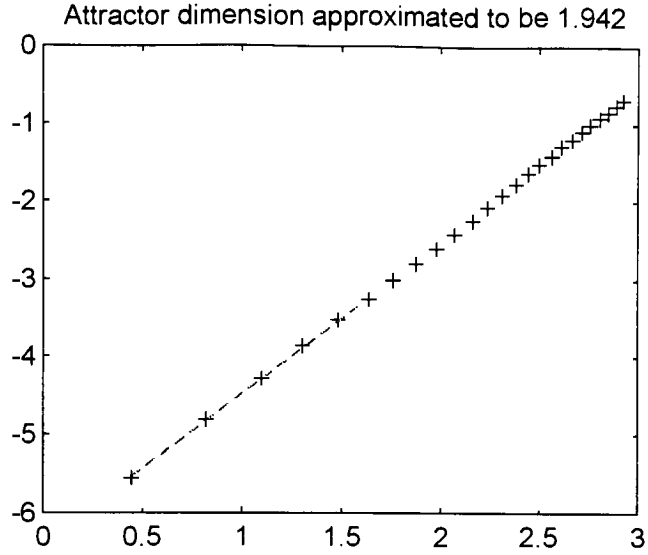
Floating point operations = 6.421e+009.
normal termination

Time series analysis program by William Robertson.
4-Jul-94
22:55:35.98
Loading time series.
Read 300001 points from time series.
Using 5.000 points of time series, every 50 point(s).
Calculating first phase for time series.
Calculating autocorrelation function.
Autocorrelation zero at lag 6.
22:56:6.41
Elapsed time 30.82 seconds

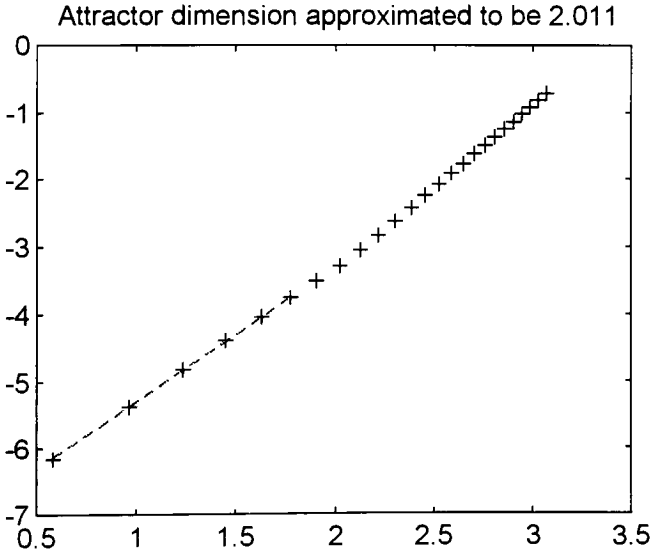
2-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4995
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
2:13:6.99
Elapsed time 1.185e+004 seconds.



3-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4994
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
5:36:58.62
Elapsed time 2.408e+004 seconds.

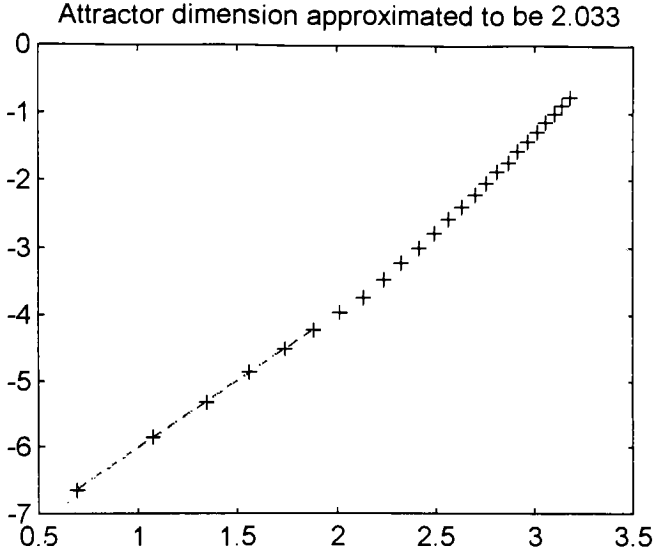


4-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4993
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
9:7:35.77
Elapsed time 3.672e+004 seconds.

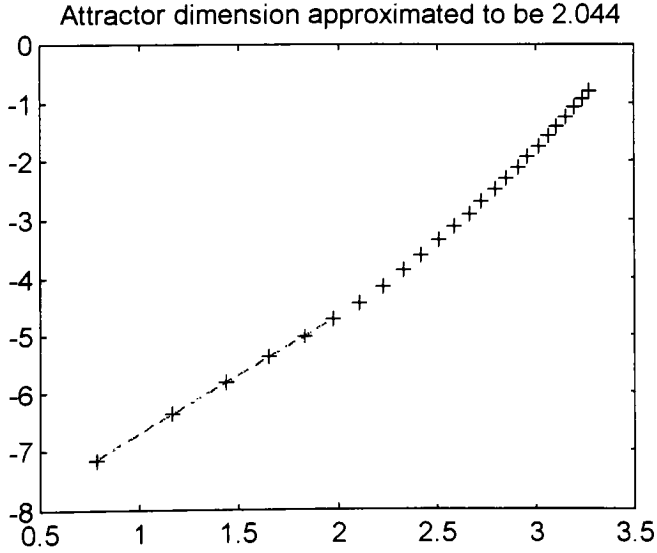


5-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4992
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
12:45:12.65

Elapsed time 4.978e+004 seconds.

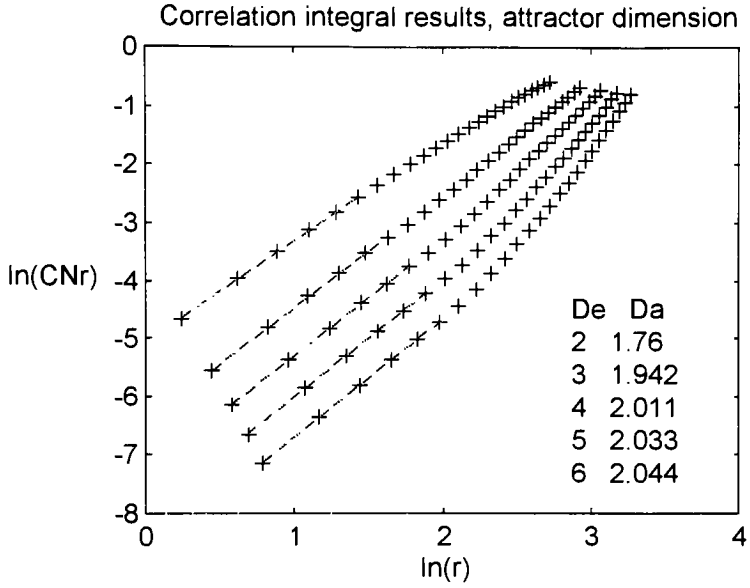


6-Dimensional embedding.
Using embedding delay of 1.
Number of embedded points = 4991
Using W factor of 2.
Finished correlation integral.
Finished LINREG.
16:34:2.65
Elapsed time 6.351e+004 seconds.
Embedding procedure complete.



Embedding Dimension	Attractor Dimension
2	1.76
3	1.942
4	2.011

5 2.033
6 2.044



16:34:7.21
Elapsed time 6.351e+004 seconds.
Floating point operations = 5.111e+009.
normal termination