SPACE TIME EVOLUTION OF DYNAMICAL SYSTEMS WITH FEW DEGREES OF FREEDOM

by

Ergun Eray Akkaya

Submitted to Graduate School of Natural and Applied Sciences

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Physics

Yeditepe University

2016

SPACE TIME EVOLUTION OF DYNAMICAL SYSTEMS WITH FEW DEGREES OF FREEDOM

APPROVED BY:

Prof. Dr. Avadis S. Hacınlıyan (Thesis Supervisor)	[illegible] …………………………
Prof. Dr. Yani Skarlatos	[illegible] …………………………
Assoc. Prof. Dr. Ertan Akşahin	[illegible] …………………………
Assoc. Prof. Dr. Ş. İpek Karaaslan	[illegible] …………………………
Assist. Prof. Dr. Hacı Ahmet Yıldırım	[illegible] …………………………

DATE OF APPROVAL: …./…./2016

ACKNOWLEDGEMENTS

I would like to acknowledge the support and help of my Professor Avadis Simon Hacınlıyan and thank him for his support since my undergraduate days.Working for my thesis under his guidance and supervision has been an experience that broadened my mind and presented an unlimited source of learning.

I thank Assist.Prof.Dr. Hacı Ahmet Yıldırım for his kind advice, help and support for this research, including his suggestion for analyzing the Sakarya river data.

Finally, I would like to thank my family for their endless love and support, which provided me a very productive environment thus making the process more beautiful.

ABSTRACT

SPACE TIME EVOLUTION OF DYNAMICAL SYSTEMS WITH FEW DEGREES OF FREEDOM

The thesis work involves analysis of chaotic behavior in physical systems modelled by low dimensional equation of motion. The paradigm of chaos became prominent in the late 1960’s. This has stimulated interest in this field, causing a rapid rise in the quality and quantity of research.

As one example of analysing chaotic behavior from a Lagrangian dynamical system, the Yang-Mills-Higgs system is studied which exhibits local instability but possesses a globally ordered phase by spantaneous symmetry breaking. Chaotic behavior and chaos to order transitions are analyzed. Addition of oscillatory term the region where the chaos-order transition occurs identified with an eye on transition back to order.

As a second example, regions of chaotic behavior in the parameter space of the Maxwell-Bloch equations (also Lorenz-Haken equations) has been studied as a constrained system.

The main part of this work involves identification of chaos in a set of experimental data, the monthly average discharge data of Sakarya River directly. Basic characteristics of chaos such as the irregularity of motion, unpredictability and sensitivity to intial conditions can thus be understood using nonlinear time series methods.. Using this data, possible low dimensional chaotic behavior of Sakarya river flow is investigated. To reveal the chaotic dynamics, maximal positive Lyapunov exponent is calculated from the reconstructed phase space obtained using the phase space reconstruction method. The approach reconstructs a locally equivalent phase space from the scalar time series from which the real system’s invariants can be estimated. Positive values for the maximal Lyapunov exponents have been calculated and this is an accepted indicator for chaotic behavior possibility. Analysed data contains the montly average flow rates of eleven main branches of Sakarya river through the years 1960-2000.

ÖZET

DÜŞÜK BOYUTLU DİNAMİK SİSTEMLERİN UZAY ZAMAN EVRİMİ

Bu tez çalışmasında düşük boyutlu hareket denklemleriyle modellenmiş fiziksel sitemlerin kaotik davranışları incelenmektedir. Kaos paradigması 1960 ların sonunda ortaya atıldı. O zamandan beri, bu alandaki araştırmaların niteliği ve niceliği hızlı bir şekilde artmaktadır.

İlk örnek olarak, Lagranjiyen bir dinamik sistem olan Yang-Mills-Higgs sisteminin kaotik yapısı çalışılmıştır. Bu sistem yerel düzensizlikler gösterir fakat ani simetri kırılmalarıyla küresel düzenli faza geçiş gösterir. Kaotik davranış ve kaostan düzene geçiş analiz edilmiştir. Salınım terimi eklenerek oluşan kaostan düzene geçiş bölgesi saptanmıştır.

İkinci örnek olarak, kısıtlı sistem olarak alınması gereken (Lorenz-Haken denklemleri olarak da bilinen) Maxwell-Bloch denklemlerinin parametre aralıklarının kaotik davranış bölgeleri incelenmiştir.

Bu tezin ana çalışması, Sakarya nehrinin aylık ortalama debi verilerinin yani deneysel olan zaman serisi verilerinde kaotik davranış saptanması içermektedir. Hareketteki düzensizlik, başlangıç koşullarına olan aşırı hassasiyet ve tahmin edilememezlik gibi kaosun temel karakteristikleri doğrusal olmayan zaman serileri metodları ile anlaşılmıştır. Bu verileri kullanarak, Sakarya nehir akışının olası düşük boyutlu davranışı araştırılmıştır. Kaotik dinamiği ortaya çıkarmak için, en büyük pozitif Lyapunov üsteli hesaplanması faz uzayının tekrar inşa edilmesi metodu ile elde edilmiştir. Yaklaşım skaler zaman serilerinden gerçek sistemin değişmezlerinin kestirilebileceği yerel eşdeğer faz uzayını yeniden yapılandırır. En büyük Lyapunov üstelin pozitif değeri hesaplanması (gözlenmesi), olası kaotik davranışın işareti olarak görülüp, kabul edilmiştir. Analiz edilen veri Sakarya nehrini besleyen on bir alt nehrin 1960-2000 yılları arasında aylık ortalama debi değerlerini içermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\vec{A}$	Vector potential
B	Parameter in Lorenz equations
$\vec{E}$	Electric Field
g	Coupling constant
H	Hamiltonian equation
k	Loss rate
L	Lagrangian
n	Integer value
P	Momentum
p	Oscillator parameter
$\overrightarrow{P}$	Polarization
$\vec{R}$	Any Vector
R	Parameter in Lorenz equations
x	Position coordinate
y	Position coordinate
$y_{nn}(k)$	Integrated time series in box n
$s_{nn}(k)$	Fourier transform of correlation function
$R_d(k)$	Distance in d dimensions
$\gamma_\perp$	Perpendicular loss rate
$\gamma_\parallel$	Parallel loss rate
$\vec{\Delta}$	Population inversion
$\vec{\Delta}_\mathbf{0}$	Population established by the pump mechanism
λ	Lyapunov exponent
λ	Eigenvalue
σ	Parameter in Lorenz equations
τ	Delay time
κ	Strength of anharmonic term

1. INTRODUCTION

Although in everyday life the word chaos is used to describe as a state of disorder and irregularity, chaos has a very different and specialized meaning in science. Low dimensional chaos involves sensitive dependence on initial conditions[1]. Traditionally, laws of nature reflected a complete relation of effects to causes. Until recently, it was assumed that one can make arbitrarily accurate predictions for arbitrarily long times of any physical system as long as one knows the starting conditions accurately enough. This is very obvious for linear systems where the cause and effect connection is clear. In the case of nonlinear systems, the cause and effect connection can sometimes be not so simple. Thus the problem of controlling such a system arises. Although the systems that chaos theory deals with are complicated and most of the time unpredictable, the basic concepts of chaos are not very difficult to understand. Now comes the question of how chaotic systems manifest themselves. A system can show multiple periodicity or apparent broadband noise as a result of underlying chaotic dynamics. To qualify and quantify chaos there are several dynamical properties to be used. Fractal dimension of the reconstructed phase space of the system, Kolmogorov entropy and Lyapunov spectrum are most important of these dynamical properties.

The connection between chaos theory and everyday life involves the analysis of time series obtained from real systems. The difficulty arises from the projection of a multidimensional system on a scalar one dimensional time series. Linear methods look for regular structures in a data set which means that the intrinsic dynamics of the system is governed by the paradigm that small variations cause small effects. So possible irregular behavior of the system can not be understood by linear paradigms where small variations in input can lead to large variations in the results.

Before explaining all terms first, let us understand dynamical system’s meaning. [2]. The word Dynamic means changing. Dynamical is concerning what is changed. For example, position of an artificial satellite changes relative to earth very rapidly so, we can say that this satellite is very dynamic. Furthermore, orbital energy of satellite as mentioned before changes very little, therefore, we can say that orbital energy is an approximate invariant.

Like many other studies, modern dynamic systems studies date to 1900s. However before 1957 the subject was a specialized field known mostly to astromers and physicists. When artificial satellites came into human life, also computers and powerful algorithms entered the stage with quick and easy calculations which are necessary, the study become a topic of concern for a broader portion of the scientific and engineering communities.

Dynamics began with classical mechanics and its application to planetary motions. In 19th Century electrical circuits provided more dynamical problems that could be solved. Today, high technical progress and computer simulations help to solve these problems, also sophisticated perturbation techniques can solve the same types of problems. Soviet mathematical physicists' work is a good historical example for superconvergent perturbation techniques. They provided humankind a lot of techniques caused by lack of resources to build laboratories and fast computers.

In addition population and ecology have provided some classical models to dynamics. The problem that remains for these specialities is to come up with enough data to create more refined forecasts. More promising is the study of biological subsytems, such as the heart and its neurological control system, because data can be collected systematically in a laboratory environment. These are many large models of the economy, some highly dynamical and others hardly at all. No methodology has won the day, because all give forecasts that are not nearly dependable.

Some models in economics and other fields violate a cardinal rule. When a model works poorly, the answer is not make it bigger, but to change something fundamental. Main thing to do with any kind of problem is to try to determine the limits of applicability to the system under study.

Many business forecasts use predictive methods however, they have little or no theoretical justification. They are only curve-fitting techniques needed for extrapolation aimed at prediction..

Today people with little or no specialized scientific training can use dynamics. Study of dynamics is not restricted to a small number of scholars and specialists since computers have made practical various numerical methods. The analytic theories developed by professional scientists do not need the help of computers and numerical solutions. In this thesis, basic numerical techniques and simple software programs to analyze phenomena have also been used.

Chaotic systems can be easily analysed by people because, numerical analysis and computer graphics are the tools that have made the study of chaotic systems accessible.

One of the pioneers in using computers in physics, especially for the study of chaos, was Mitchell J. Feigenbaum's work starting in 1974. In the 1960s the MIT meteorologist Edward Lorenz had discovered a chaotic dynamical system using approximations to the fluid flow, heat conduction and convection equations. Later Lorenz's results were used for different physical models and dynamics Feigenbaum's work spread knowledge about chaos and its usefulness as a modeling tool and teaching device was appreciated throughout the scientific community.

The novelty and exotic behavior of chaotic systems has been over emphasized during the period 1974-1990. In the future, present tools will be improved and new tools will become available, but these most likely will be further refinements of existing methods. Perhaps there will be great breakthroughs, but models adequate for most practical purposes can be constructed without waiting for them.

Dynamic systems can be classified into two groups. The first group involves the simpler linear systems. Linear systems are easy to understand since they obey the superposition principle according to which, for example, doubling the input doubles the output. Therefore, linear systems can be easier to solve and their long term behavior can be easily predicted. However, because of this property, they require initial conditions in order to become well-posed and set magnitudes for physical phenomena. The second group includes nonlinear systems with more complex patterns of behavior. The relation between input and output is much more complicated and impossible to understand in terms of results from the linear theory. This makes non-linear systems more complicated. Results from non-linear systems are usually called chaos because of this reason. But this does not mean chaos only occurs in complex systems or equations. Chaotic behavior can easily occur in a nonlinear equation which is not one to one invertible, if this equation is observed as a generator of time series such as;

$$x_{t+1} = 1.9 - x_t^2 \qquad (1.1)$$

Many features of Chaos Theory can be understood by the statement “ the behaior of certain nonlinear systems depend sensitively on the initial conditions .” [1]. An example can be given by comparing the behavior of a system of two nonlinear pendulums. The difference between the initial conditions that give rise to a total seperation of trajectories after a short while is only one second of arc. Actually, the exact opposite can also be true. Irrespective of the initial conditions, the system can finish on a common topological structure, such as the well known butterfly in the Lorenz system.

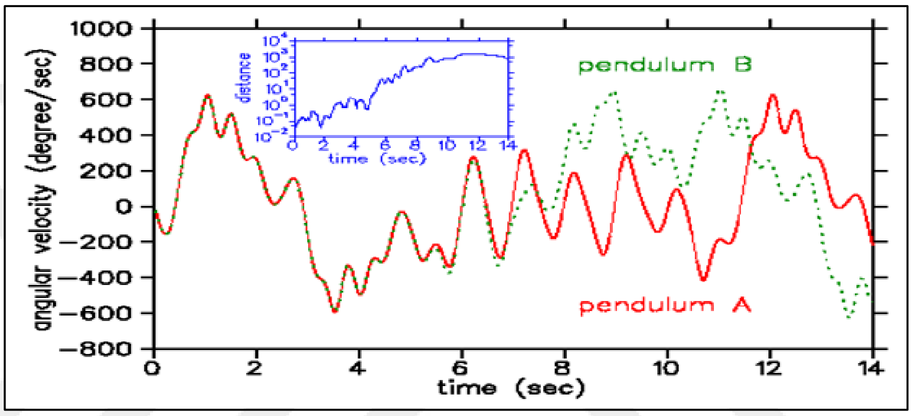

Figure 1.1. Example plot of two nonlinear pendula that small different starting angles.

The explanation for this latter type of chaos is topological transitivity. This roughly means that neighborhoods of points are mapped by the system into topologically big sets instead of sticking together in one or more localized neighborhoods.

Chaos Theory can be observed in most science fields for example, in systems that model species population and competition between species, in lasers, in heart movements, electrical signals from certain nonlinear circuits. The formulation of entropy by Boltzmann, in which the distribution of molecular speeds assume that position and velocity be uncorrelated was one of the first instances of the use of the term “collision number ansatz” that became known as the hypothesis of molecular chaos. Indeed, entropy is one of the parameters that characterize chaos.

Chaotic motion exhibits a lack of determinism even when all random ingredients are not present. The nature of a system and the topological structure in which it exists can impose limitations on a system, so that although two nearby points can separate exponentially, the system remains on a limited structure of the space which can allow it to move in a seemingly regularly way on a seemingly smooth space, while its chaotic motion lies in a highly complicated subspace, also called “ strange attractor.” The well known Lorenz attractor is an example.

In general, mathematicians study the mathematical fundamentals of this theory, in terms of the theory of dynamical systems and topological measure. On the other hand, there are several physical phenomena that can only be explained by nonlinear models. Fortunately, chaos theory does not need higher mathematical knowledge. Its fundamentals can be understood and applied in terms of relatively simple mathematical concepts such as basic algebra, calculus and ordinary differential equations. In this respect application of chaos theory to physical systems is not much different from linear mathematical problems.

To summarize, chaos shows complex, unsystematic motion, instabilities and sudden changes because of highly sensitive dependence on initial conditions. A chaotic system may seem to be haphazard but actually its evolution and invariants (meaning the extent of predictability of its final state) only obeys different rules from those of linear systems. Actually, the Lyapunov exponent measures this predictability horizon. Seemingly deterministic nonlinear systems exhibit such behavior and this aspect of chaos theory is usually called “Deterministic chaos”. Simple systems and equations in one variable (for example, the system given in 1.1) can generate chaos. Chaos does not mean measurement error or noise, it is independent from all of these.

In this thesis, two examples of analysing chaos in terms of systems modelled by differential equations will be given. The first example involves investigating chaotic behavior in bosonic Yang Mills Fields by the aid of certain standard mathematical tools such as linearized stability analysis, Lyapunov’s direct method and lyapunov characteristic exponents.

An inspiration for studying the established knowledge of the Yang-Mills or Yang Mills expanded by Higgs mathematical statements is the significance of this framework in the starting instability or stability at, since in the beginning stages all associations were of the same quality and depended on non-abelian gauge hypotheses, of which the SU(2) Yang Mills is a first example. In this study we consider the accompanying two dimensional (four variable) Hamiltonian recommended by Biro Matinyan and Müller.

$$H = \frac{P_x^2 + P_y^2}{2} + \frac{1}{2}x^2y^2 - \frac{1}{2}y^2 + \frac{1}{8}x^4 + \frac{1}{4}\kappa y^4 \quad . \tag{1.2}$$

Here H is hamiltonian P=(p_x,p_y) is the momentum and x and y are coordinates and κ is nonlinear anharmonicity parameter. In gauge hypotheses the acknowledgedment of diverse stages, especially a limiting stage for the most part connected with a disordered field setup and the Higgs stage portrayed by an all-around requested field condensate, is a focal issue. As needs be the qualification between established dynamical frameworks having or not having a complete arrangement of nontrivial integrals of movement is of essential significance. In this part coupled Yang- Mills- Higgs framework which displays an all-inclusive requested stage because of unconstrained symmetry breaking.

A further example in this thesis is to analyse Maxwell- Bloch equations, sometimes referred to as Lorenz-Haken equations. They represent a possible working mechanism for a class of lasers. They are based on the classical equations for the electromagnetic field and quantum mechanical formulations for the particles under the combined DC and AC magnetic fields. For regions of appropriate values involving control parameters, seemingly chaotic behavior can result. Under special conditions on the parameters, the laser model is related to the Lorenz model and behave similarly. Maxwell-Bloch equations exhibit various types of routes to chaos for different parameter ranges. In this study, a constrained Lagrangian form that lead to Maxwell-Bloch equations has been obtained from the equivalent treatment of the Lorenz model. This allows us to analyze the long term behavior of its attractor.

The main part of the thesis involves the analysis of the Sakarya River flow using nonlinear time series analysis. Sakarya River is the fourth largest river in Turkey and first largest river in Marmara and Anatolian Regions (Turkey). It is 810 km long and its width is between 60 m and 150 m. In this river, there are 3 dams for hydroelectric power production. One is Sarıyar dam located in the Ankara region Nallıhan subriver. It is constructed in the 1950s. The others are Yenice and Gokcekaya dams which are located in same region of the Sakarya River.In Figure 1.2., one can see information about Sakarya River’s branches and locations.

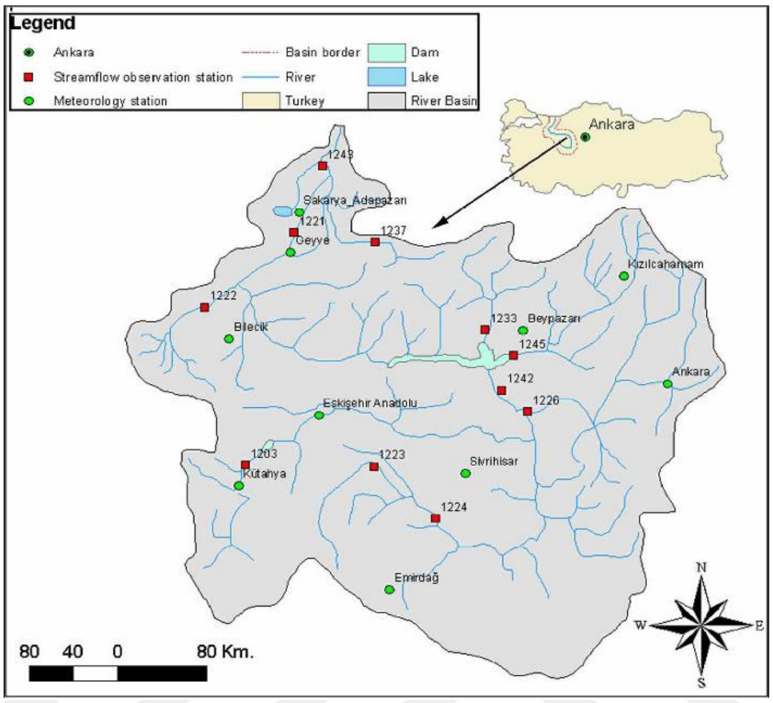

Figure 1.2. Map that demonstrates location of Sakarya River.

2. MATERIALS AND METHODS

The experimental data involves the average monthly discharge rate of Sakarya River. The data have only scalar values and are taken from the [3] EIE (General Directorate of Electrical Power Resources Survey and Development Administration). Fifty-four stream flow observation stations have been set on the Sakarya River by the EIE and the observation period spans the period 1960-2000. Flow rate time series graphs are presented below. The units on the x axis is time in months and the y axis is flow rate in $m^{3.}$

It can be seen that the flow rates for each tributary show similar patterns in spite of the fact that Sakarya river covers a relatively large and varied region involving two different climatic regions.

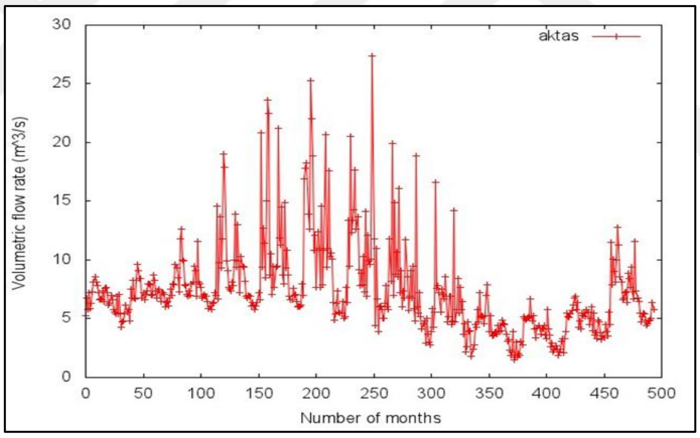

Figure 2.1. Flow rate data on given months of Aktas Subriver .

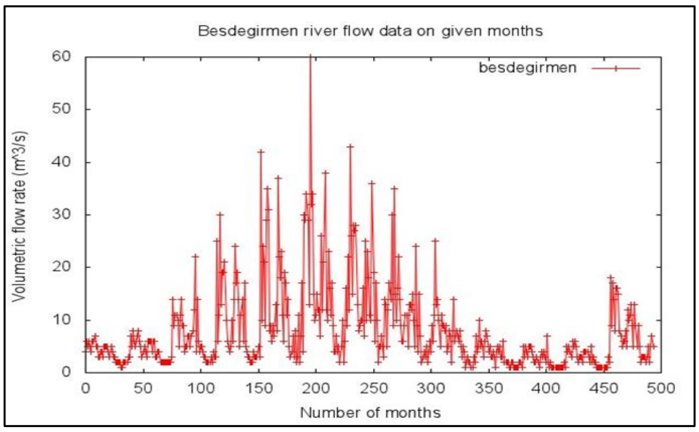

Figure 2.2. Flow rate data on given months of Besdegirmen Subriver .

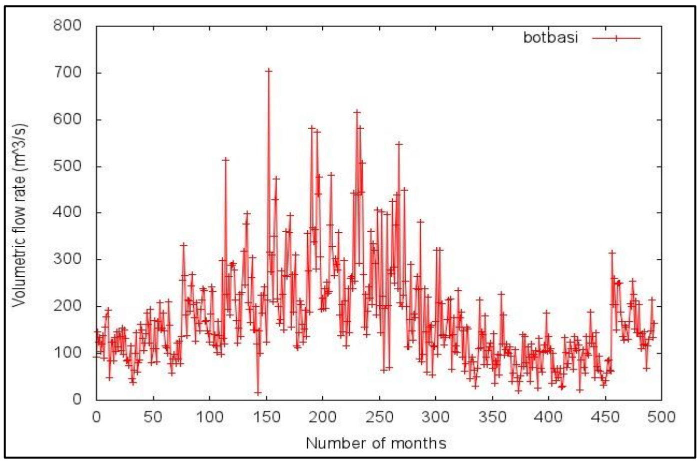

Figure 2.3. Flow rate data on given months of Botbasi Subriver .

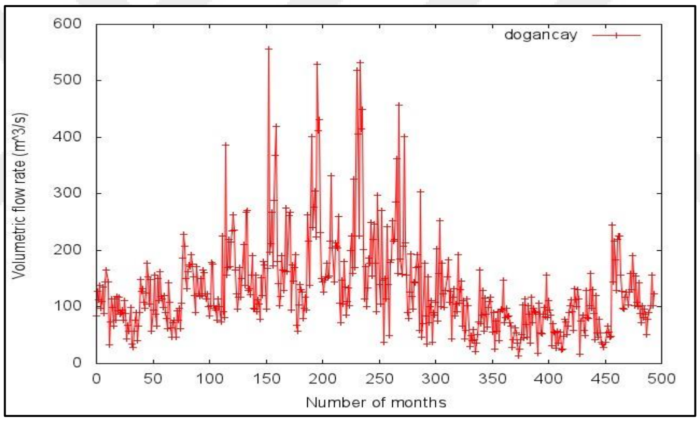

Figure 2.4. Flow rate data on given months of Dogancay Subriver .

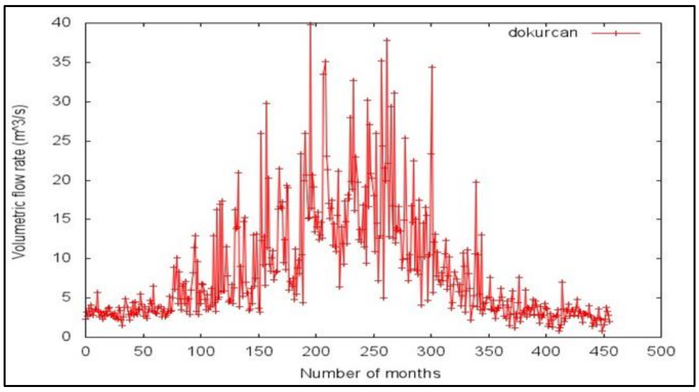

Figure 2.5. Flow rate data on given months of Dokurcan Subriver .

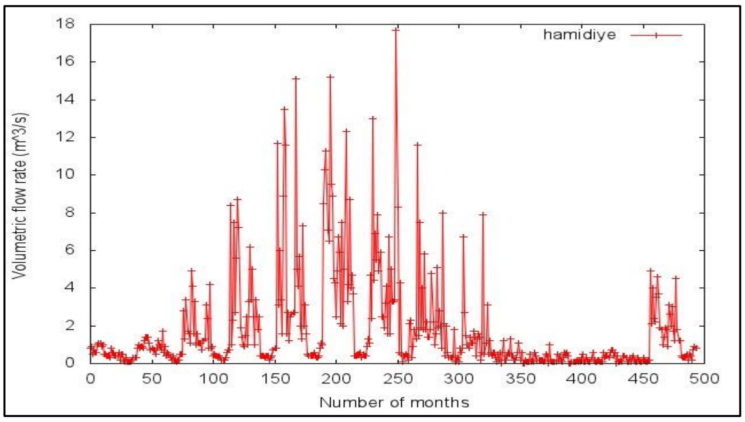

Figure 2.6. Flow rate data on given months of Hamidiye Subriver .

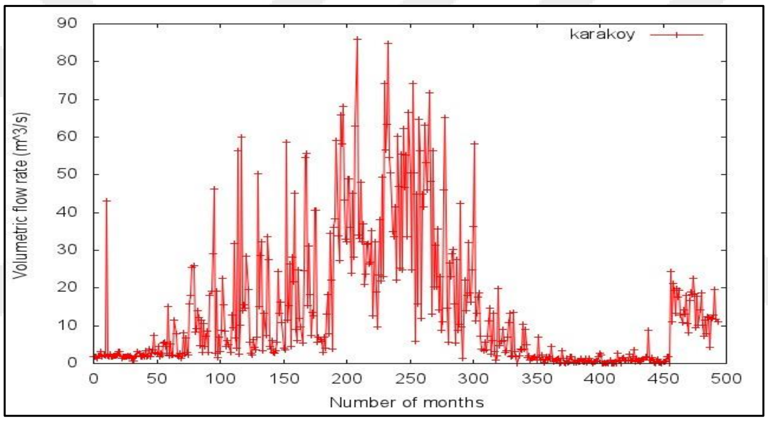

Figure 2.7. Flow rate data on given months of Karakoy Subriver .

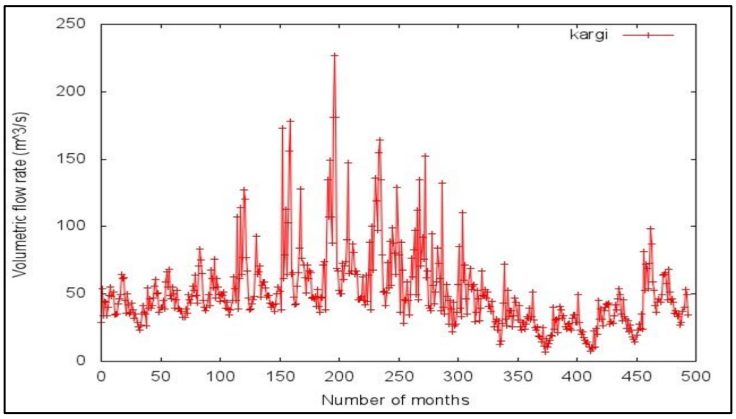

Figure 2.8. Flow rate data on given months of Kargi Subriver .

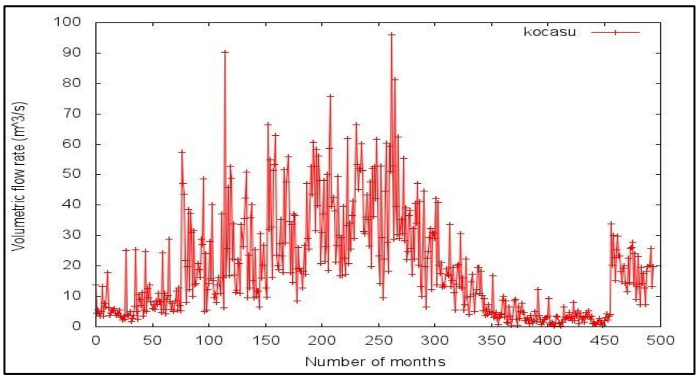

Figure 2.9. Flow rate data on given months of Kocasu Subriver .

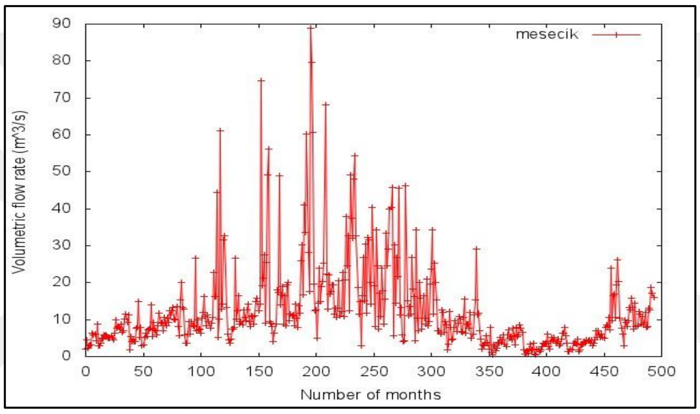

Figure 2.10. Flow rate data on given months of Mesecik Subriver .

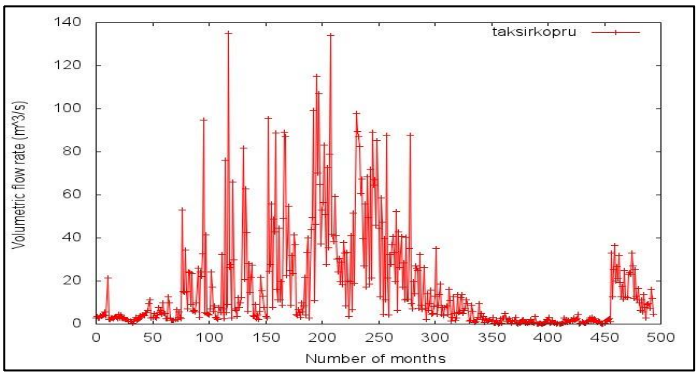

Figure 2.11. Flow rate data on given months of Taksirkopru Subriver .

If one looks at these signals in order to determine their nature, one can see both periodic and irregular behavior. A study of the correlation function confirms this conjecture. Here is the correlation function for the Aktas tributary, we observe a decrease up to about 7-8 months, but the correlation function never reaches zero. It than reveals a periodic behavior involving approximately 40 months. A similar behavior can also be seen for the Botbasi tributary.

Figures showing both correlation functions are presented below. It is clear that such behavior usually implies that the data involves multiple scales. Multiple scales does not necessarily imply chaos, however, there are several instances in which such systems make transition to chaotic behavior, an example involving electro-optical oscillators is given in [4]. In that case, the dynamical system is given. In the case of Sakarya River flow, a one dimensional time series is given and the phase space must be reconstructed. This requires setting a delay time and if multiple time scales are involved, a choice must be made between the zero of the correlation function and the first minimum of the of the mutual information. Although there is no clear indication of consistent success, the latter is usually preferred. [5]. It will be reported in Sections 4 and 5 that the mutual information analysis gives a delay time of 5-9 months. This agrees with the lower figure seen in the correlation graphs.

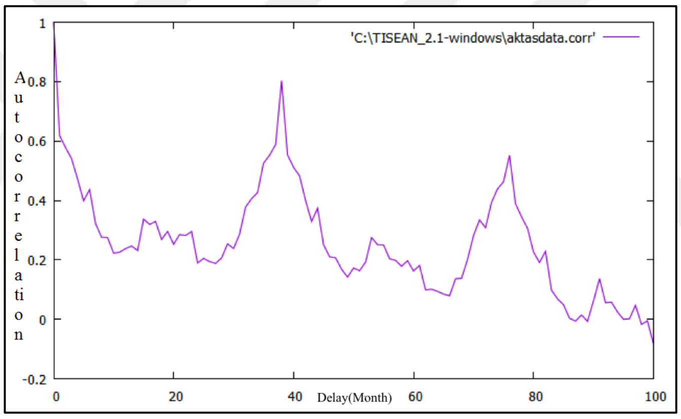

Figure 2.12. Correlation for the Aktas Subriver

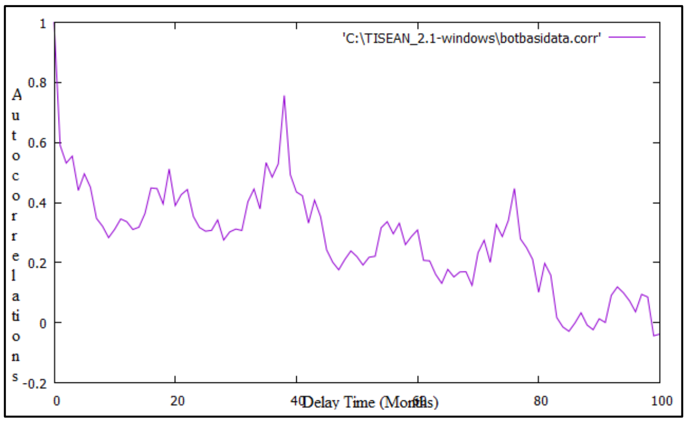

Figure 2.13. Correlation for the Botbasi Subriver

3. CHAOS THEORY AND ITS APPLICATIONS

As mentioned in introduction, chaotic systems show sensitive dependance to initial conditions. However, initial conditions may not be well known and be subject to errors, uncertainities or perturbations. Long-term predictions are impossible. Even the availability of computational support will not enable one to generate long-term predictions. Poincaré stated in 1903:

A very small cause which escapes our notice determines a considerable effect that we cannot fail to see... even if the case that the natural laws had no longer secret for us... we could only know the initial situation approximately... It may happen that small differences in initial conditions produce very great ones in the final phenomena.

Let us observe Lorenz deterministic non-periodic flow [1] which is known as one of the best example of chaos theory. In working on the behavior of a flowing fluid system subject to a temperature gradient such that conduction and convection effects compete, Lorenz used a well known approximation to the Navior-Stokes equations and obtained the following system involving three ordinary nonlinear differential equations containing six parameters:

$$\begin{aligned} \dot{x} &= -\sigma x + \sigma y \\ \dot{y} &= Rx - y - x \qquad (2.1) \\ \dot{z} &= -Bz + xy \end{aligned}$$

Since $\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y} + \frac{\partial \dot{z}}{\partial z} = -\sigma - 1 - B < 0$, the system is dissipative and has an odd number of degrees of freedom.

The parameters σ, R,B determine the behavior of the system. These three equations are extremely sensitive to initial conditions. The ordinary parameter values for demonstrating chaos are σ=10, R=28, B=8/3. If one starts with suitable initial value and solves the equations with a numerical procedure, it will be shown that the very well known butterfly pattern as in the Figure 2.1 emerges.

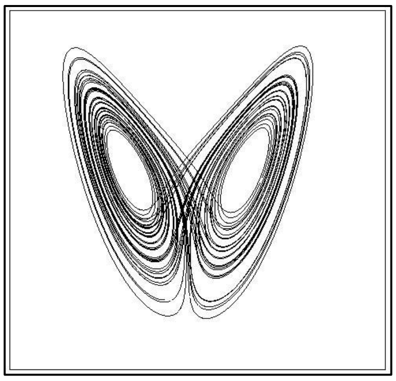

Figure 3.1. Plot of Lorenz Attractor in 3D

Figure 2.1 shows the Lorenz attractor in x, y and z dimensions. Chaos theory is not used in local weather reports because of not knowing the initial conditions at time t=0 very accurately. However, progress is observed for day to day large scale simulations.

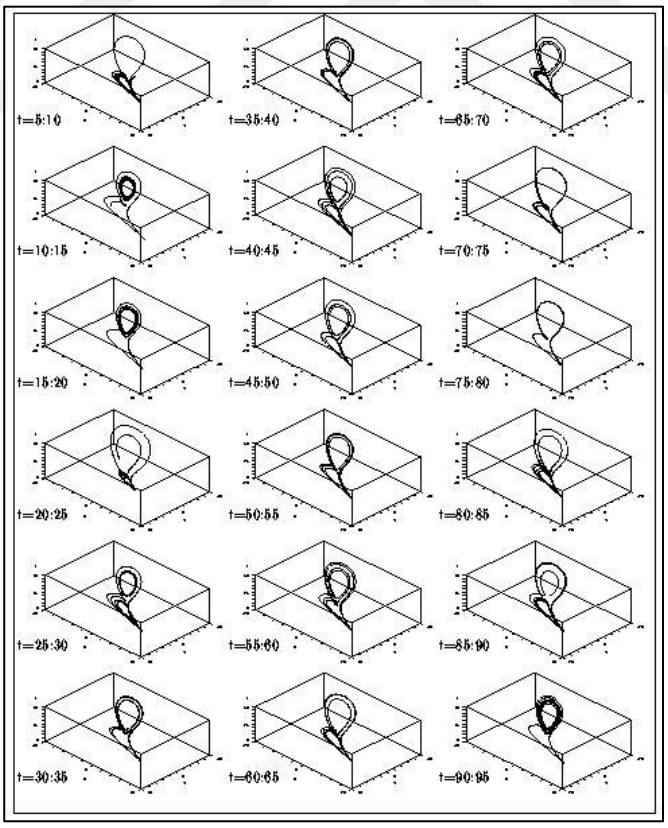

Figure 3.2. Lorenz Attractor in intervals of Δt=5 seconds.

It can be seen in Figure 2.2 that rapid changes of the Lorenz attractor which the figure demonstrates during the time development of the Lorenz attractor in steps of $\Delta t=5$ seconds.

We can categroize chaotic system by in two groups. One is the system has inherent chaotic behavior in nature, the other is generated by people either artificially or by the way it is observed. Chaotic systems can be used by industry. Chaotic systems only appear to be random, but they are indeedly deterministic, they posses an emphasizing order. This emphasizing order provide to the possibility of controlling chaotic systems.Ott and Grebogi, Willian Ditto and Louis Pecora are pioneer scientists and have created methods of controlling chaotic mechanical, electrical and biological systems [6]. Important example is the fact that syncronized chaos can be used in cyrpto electronic communications.

4. HAMILTONIAN CHAOS

Hamiltonian Mechanics is a branch of classical mechanics offering a deeper insight into the connection between Newtonian and Lagrangian mechanics. It treats generalized coordinates and moments on equal form. There are many applications such as simple harmonic oscillators, accelerators,planet orbits and the weather, to name a few. Perturbations in Hamiltonian systems give rise to chaos, which can be explained through KAM theory (Kolmogorov- Arnold-Moser)[7]. Although the study of Hamiltonian Mechanics has been around since the 1800's, chaos is only now beginning to be understood.

Hamiltonian systems satisfy a number of properties, but before adressing those, an overview of what a Hamiltonian system needs to be addressed. A Hamiltonian system with n degrees of freedom on an open subset E of R^{2n} must satisfy the following:

Let H ϵ C^2 (E) where H=H(x,y) with x,y ϵ R^n. Then the system.

$$\dot{x} = \frac{\partial H}{\partial y}\ , \dot{y} = -\frac{\partial H}{\partial x} \tag{4.1}$$

where x represents a generalized coordinate vector and y represents the correspondin conjugate momentum vector satisfies

$$\frac{\partial H}{\partial x} = \left(\frac{\partial H}{\partial x_1}, \dots\dots, \frac{\partial H}{\partial x_n}\right)^T \quad and \quad \frac{\partial H}{\partial y} = \left(\frac{\partial H}{\partial y_1}, \dots\dots, \frac{\partial H}{\partial y_n}\right)^T \tag{4.2}$$

with $\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y} = 0$ so that phase space volume is conserved and hence Hamiltonian systems have the symplectic property.

Hamiltonian systems are conservative, which can easily be extended to many physical systems.

Hamiltonian systems have numerous properties. The most well-known property is that Hamiltonian systems conserve at least one particular quantity, namely energy . More specifically, the total energy H of the Hamiltonian system remains constant along its trajectories. This can be proven by taking the derivative of H with respect to time . In doing so, we get that it equals zero, which proves that H is constant along any solution curve.

Therefore, the trajectories of the system are on the surfaces of H, which is a constant. The critical points of H correspond to the equilibrium points of the system. Furthermore, the equilibrium points are non-degenerate if the determinant of the second derivative of H is nonzero when evaluated at the equilibrium points. Stability can also be determined using the second derivative of H. When evaluating the second derivative at an equilibrium point, if all the eigenvalues have a positive real part, then that point is stable. If we consider Hamiltonian systems of degree one, then non-degenerate equilibrium points can be classified more readily. If an equilibrium point is non-degenerate, than we have that the point in question is a saddle point of the Hamiltonian system if and only if it is a saddle of the Hamiltonian function H. Furthermore, it is a center if and only if it is a local maximum or a local minimum of H.

In a non-chaotic Hamiltonian system near a point of stable equilibrium, the motion is oscillatory. Thus, geometrically we get that the orbits of the system move on tori. But, for chaos to be introduced into the system, the tori need to be destroyed. By destroying the invariant tori, the system in turn creates a cantori which is system of cantor sets. Chirikov first discovered that for local chaos to ocur in a Hamiltonian system, stable and unstable manifolds had to intersect. And this chaos occurs when $S^2 > 1$, where $S = \frac{\Delta\omega_r}{\Omega_d}$, $\Delta\omega_r$ is the frequency and Ω_d is the distance frequency between two unperturbed resonances.

It should be noted that the tori are not perturbed in the integrable Hamiltonian system H that was defined earlier. Rather, the system becomes perturbed when a nonintegrable Hamilton perturbation is added. In doing so, the following equation is obtained:

$$H = H_0(y) + \epsilon H_1(x, y) \tag{4.3}$$

In adding the nonintegrable function, tori begin to deform, and those that survive are " sufficiently irrational." In fact, according to KAM theory, tori survive for $C^{'}$ perturbations if $\left|\frac{dH_0}{dy_0} * k\right| > \lambda|k|^{-\tau}$ for $N-1 < \tau < 1/2r-1$ and λ is of order $\sqrt{\epsilon}$ for small ϵ , where λ is based on frequency. But for chaos to ocur, even these "sufficiently irrational" tori become perturbed.

4.1 CHAOS IN YANG-MILLS EQUATIONS

In the electromagnetic case, the gauge transformation that led to the minimal substitution $P_{mech} = P_{can} - \frac{e}{c} A_i$ depended on a single parameter, the electric charge e. This gauge transformation is Abelian since two successive transformations commute.

It is possible to consider more complicated cases. The equations of motion are still in 3+1 dimensional Minkowski space time. However the potential now depends on an internal degree of freedom respecting an $\mathrm{SU}(2)$ symmetry. This theory is the free Yang Mills fields with the equations of motion, involving the field strength F and a coupling constant g [15]:

$$\partial^{\mu} F^{a}_{\mu\nu} + g\varepsilon^{abc} A^{a\mu} F^{c}_{\mu\nu} = 0 \tag{4.4}$$

The field strength is derived from a vector potential $\vec{A}^{a}_{\mu}$:

$$F^{a}_{\mu\nu} = \partial_{\mu} A^{a}_{\nu} - \partial_{\nu} A^{a}_{\mu} + g\varepsilon^{abc} A^{b}_{\mu} A^{c}_{\upsilon} \tag{4.5}$$

Here, the Latin letters denote the inward degrees of flexibility and take the range of 1,2 or 3 while the Greek letters denote spacetime parts taking the range 0,1,2,3. For the electrodynamic case the third term vanishes since there is one only one internal degree of freedom that is the electric charge. A summation over the full range over repeated indices is assumed throughout. The energy momentum tensor can be obtained as

$$T_{\mu\nu} = -F^{a\lambda}_{\mu} F^{a}_{\nu\lambda} + \frac{1}{2} g_{\mu\nu} F^{a}_{\lambda\rho} F^{a\lambda\rho} \tag{4.6}$$

$g_{\mu\nu}$ is the Minkowski space time metric tensor. We seek a class of solutions of these equations for which the Poynting vector vanishes in an appropriate coordinate frame , $T_{0j} = F^{a}_{0i} F^{a}_{ji} = 0$ which means there is no energy flow .

Using this condition and fixing the gauge by $A^{a}_{0} = 0$, equation 3.4 can be written as

$$\ddot{A}^{a}_{i} - F^{a}_{ji,j} + g\varepsilon^{abc} A^{b}_{j} F^{c}_{ji} = 0 \tag{4.7}$$

This solution admits two conserved quantities, namely the angular momenta $M_i = \varepsilon_{ijk} A_j^a \dot{A}_k^a$ and the corresponding quantity in Yang-Mills internal indices $N^a = \varepsilon^{abc} A_i^b \dot{A}_i^c$.

This condition implies that N^a is zero for the solution representing the vacuum and the external color charge density if not. Here dots are used for the time derivatives, the Latin indices i, j, k denote spacelike ones. The notation $F_{ij,k}$ means a space gradient $\partial_k F_{ij}$ .The constraints $N^a = 0$ and $\dot{M}_i = 0$ lead to

$$\dot{A}_i^a \left(A_{i,j}^a - A_{j,i}^a \right) = 0 \tag{4.8}$$

This equation implies one of the following conditions:

- $A_{i,j}^a = 0$,
- $\dot{A}_i^a = 0$,
- $A_{i,j}^a - A_{j,i}^a = 0$,

corresponding to homogeneous, static and irrotational vector potentials respectively.In the homogenous case Yang Mills equations becomes

$$\ddot{A}_i^a - g^2 \left(A_j^a A_j^a A_i^b - A_j^b A_j^b A_i^a \right) = 0 \tag{4.9}$$

We can derive these relations from the following Hamiltonian,

$$H_{YM} = \sum_a \frac{1}{2} \left(\dot{A}_a \right)^2 + \frac{1}{4} g^2 \sum_{a,b} \left(A^a \times A^b \right)^2 \tag{4.10}$$

Here, A^a denotes the three dimensional vector potential having components A_i^a . The spherical symmetry if this Hamiltonian in both space and internal degrees of freedom is manifest. The system given by this Hamiltonian is also known as classical Yang Mills mechanics, where it was originally used to prove instability of pure Yang Mills fields.

In gauge theories the different phases' realization, particularly a confining phase is usually associated with a disordered field configuration and the Higgs phase featured by a globally ordered field , is a central situation. Accordingly the existence or otherwise of a full set of nontrivial first integrals of motion in a classical dynamical systems is crucial for the stability issue. In this part we will add add a scalar doublet (in gauge space) of fields and will denote this as a coupled Yang-Mills-Higgs system. Rapid symmetry breaking will force this system from the initial unstable state to a globally ordered phase.

This system has SU(2) as the internal isospin gauge group. It contains an isodoublet scalar field besides the customary Yang Mills gauge fields. Its Hamiltonian density in the $A_0^a = 0$ gauge is [8]

$$H = H_{YM} + \frac{1}{2}(\dot{\sigma}^2 + B_a^2) + \frac{g^2}{4}(A_i^a)^2(B_a^2 + (\sigma - \upsilon)^2) + \frac{\lambda^2}{2}(B_a^2 + (\sigma - \upsilon)^2 - \upsilon^2)^2 \tag{4.11}$$

Here the scalar isodoublet Higgs field φ is represented by

$$\phi = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} iB_1 + B_2 \\ \sigma - iB_3 - \upsilon \end{pmatrix} \tag{4.12}$$

With υ being related to the vacuum expectation value of the Higgs field by

$$\langle \phi \rangle = \begin{pmatrix} 0 \\ -\dfrac{\upsilon}{\sqrt{2}} \end{pmatrix} \tag{4.13}$$

λ is a coupling constant can be described the strength of the quartic self-interaction of the Higgs fields. After the equation of motion following from the Hamiltonian there is a constraint equation which is the generalization of the nonabelian Gauss law

$$\varepsilon^{abc} A_i^b \dot{A}_i^c = \frac{1}{2}\left(\varepsilon^{abc} B_b \dot{B}_c + \dot{\sigma} B_a - (\sigma - \upsilon)\dot{B}_a\right) \tag{4.14}$$

Let take consider again homogeneous gauge field configuration and a Higgs field tending to one point with its vacuum expectation value $\langle \varphi \rangle$, requiring $\sigma = B_a = 0$, one arrives at the following simplified Hamiltonian for $n = 2$ degrees of freedom [9]

$$H = \frac{1}{2}\left(\dot{x}^2 + \dot{y}^2\right) + \frac{1}{2}x^2y^2 + \frac{g^2\upsilon^2}{4}\left(x^2 + y^2\right) \tag{4.15}$$

According to the physical perception, at high energy, unification with strong fields or very wide amplitude motion in the model system, the addition of the Higgs condensate $g\upsilon$ is not important and the motion is still chaotic.(v=0 is always chaotic, if v is very large compared to x and y on a closed orbit the motion is simple harmonic.) Meanwhile, for weak fields the nonlinear coupling x^2y^2 becomes insignificant and the motion is prevalent by stable, ordinary oscillations.

By the method of rescaling of the amplitudes and time it is anything but difficult to see that the motion administered by the Hamiltonian is controlled by a single dimensionless parameter

$$K = \frac{g^2v^4}{4H} \tag{4.16}$$

Here H is the conserved energy. For $K = 0$ the system converges to pure Yang Mills mechanics. Chaos appears then for small values of the parameter K , in spite of the motion becomes ordinary when K is large.

It can be deduced that the spontaneous breaking of the gauge symmetry by the Higgs mechanism has a stabilizing effect on the nonabelian gauge field dynamics.

Beyond these results, if we consider Higgs field as a dynamical field coupled to the gauge field, the equations of motion are that of the $SO(3)$ Georgi-Glashow model [15]

$$(D_\nu F^{\mu\nu})^a = -g\varepsilon^{abc}\phi^b(D_\mu\phi)^c \tag{4.17}$$

$$(D_\mu D^\mu\phi)^a = -\lambda\phi^a(\phi^b\phi^b - \frac{m^2}{\lambda}) \tag{4.18}$$

where $D_\mu^{ab} = \delta^{ab}\partial_\mu + g\varepsilon^{abc}A_\mu^c$ is the covariant derivative. Making the ansatz $A_i^a = \varepsilon_{aij}a_j(t)$ and $\varphi^a = \sqrt{2}b^a(t)$ in the $A_0^a = 0$ gauge a purified system of equation can be derived. Setting $a_i = x$ and $b^a = y$, it becomes

$$g^{-2}\ddot{x} = -6y^2x - 3x^3 \tag{4.19}$$

$$g^{-2}\ddot{y} = -6x^2y + y - py^3 \tag{4.20}$$

Performing the scale transformations $t \to \alpha t$, $x \to \beta x$ and $y \to \beta y$ with $\alpha = \frac{1}{m}$ and $\beta = \frac{m}{g\sqrt{6}}$ we arrive at the following simple dynamical system

$$\ddot{x} = -xy^2 - \frac{1}{2}x^3 \tag{4.21}$$

$$\ddot{y} = -x^2y + y - py^3 \tag{4.22}$$

containing only one scale parameter $p = \frac{\lambda}{g^2}$.It has been shown that this system contains chaotic motion in a large range of values of the energy and of the parameter p.

In this section, we compute Lyapunov exponents with the aid of Fortran code that implements the Wolf algorithm[10] explained below.

First of all we investigate how exponents are changing with respect to the scale parameter, p. And we found that system possesses chaotic motion in wide range of value of p. Especially we scan for the interval from $p = 0.05$ to $p = 4$.Here are some of graphs for the specific values of p.

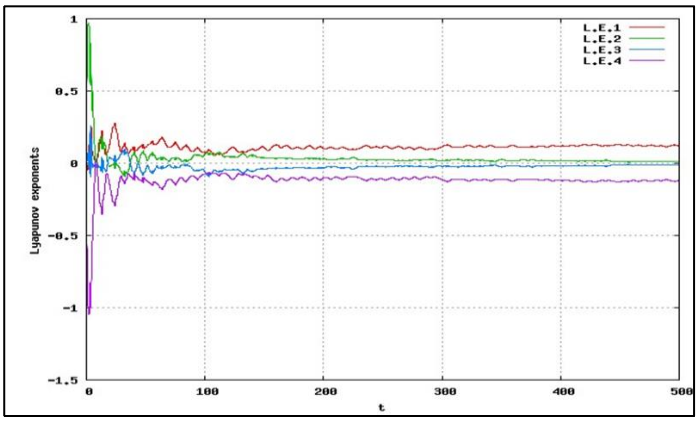

Figure 4.1. Lyapunov exponents vs time for

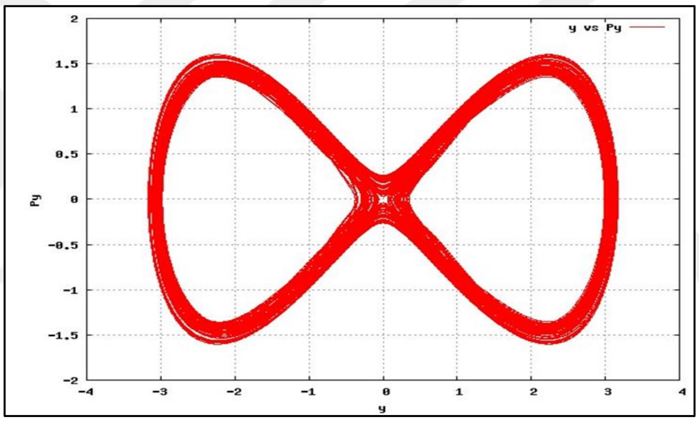

Figure 4.2. Trajectory of y vs Py

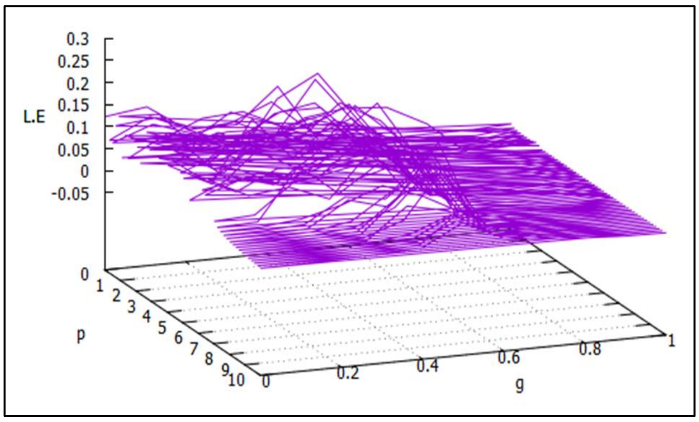

Figure 4.3. Lyapunov exponent values according to change in κ and g parameters

In Figure 4.3. Changing of parameters κ and g parameters are being changed by the use of Wolf Algorithm [10] and Lyapunov exponent values are seen below zero and stabilesizes when g parameter came to 5 that parameter values are arbitrary numbers.

Lyapunov Exponents are dynamical invariants that can be calculated with relative ease for systems modelled with differential equations (continuous time systems), maps (discrete time systems) and time series. The Lyapunov exponents measure the exponential rate of separation of nearby trajectories as shown in Figure 4.4.

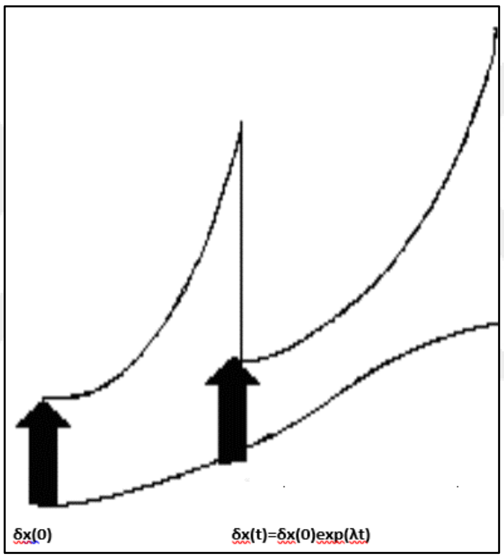

Figure 4.4. Lyapunov Exponents

Wolf's algorithm [10] is one of the simplest methods for the calculation of Lyapunov exponents One starts from a fiducial trajectory $x(t)$ As mentioned before Lyapunov exponents are defined by the long term evolution of an infitesimal hypersphere of states. The dynamical system

$$dx_i(t)/dt = f_i(x_{\{j\}}) \tag{4.23}$$

is augmented by N neighboring trajectories that obey the variational equations.

$$d(\delta x(t))/dt = \left(\frac{\partial f}{\partial x}\right)_{x(t)} \delta x(t) \qquad (4.24)$$

where δx(0) is the initial separation between neighboring trajectories, δx(t) is the separation at time t, dots denote time derivatives and $\left(\frac{\partial f}{\partial x}\right)_{x(t)}$ is the Jacobian of the dynamical system evaluated on the fiducial trajectory. Each variational equation obeys the initial condition $\left|\delta x_i\right|_j = \delta_{ij}$ denoting that the jth component of the i'th variational equation is 1, the other components are 0, thus forming an hypersphere around the fiducial trajectory. The variational equations can be obtained from the dynamical system using a symbolic programming language. The REDUCE code. The Fortran code is that given by Wolf. As a numerical integrator for the fiducial trajectory and the variational equations, the reference to the proprietary IMSL routines have been replaced by Numerical Recipes routines for fourth order Runge Kutta integration with adaptive quality control [11].

The fourth order Runge Kutta method has a number of advantages. First of all it is self starting. Secondly it is a robust method for most systems except those that are unusually stiff [23]. Thirdly, for a step size of h, the error is O(h^5). This enables one to make a check by assuming that the exact integral I, and the Runge Kutta result of stepsize h are related by $I = I(h) + Ah^5$. Since function evaluations for a stepsize 2h are included in the stepsize h, one can obtain I(h) and I(2h) given as $I = I(2h) + 32Ah^5$. One can combine these to estimate the error term Ah^5 and adjust the stepsize for quality control.

Chaotic behavior involves stretching which corresponds to positive Lyapunov exponents and folding which corresponds to negative ones. As the hypersphere evolves, the directions corresponding to stretching will grow exponentially and dominate. To avoid information loss due to truncation errors that will come from the finite precision, the rates of increase and decrease must be recorded and Gram Schmidt orthogonalization must be performed in order to accurately follow the evolution of the hypersphere about the fiducial trajectory.

4.2 CHAOS IN MAXWELL-BLOCH IN LAGRANGIAN FORM

The Lorenz and Maxwell-Bloch systems are not in Hamiltonian form. However, pretty much as on account of the Lorenz model, [12] it can be reexpressed as an constrained , velocity subordinate Lagrangian system.

The Lorenz system can be composed in Lagrangian form and can be studied as a mechanical syste[16] along in spite of the fact that it is a compound mechanical thermodynamical dissipative system. The Maxwell-Bloch conditions have a bigger number of parameters than the Lorenz framework however this framework can be changed into the Lorenz framework, if $x=\vec{E}$, $y=g\vec{P}/k$ and $z=\vec{\Delta}_0-\vec{\Delta}$ changes are made. Then, the following parametric identifications also need to be made $k=\sigma$, $\gamma_{\perp}=g^2/k$, $R=g^2\Delta_0/k$, $\gamma_{||}=B$.

Consider a one-particle system with a velocity dependent Lagrangian of the form.

$$L=\frac{1}{2m}\,[\![(m\mathbf{v}-\lambda\vec{A})^2=\frac{1}{2}m\vec{v}^2-\lambda\vec{\boldsymbol{v}}\cdot\vec{A}+\frac{\lambda^2}{2m}\vec{A}^2 \qquad (4.24)$$

$\vec{A}$ is the vector potential and $V=(\lambda^2\vec{A}^2)/2m$ is the scalar potential. The explicit form of our nonlinear potential can be written;

$$\frac{\lambda}{m}\vec{A}\left(\vec{E},\frac{g\boldsymbol{P}}{k},(\vec{\Delta}_{\mathbf{0}}-\vec{\Delta})\right)=\frac{1}{\tau}\begin{bmatrix} k\left(-\vec{E}+\frac{g\vec{P}}{k}\right) \\ \vec{E}\left(\frac{g^2\Delta_{\mathbf{0}}}{k}-(\vec{\Delta}_0-\vec{\Delta}\right) \\ \frac{g(\vec{\boldsymbol{E}}\cdot\vec{\boldsymbol{P}})}{k}-\gamma_{||}(\Delta_{\mathbf{0}}-\Delta) \end{bmatrix} \qquad (4.25)$$

We now proceed to determine the constrained form;

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{x}}-\frac{\partial L}{\partial x}=0 \qquad (4.26)$$

Using Equation 3.25, the constrained form can be written and equation of motion can be observed.

After determining constrained form, equation of motion can be written as;

$$m\dot{v}=\frac{\dot{\boldsymbol{p}}}{m}+\frac{\lambda}{m}\dot{A} \qquad (4.27)$$

This yields the following equations of motion:

$$\dot{E} = -kE + gP$$

$$\dot{P} = -\gamma_{\perp}P + gE\,\Delta$$

$$\dot{\Delta} = -\gamma_{\parallel}(\Delta - \Delta_0) - 4gPE \tag{4.28}$$

The generalized equations of motion yielding the butterfly shaped attractor is given for $P=0$, which leads to chaotic behavior for certain choices of parameters, in particular the case for the operating point k=11.75, g=6.06, g⊥ =2.66, $g_{||}$=2.75, Δ_0 =28 is illustrated in

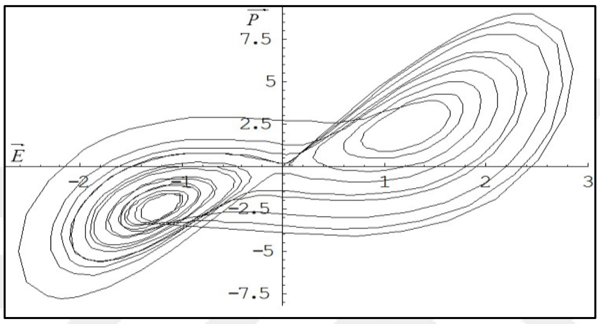

Figure 4.5. The basic operating point shows "Throw and catch" chaotic behavior similar to the Lorentz attractor.

Figure 4.5. The dominant chaotic behavior exhibited by the system is generated by the so-called “Throw and Catch” mechanism, mechanism [13] where an unstable fixed point is surrounded by two stable fixed points. More specifically, two linearized equilibrium point with a pair of complex conjugate eigenvalues with slightly positive real parts (assumed stable) surround an unstable third equilibrium point with eigenvalues (+ - -). The unstable point throws the system from the region of one stable point to the other one. Both the Lorenz model and this model has an unstable fixed point surrounded by two stable fixed points as described.

For the operating point indicated, the linearized eigenvalues for the unstable fixed point at the origin (0,0,0) are $\{-39.5920008645444, 25.1820008645444, -2.75\}$, while the surrounding fixed points are located at {E -> ±1.26036, P -> ±2.44376, Δ -> 0.851088} have the linearized eigenvalues {-18.4623, 0.651145 ±17.2217 I} that correspond to the throw catch mechanism.[14]

Such a phase diagram is indicative of a relatively rich bifurcation scheme including Hopf bifurcation and a t ypical bifurcation diagram involving the varied variable g against E where a branch point and two Hopf bifurcation points are indicated. The diagram has been generated by Kuznetsov’s MATHCONT software[15].

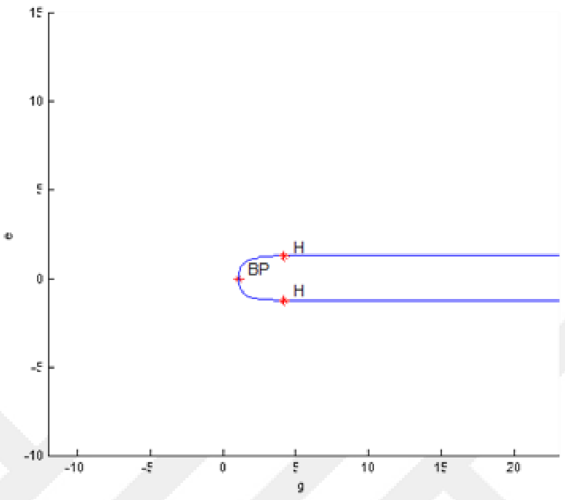

Figure 4.6. Bifurcation diagram of the Maxwell Bloch system

The transition from a regular system to a single center, throw and catch mechanism and a limit cycle like structure can clearly be seen as g is increased.

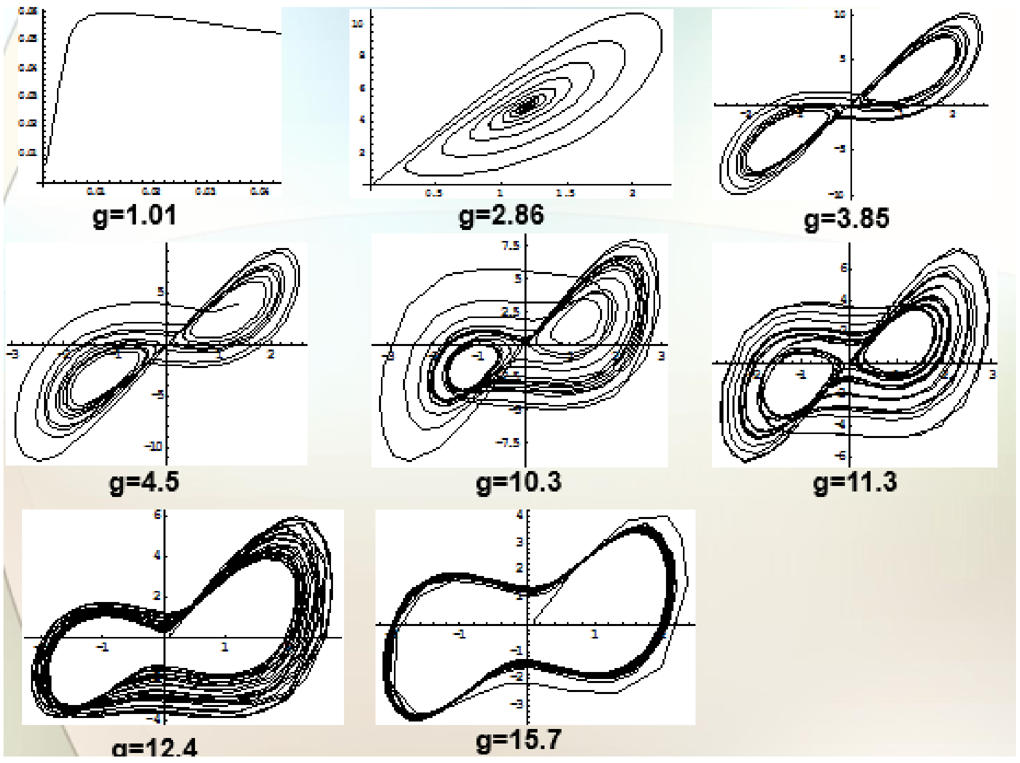

Figure 4.7. Attractors of the system as g is varied about the operating point

5. NONLINEAR TIME SERIES ANALYSIS

When one observes complicated behavior in nature, it is natural to look for a simple underlying cause. With one dimensional experimental data only, one asks whether the dynamics underlying the data involve a low dimensional chaotic system or nondeterministic and random components. In most cases, only a single sequence of measurements at successive times (a time series) would be available .

A natural phenomenon like river flow is highly complicated and most of the time treated as nondeterministic. Understanding the behavior of its underlying dynamics will lead to a more reliable base for choosing an appropriate modeling and prediction method. Some recent studies [16] showed that low-dimensional deterministic techniques can be applied as an alternative method for modeling and the results are encouraging. A complicated behavior in nature can be identified as deterministic and chaotic or nondeterministic and random, subjected to its underlying dynamics. Studies using low- dimensional deterministic techniques for modeling and prediction of river flow dynamics are attracting interest and producing encouraging results. Thus, low-dimensional deterministic techniques constitute an available alternative for studying river flow dynamics provided that sufficient care is exercised in their application and in interpreting the results [16].

Observational data obtained from natural phenomena adds another complication via measurement errors and scalar values which purely represents the underlying dynamical system. A well-known and widely used approach to overcome these difficulties is phase space reconstruction method. Based on the theorem of Takens, one can construct a phase space which successively resembles the global behavior of the original dynamical system from scalar measurements. A brief outline of the technique is given in Figure 5.1. When one observes complicated behavior in nature, one seeks a simple underlying cause. With only experimental data, one asks whether the dynamics are deterministic and chaotic or nondeterministic and random. In most cases, one might have only a single sequence of measurements at successive times.

.

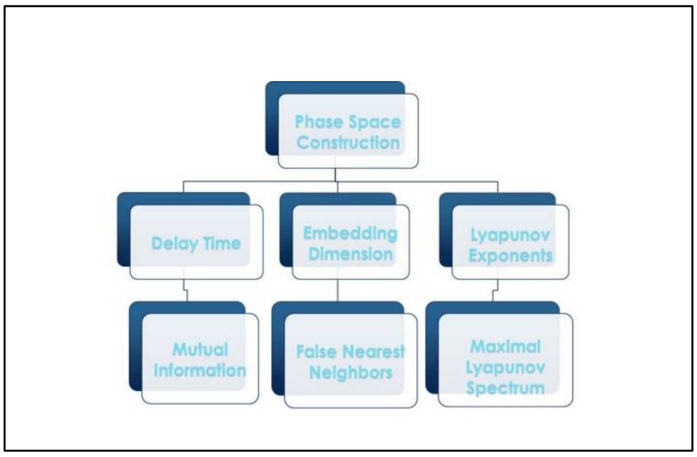

Figure 5.1. Steps of data analysis in time series

5.1 PHASE SPACE RECONSTRUCTION

As the scalar measurements are taken at arbitrary but equally spaced time intervals, a suitable delay time is the key point to preserve the global behavior of the dynamics. A small delay time can lead to a strongly correlated phase space vectors; on the other hand, information loss is inevitable if a large delay time value; the delay time can be estimated from a) Mutual information and b) autocorrelations [17].

5.1.1 Mutual Information

As discussed before in previous section delay time is an important ingredient of the step by step procedure to construct the phase space. If time delay is very short when it is taken reconstructed vectors' components will be too close to each other, causing the state space frame to be seen on the diagonal line, so loss of information about the real system will be had. If it is too long, important periodicity informaton will be lost.

In order to start the phase space reconstruction from the scalar flow rate $s_{nn}(k)$, where k is the time step, we need to construct the delay vector $\vec{y}_{nn}(k)$ given by;

$$\vec{y}_{nn}(k) = [s_{nn}(k),\ s_{nn}(k+\tau), s_{nn}(k+d-1)\tau)\ ] \qquad (5.1)$$

τ is the delay time and d means the embedding dimension. Time delay can be found from the first zero of the correlation function (linear criterion) or first minimum of the average mutual information [18].

The logic behind this approach can be summarized as follows. Let the underlying dynamical system be given by Equations (3.23a). Then using the Taylor expansion $s(t+a)=s(t)+a\left(\frac{ds}{dt}\right)+\ldots_t = s(t)+a\sum_n \frac{\partial s}{\partial x_n} f_n + \ldots$ we see that the one dimensional signal contains information about the underlying N dimensional system.

Using a very long delay time will cause the correlations between the parts of reconstructed vectors to be lost and signals will be errorly recognized as random. That’s why information between a random variable and another random variable is known mutual information.

One can only see the information sent to a channel and the information that is sent back from the channel. As an example, assume that X and Y are random variables with a common probability distibution P(X;Y). The interaction probability of observing x by a measurement of X and observing y by a measurement of Y, namely $P_{xy}(x;y)$, should be different from the product of the individual probabilities P(x) of measuring x out of the set X and P(y)of measuring y out of the set Y respectively. This implies that the two are not independent and the two sets are correlated. The logarithm of that ratio in bits is therefore called the average mutual information of X and Y and is given by

$$\log_2 \frac{P_{XY}(x;y)}{P(x)P(y)} \tag{5.2}$$

Thus we can sum over the sets X and Y and obtain the average mutual information as;

$$I(X;Y) = -\sum_x \sum_y P_{XY}(x;y) log_2 \frac{P_{XY}(x;y)}{P(x)P(y)} \tag{5.3}$$

To apply this formula to time series analysis, we take S(n) as set X and S(n+τ) as set Y. The average mutual information is then given by

$$I(\tau) = -\sum_x \sum_y P(s(n+\tau), s(n))) log_2 \left[\frac{P(s(n+\tau), s(n))}{P(s(n+\tau))P(s(n))}\right]$$

$$I(X;Y) = D(p(x,y)||p(x)p(y)) \tag{5.4}$$

$$I(X;Y) = \sum_x \sum_y p(x,y) log \frac{p(x,y)}{p(x)p(y)}$$

Here p(x) and p(y) are the probability distributions and the entropy is the logarithm of the ratio between the actual distribution and the distribution where the mutul informations are equal. Figure mutual shows the Delay time vs Information in bits of Sakarya river's branch Aktaş. Figure.mutual shows us the first minimum of data gives us delay time of the Aktas tributary which is 5. This calculations are made by using the TISEAN "Time Series Analysis package software in C++" [19]

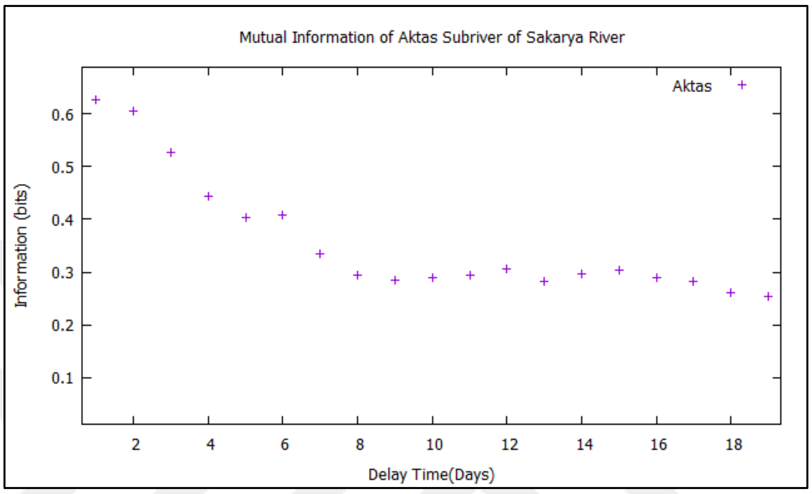

Figure 5.2. Mutual information of Aktaş Subriver of Sakarya River

To estimate minimum mutual information is found as five. The delay time is necessary for constructing the 2n+1 dimensional delay vector from the data We also need to determine the embedding dimension in order to construct the delay vector. If the embedding dimension is smaller than the proper dimension, points that are not neighbors on the original attractor will merge erroneously into the same neighborhood via projection to a lower dimensional space. Then finding false-nearest neighbors of all those points on embedded attractor requires increasing the dimensionality of the embedding space until the number of neighbors stabilize. Figure 4.3 shows mutual information of botbasi subriver.

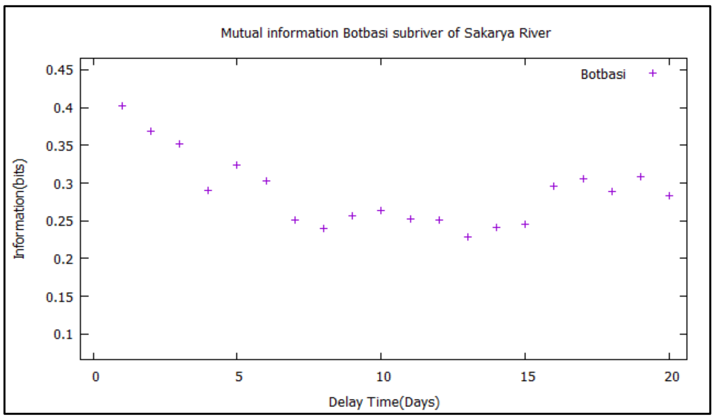

Figure 5.3. Mutual information of Botbaşı subriver of Sakarya River

As one can see in Figure 4.3. first minimum is four. The minima of the mutual information for all tributaries are given in Table 5.1 and range between 5 and 9.

5.1.2 False-Nearest Neighbors

One of the main problems of reconstructing a phase space from a scalar time series is choosing a suitable embedding dimension, which will at least topologically preserve the global properties of the dynamical system at least locally. Embedding dimension directly affects the attractor trajectory in the phase space, which alters the neighborhood of the points. If the embedding dimension is chosen to be smaller than the actual attractor dimension, projection of the trajectory will map false values into other neighborhoods of values; these are called the false neighbors. The calculation goes as follows: Choose a vector $\overrightarrow{R_i}$ constructed using the delay time suggested by mutual information and calculate the distance between its nearest neighbors $\overrightarrow{R_j}$ in an arbitrary dimension. Iterate this procedure for all the successive vectors and calculate R_i using the following equation.

$$R_i = \frac{|R_{i+1} - R_{j+1}|}{\|\overrightarrow{R_i} - \overrightarrow{R_j}\|} \tag{5.5}$$

A point of data is selected as a false neighbor if the distance, R_i exceeds a given threshold. A typical false neighbor's calculation is shown in Figure 5.4.

Displacement of (d+1) dimension and nearest neighbor's delayed time vector can be calculated as follows,

$$R_{d+1}(k)^2 = \sum_{m=1}^{d+1}[s_{nn}(k+(m-1)\tau) - s^{nn}(k+(m-1)\tau]^2 \tag{4.6}$$

Displacement of (d+1) dimension and displacement of d dimension R_{d+1}/R_d will be calculated. Where this ratio achieves a threshold points further increasing the dimensionality of the delay vector will no longer introduce false neighbors, since we are not embedding the system into a lower dimensional manifold.

$$\sqrt{\frac{R_{d+1}^2(k)-R_d^2(k)}{R_d^2(k)}} = \frac{|s(k+d\tau)-s^{nn}(k+d\tau)|}{R_d(k)} \tag{5.7}$$

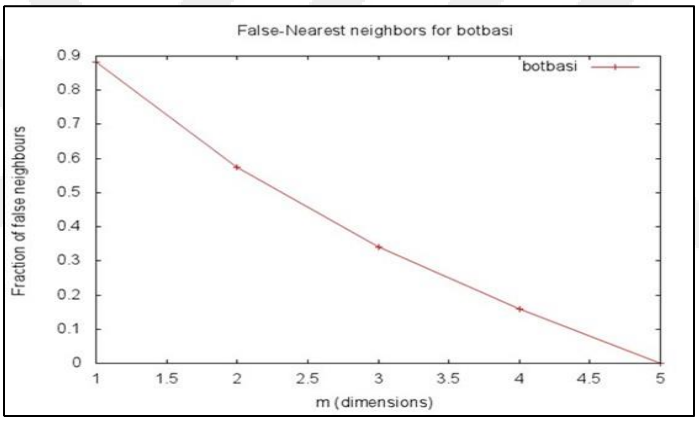

Figure 5.4. False-Nearest Neighbors for Botbasi branch of Sakarya River

5.1.3 Maximal Lyapunov Exponent

Lyapunov exponent is a measure of divergence or convergence of orbits in a phase space, which can also be calculated for a time series. As the reconstructed phase space preserves the topology of the underlying dynamics, Lyapunov exponents calculated for the embedded phase space will show the chaotic nature of the original attractor. The rate of exponential growth between the nearby trajectories is called as the maximal Lyapunov exponent and a positive rate indicates chaotic behavior. The following equation is used to calculate the stretching of the trajectories;

$$S(\epsilon,m,t) =< \ln \frac{1}{u_n} \sigma s_n \epsilon u_n |s_{n+t} - s_{n'+t}| > \tag{5.7}$$

$s_{n'}$ is a neighboring point to s_n in the phase space in the course of the attractor, ϵ is the box size. At a future time t the distance between these points will be $s_{n+t} - s_{n'+t}$. So the formula measures the growth of this distance in time from the initial distance. If *S (ϵ,m,t)* is linear for a range of iterations, the slope of this line gives an approximate value for the maximal Lyapunov exponent. If a robust increase, which is sufficient to determine its sign, is observed, this can be taken as an indicator of chaotic behavior.

5.1.4 BASIC TIME SERIES ANALYSIS PROCEDURE USING TISEAN

In order to perform time series analysis, attractor reconstruction and Lyapunov exponent calculation using TISEAN, the following steps should be carried out:

Using the autocorrelation function one gets a linear estimate of the delay time from its first zero. A better estimate of the delay time can be found from the mutual information by plotting it versus time. Using its first minimum, one can find a suitable delay time. The relevant TISEAN command is:

mutual: Estimates the time delayed mutual information of the data set. To estimate the embedding dimension, one finds the lowest dimension where fraction of the false nearest neighbors stabilize by plotting this fraction against embedding dimension. time. The relevant TISEAN command is:

false_nearest: Determines the fraction of false nearest neighbors.To estimate the Maximal Lyapunov Exponent, one uses the TISEAN commands lyap_k or lyap_r.

lyap_k: Estimates the maximal Lyapunov exponent using the Kantz algorithm.

lyap_r: Estimates the maximal Lyapunov exponent using the Rosenstein algorithm.

corr: To compute the autocorrelation of a scalar data set.

6. RESULTS AND DISCUSSION OF RIVER FLOW

Table.6.1 shows us mutual information values, embedding dimension values and Lyap$_r$ values which are the largest Lyapunov exponent of a given scalar data set using the algorithm of Rosenstein et al [20]; (this is claimed to be a better estimate for smaller data sets as explained below). Lyap$_k$ values are largest Lyapunov exponent of a given scalar data set using the algorithm of Kantz [21], with further explanation in [22].

Table 6.1. Nonlinear Time Series Analysis results of Sakarya River's tributaries

Branches of Sakarya river	**Mutual information**	**Embedding dimension**	**Lyap$_r$ values**	**Lyap$_k$ values**
Aktas	5	6	0.012	0.008
Besdegirmen	4	6	0.018	0.021
Botbasi	4	5	0.014	0.016
Dogancay	4	4	0.016	0.012
Dokurcan	4	5	0.016	0.016
Hamidiye	7	6	0.015	0.008
Karakoy	4	6	0.021	0.034
Kargi	9	4	0.010	0.012
Kocasu	4	6	0.018	0.030
Mesecik	8	6	0.009	0.020
Taksirkopru	4	11	0.015	0.039

The algorithm proposed by Kantz establishes that the divergence rate trajectories fluctuate along the trajectory, with the fluctuation given by the spectrum of effective Lyapunov exponents. Rosenstein et al. have proposed a similar algorithm where the distance between the trajectories is defined as the Euclidian norm in the reconstructed phase space and they have also used only one neighbor trajectory. The algorithm suggested by Rosenstein is more effective when the number of data is relatively small. In our study, the results obtained from each algorithm are in parallel with each other. A typical Lyapunov Exponent by stretching exponent calculation using the Rosenstein approach is illustrated in Figure.6.2, while the calculation using the standard Kantz approach is illustrated in Figure 6.3. The statistical issues involved in the selection of the approach are discussed extensively in [23] and [24].

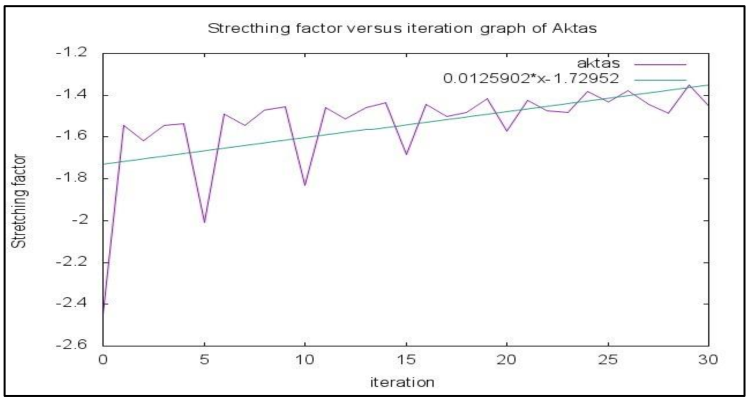

Figure 6.1. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Aktas branch

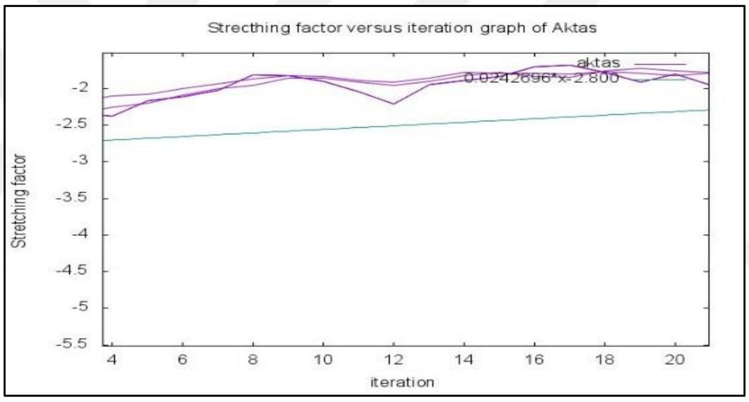

Figure 6.2. Stretching factor v.s. iteration graph using the Kantz Algorithm for Aktas branch

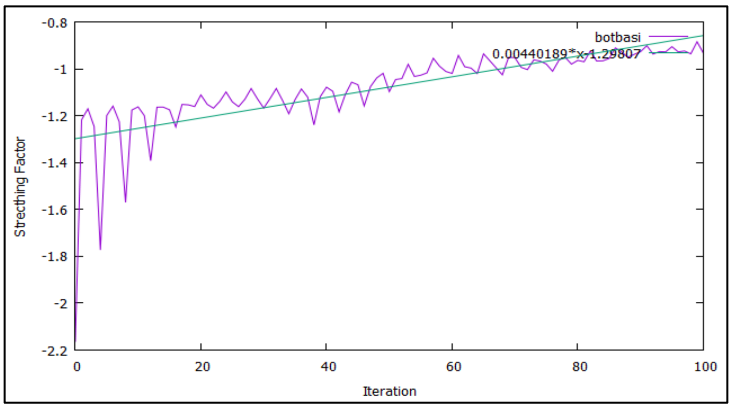

Figure 6.3. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Botbasi branch

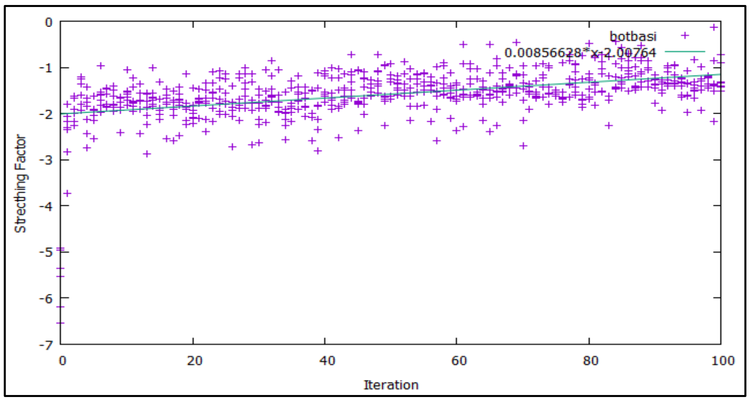

Figure 6.4. Stretching factor v.s. iteration graph using the Kantz Algorithm for Botbasi branch

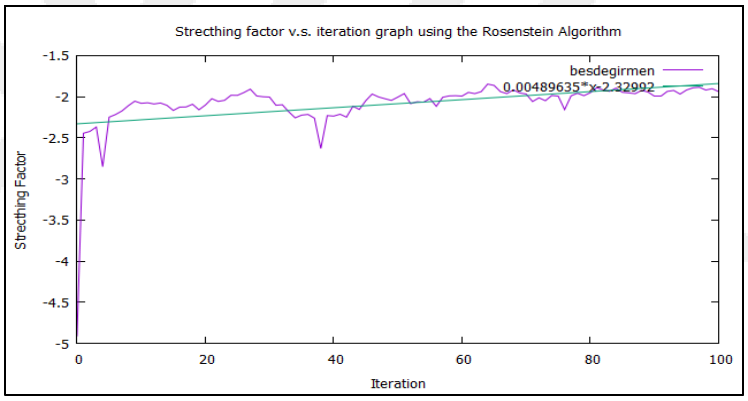

Figure 6.5. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Besdegirmen branch

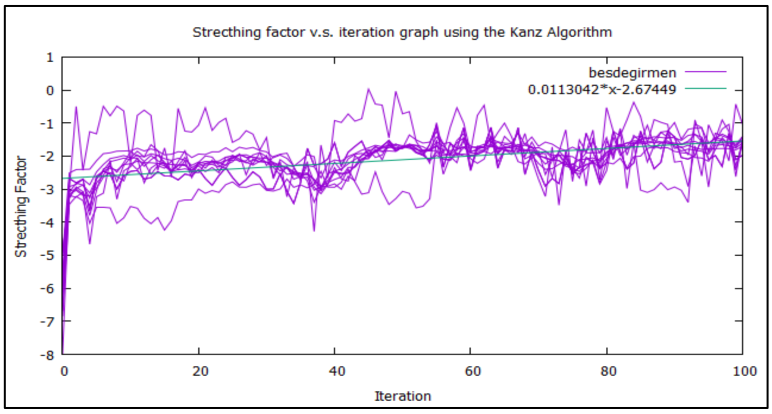

Figure 6.6. Stretching factor v.s. iteration graph using the Kantz Algorithm for Besdegirmen branch

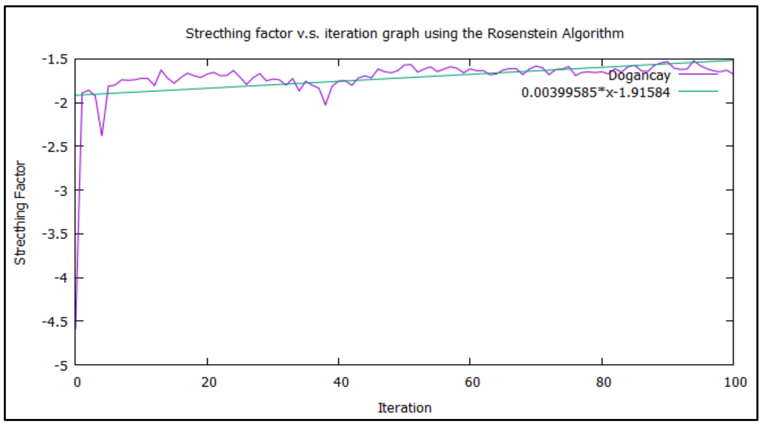

Figure 6.7. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Dogancay branch

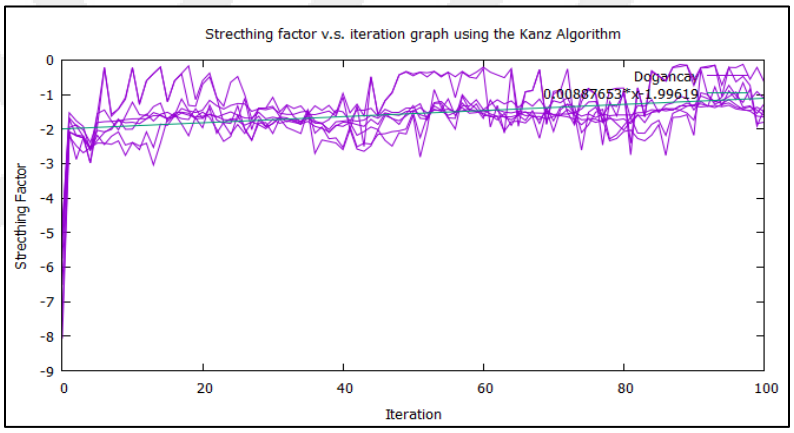

Figure 6.8. Stretching factor v.s. iteration graph using the Kanz Algorithm for Dogancay branch

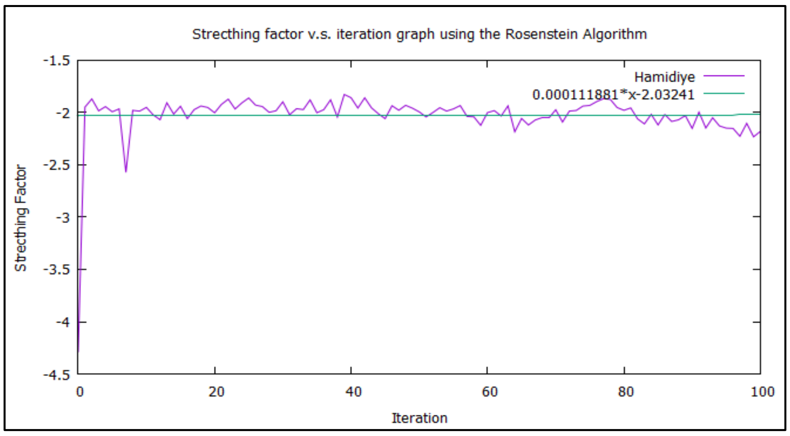

Figure 6.9. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Hamidiye branch.

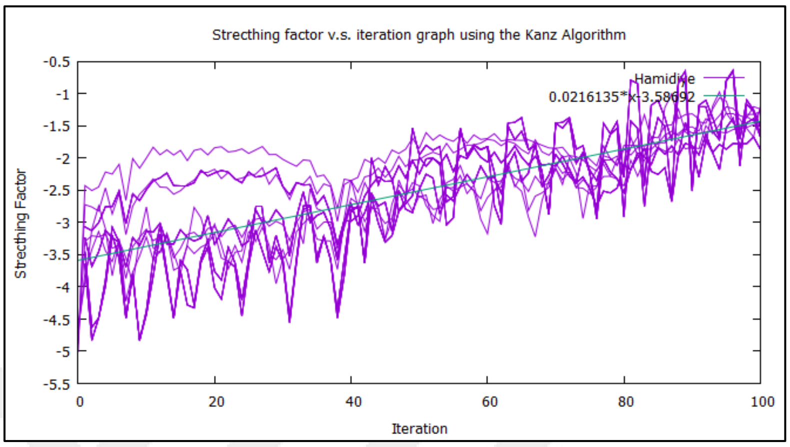

Figure 6.10. Stretching factor v.s. iteration graph using the Kantz Algorithm for Hamidiye branch.

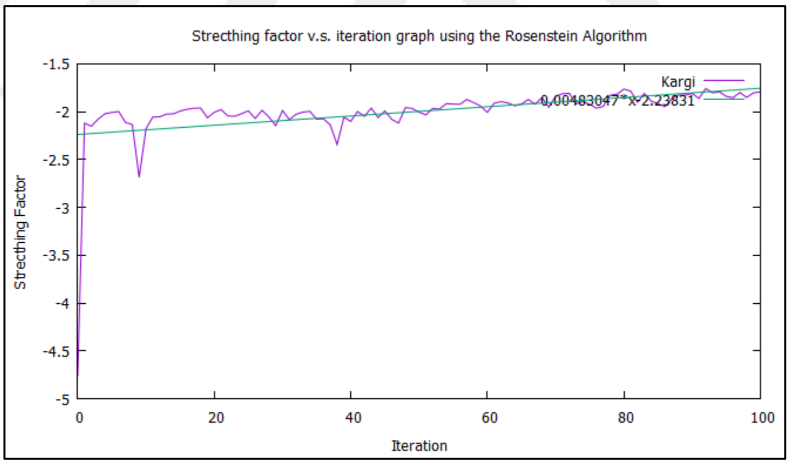

Figure 6.11. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Kargi branch.

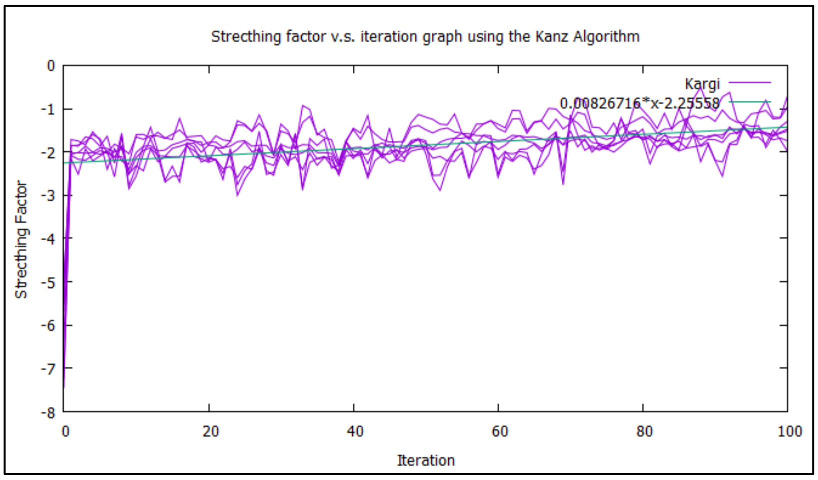

Figure 6.12. Stretching factor v.s. iteration graph using the Kantz Algorithm for Kargi branch.

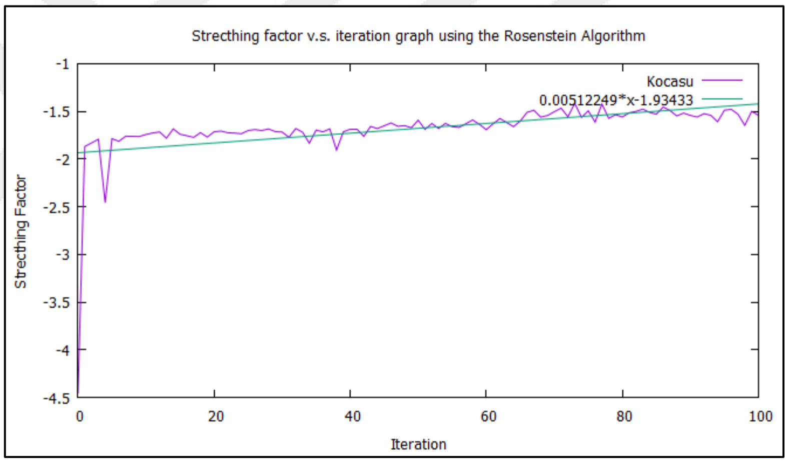

Figure 6.13. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Kocasu branch.

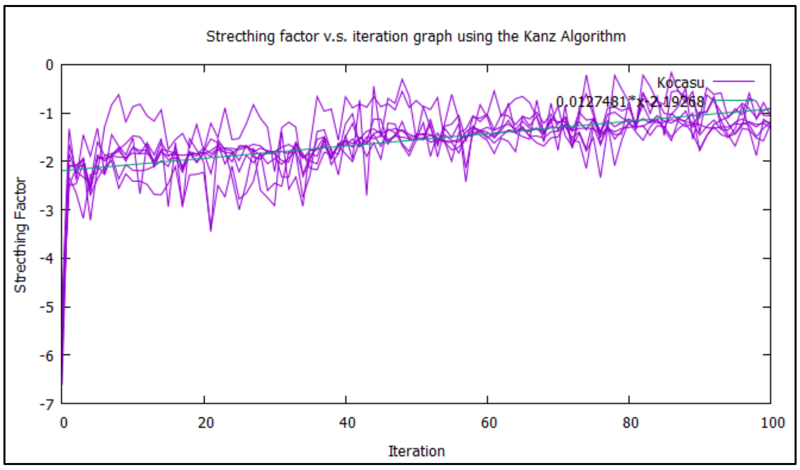

Figure 6.14. Stretching factor v.s. iteration graph using the Kantz Algorithm for Kocasu branch.

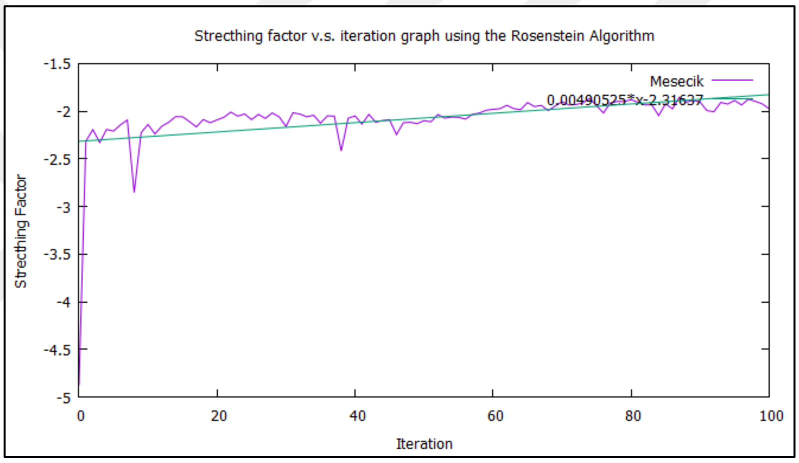

Figure 6.15. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Mesecik branch

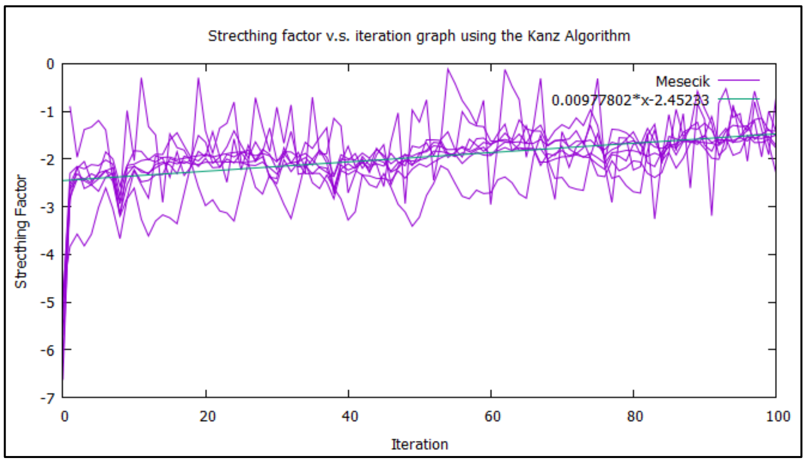

Figure 6.16. Stretching factor v.s. iteration graph using the Kantz Algorithm for Mesecik branch

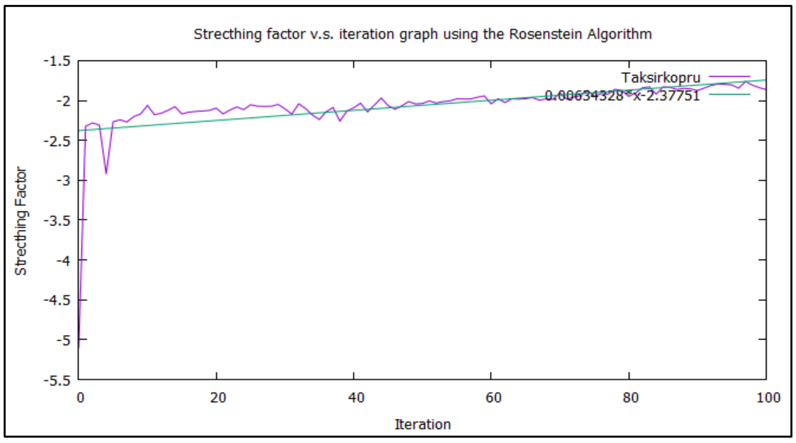

Figure 6.17. Stretching factor v.s. iteration graph using the Rosenstein Algorithm for Taksirkpopru branch

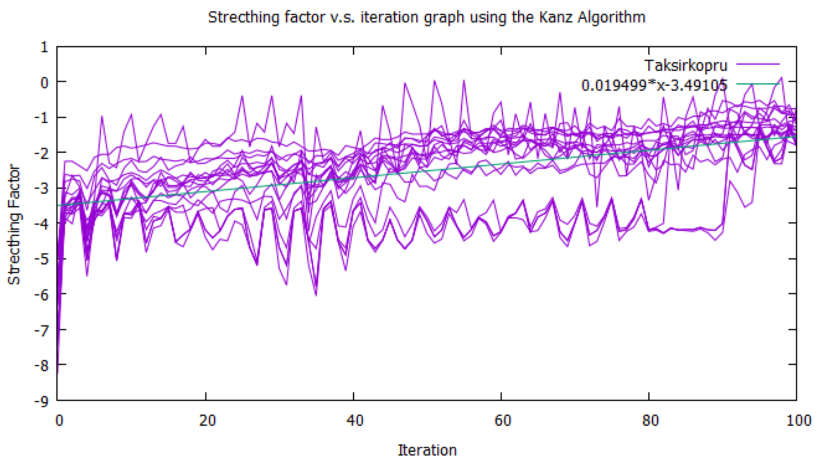

Figure 6.18. Stretching factor v.s. iteration graph using the Kanz Algorithm for Taksirkpopru branc

7. CONCLUSION

Understanding the dynamics of river flow is crucial to select a feasible modeling method to forecast river discharge. In this study, phase space reconstruction method is used to obtain a depiction of the underlying dynamics, which will preserve the global invariants of the system. Maximal Lyapunov exponent, which is a very strong evidence for chaotic behavior for eleven branches of the Sakarya River has been calculated. These results are encouraging for applying chaotic modeling routines instead of probabilistic methods.

As a result, monthly mean flow values of Sakarya River show chaotic behavior as quantified by the maximal Lyapunov Exponent. That may imply that the river has no long time trend, which is observed by [24]. That seems to be due to the dams on the river and the increasing use of the river's water by the people and also climate changes. According to article [25], Benue River in Nigeria has a comparable trend but no low dimensional phase space chaotic dynamics has been observed there. Therefore, we can say Sakarya River has limited future for electricity from dams and other human exploitation because of its chaotic dynamics. We have studied the article by S. Isik et al. [16] that reaches the same conclusions using quasi-linear time series analysis methods from regular statistical analysis. This work corroborates the findings and finally demonstrates that the phenomenon may be better understood by nonlinear time series analysis than stochastic techniques.

Maxwell-Bloch equations which is related to Lorenz type systems was transformed into Lagrangian form and equations of motions are constructed in this formalism. The equations of motion can be used to analyze situations under this system is going to chaos.

Then again, Hamiltonian systems protect the stage space volume along these lines a butterfly sort attractor can not be straightforwardly communicated in this notation. The benefit of utilizing the present formalism empowers one to attempt the wide assortment of instruments accessible for building approximate solutions of Hamiltonian systems.The below results obtained numerically by the use of a Fortran code that include wolf algorithm in order to determine trajectories and lyapunov exponents of dynamical system, and a reduce code which calculates the variational equations needed for wolf algorithm. In the equation of motions the parameter p is kept constant and the coefficient of oscillator term g varied from 0 to 2. It is seen that regular motion become dominant when the coefficient of oscillatory term increased. We also analyze maximal lyapunov exponents of the system under the specific range of parameter. It is observed that when the parameter g is converging to zero lyapunov exponents gets bigger. Some of the phase space trajectories and lyapunov exponents were presented to illustrate this.

REFERENCES

1. E. N. Lorenz ,Deterministic Nonperiodic Flow,*Journal of the Atmospheric Sciences*, 1963,

2. F. Morrison , *The Art of Modeling Dynamic System*, Dover, 2008

3. Directorate of Electrical Power Resources survey and Development Administration, "Monthly Mean Flows of Turkey Rivers (1935-2000)", EIE (2003)

4. M Peil, M. Jacquot, Y.K. Chembo, L. Larger, T. Erneux, Route to Chaos and Multiple Time Scale Dynamics in Broadband Bandpass Nonlinear Delay Electro-Optic Oscillators, *Physical Review E 79* : 026208,2009

5. J.M. Matinerie, A.M. Albano, A.I. Mees, P.E. Rapp, Mutual Information, Strange Atrractors and the Optimal Estimation of Dimension, *Physical Review A45*, 10: 7058-7064, 1992.

6. W. Ditto and L. Pecora , Mastering Chaos , *Scientific American*, 78-84,August 1993

7. H.W. Broer KAM Theory: The Legacy of Kolmogorov's 1954 Paper, *University of Groningen*, 2003

8. T.S. Biro, S.G. Matinyan, B. Müller, Chaos and Gauge Field Theory, *World Scientific* , 56, 1994.

9. O.O. Aybar, A.S. Hacinliyan, I. Kusbeyzi Aybari, K. Koseyan, B. Deruni, Stability and Chaos in a Classical Yang-Mills-Higgs System", *Chaotic Modelling and Simulation*, 6: *215-221, 2013.*

10. A. Wolf, J.B. Swift, H.L. Swinney and J.A Vastano, Determining Lyapunov Exponents from a Time Series, *Physica D*, 16:285-317, 1985.

11. W. H. Press S.A. Teukolsky , W. T. Vetterling, B. P. Flannery,"Numerical Recipes in Fortran 77, Second Edition", *Cambridge University Press*, 521-43064-X ,1992

12. M.J. Finn, *Classical Mechanics*. Infinity Science Press, 2008

13. Aslı Umur, Local Non-Perturbative Methods for Algebraic Calculations of Lyapunov Exponents, *M.S. Thesis Bogazici University*, 1997.

14. A.S. Hacinliyan, E.E. Akkaya, I. Kusbeyzi, O.O Aybar, Maxwell-Bloch Equations as Predator-Prey System, *Second Chaotic Modeling and Simulation International Conference,* Chania Grecee 2009.

15. *Y.A.* Kuznetsov*, Elements of Applied Bifurcation Theory, Springer, 1995.*

16. S. Isik , E. Dogan , L. Kalin , M. Sasal and N. Agiralioglu , Effects of Anthropogenic Activities on the Lower Sakarya River. *Catena*, 75:172-181, Elsevier Publishing 2008

17. H.D.I. Abarbanel, R. Brown, J.J. Sidonowich, L.S. Tsimring, The Analysis of Observed Chaotic Data, *Reviews of Modern Physics*, 65, 4:1331-1392, 1993.

18. A. M. Fraser and H.L. Swinney, Independent Coordinates for Strange Attractors from Mutual Information, *Phyical Review A*, 33:1134-1140, 1986.

19. R. Hegger , H. Kantz , and T. Schreiber , Practical Implementation of Nonlinear Time Series Methods: The TISEAN Package, *CHAOS* 9: 413, 1999

20. M.T. Rosenstein , J.J. Collins and C.J. De Luca , A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets, *Physics D* 65: 117, 1993

21. H. Kantz , A Robust Method to Estimate the Maximal Lyapunov Exponent of a Time Series. *Physical Letter A* 185:77, 1994

22. B. Pompe , Measuring Statistical Dependences in a Time Series *Journal of Statistical Physics*, 73:587, 1993

23. M. Palus, Testing for Nonlinearity Using Redundancies: Quantitative and Qualitative Aspects, *Physica D* 80:186, 1995

24. A. Atalay and C. Ikiel . Trend Analysis of Monthly and Annual Flow Values of Sakarya River (Turkey). *GEOMED 2007 International Symposium*,2007

25. O. Martins, M.A. Sadeeq and I.E. Ahanequ , Nonlinear Deterministic Chaos in Benue River Flow Daily Time Sequence, *Journal of Water Resource and Protection*, 3:747-757, 2011

26. H.A. Yildirim, E.E. Akkaya, A.S. Hacinliyan, G. Sahin, Maxwell-Bloch Equations a Lorenz Type Chaotic System in Lagrangian Form, *Chaotic Modelling and Simulation International Conference*, 749-751, 2013.

27. G. Witvoet, Control of Chaotic Dynamical Systems using OGY, *Traineeship Report Technische Universiteit Eindhoven*, DCT 2005.36

28. T. S. Parker, L. O Chua, Practical Numerical Algorithms for Chaotic System, *Springer Verlag*, 0-387-96688-9, 1989

29. B. Sivakumar, Nonlinear Determinism in River Flow: Prediction as a Possible Indicator, *Wiley Earth Surface Processes and Landforms*, 32,7:969-979, 2007.

30. B. Sivakumar, V.P. Singh, Hyrologic System Complexity and Nonlinear Dynamic Concepts for a Catchment Classification Framework, *Hydrology and Earth System Sciences*, 16: 4119-4131, 2012

31. B. Sivakumar, Dynamics of Sediment Transport in the Missisippi River Basin: A Temporal Scaling Analysis, *MODSIM* , A05, 2007

32. J. Wu, J. Lu, J. Wang, Application of Chaos and Fractal Models to Water Quality Time Series Prediction, *Elsevier Environmental Modelling and Software*, 24: 632-636, 2009.

33. E. N. Lorenz, Designing Chaotic Models, *Journal of the Atmospheric Sciences*, 62: 1547-1587, 2004

34. M.A. Ghorbani , R. Daneshfaraz, H. Arvanagi, A. Pourzangbar, S. M Saghebian, S. K. Kar, Local Prediction in River Discharge Time Series, *Journal of Civil Engineering and Urbanism*, 2:51-55, 2012

APPENDIX A: MUTUAL INFORMATION C++ CODE

In Mutual Information part delay time is calculated by the use of TISEAN software package. Estimates the time delayed mutual information of the data. It is the simplest possible realization. It uses a fixed mesh of boxes. No finite sample corrections are implemented so far. This package is called by typing mutual.exe in command prompt then type data which is time series then output file name is created. Result calculations are collected in given name output file in computer documentary.

Using C++ code below, one can enter following ;

Enter the command line.

“ mutual.exe filename.dat -D (maximal time delay example 20) -ofilenamemut.dat”

The first line contains the number of occupied boxes, the second one the shannon entropy (normalized to the number of occupied boxes), the last D lines the mutual information (first column: delay, second column: mutual information).

Here is the C++ code below:

```
/*Author: Rainer Hegger. Last modified, Sep 20, 2000 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>
#include <string.h>
#include "routines/tsa.h"

#define WID_STR "Estimates the time delayed mutual information\n\t\
of the data set"


char *file_out=NULL,stout=1;
char *infile=NULL;
unsigned long length=ULONG_MAX,exclude=0;
unsigned int column=1;
unsigned int verbosity=0xff;
long partitions=16,corrlength=20;
long *array,*h1,*h11,**h2;

void show_options(char *progname)
{
  what_i_do(progname,WID_STR);
  fprintf(stderr," Usage: %s [Options]\n\n",progname);
  fprintf(stderr," Options:\n");
  fprintf(stderr,"Everything not being a valid option will be interpreted"
          " as a possible"
          " datafile.\nIf no datafile is given stdin is read. Just - also"
          " means stdin\n");
  fprintf(stderr,"\t-l # of points to be used [Default is all]\n");
  fprintf(stderr,"\t-x # of lines to be ignored [Default is 0]\n");
  fprintf(stderr,"\t-c column to read  [Default is 1]\n");
  fprintf(stderr,"\t-b # of boxes [Default is 16]\n");
  fprintf(stderr,"\t-D max. time delay [Default is 20]\n");
  fprintf(stderr,"\t-o output file [-o without name means 'datafile'.mut;"
              "\n\t\tNo -o means write to stdout]\n");
  fprintf(stderr,"\t-V verbosity level [Default is 1]\n\t\t"
          "0='only panic messages'\n\t\t"
          "1='+ input/output messages'\n");
  fprintf(stderr,"\t-h  show these options\n");
  fprintf(stderr,"\n");
  exit(0);
}

void scan_options(int n,char** in)
{
  char *out;

  if ((out=check_option(in,n,'l','u')) != NULL)
    sscanf(out,"%lu",&length);
  if ((out=check_option(in,n,'x','u')) != NULL)
    sscanf(out,"%lu",&exclude);
  if ((out=check_option(in,n,'c','u')) != NULL)
    sscanf(out,"%u",&column);
  if ((out=check_option(in,n,'b','u')) != NULL)
    sscanf(out,"%lu",&partitions);
  if ((out=check_option(in,n,'D','u')) != NULL)
    sscanf(out,"%lu",&corrlength);
  if ((out=check_option(in,n,'V','u')) != NULL)
    sscanf(out,"%u",&verbosity);
  if ((out=check_option(in,n,'o','o')) != NULL) {
    stout=0;
    if (strlen(out) > 0)
      file_out=out;
  }
}

double make_cond_entropy(long t)
{
  long i,j,hi,hii,count=0;
  double hpi,hpj,pij,cond_ent=0.0,norm;

  for (i=0;i<partitions;i++) {
    h1[i]=h11[i]=0;
    for (j=0;j<partitions;j++)
      h2[i][j]=0;
  }
  for (i=0;i<length;i++)
    if (i >= t) {
      hii=array[i];
      hi=array[i-t];
      h1[hi]++;
      h11[hii]++;
      h2[hi][hii]++;
      count++;
    }

  norm=1.0/(double)count;
  cond_ent=0.0;

  for (i=0;i<partitions;i++) {
    hpi=(double)(h1[i])*norm;
    if (hpi > 0.0) {
      for (j=0;j<partitions;j++) {
            hpj=(double)(h11[j])*norm;
            if (hpj > 0.0) {
              pij=(double)h2[i][j]*norm;
              if (pij > 0.0)
                cond_ent += pij*log(pij/hpj/hpi);
            }
      }
    }
  }

  return cond_ent;
}

int main(int argc,char** argv)
{
  char stdi=0;
  long tau,i;
  double *series,min,interval,shannon;
  FILE *file;

  if (scan_help(argc,argv))
    show_options(argv[0]);

  scan_options(argc,argv);
#ifndef OMIT_WHAT_I_DO
  if (verbosity&VER_INPUT)
    what_i_do(argv[0],WID_STR);
#endif

  infile=search_datafile(argc,argv,&column,verbosity);
  if (infile == NULL)
    stdi=1;

  if (file_out == NULL) {
    if (!stdi) {
      check_alloc(file_out=(char*)calloc(strlen(infile)+5,(size_t)1));
      strcpy(file_out,infile);
      strcat(file_out,".mut");
    }
    else {
      check_alloc(file_out=(char*)calloc((size_t)10,(size_t)1));
      strcpy(file_out,"stdin.mut");
    }
  }
  if (!stout)
```

```
    test_outfile(file_out);

  series=(double*)get_series(infile,&length,exclude,column,verbosity);
  rescale_data(series,length,&min,&interval);

  check_alloc(h1=(long *)malloc(sizeof(long)*partitions));
  check_alloc(h11=(long *)malloc(sizeof(long)*partitions));
  check_alloc(h2=(long **)malloc(sizeof(long *)*partitions));
  for (i=0;i<partitions;i++)
    check_alloc(h2[i]=(long *)malloc(sizeof(long)*partitions));
  check_alloc(array=(long *)malloc(sizeof(long)*length));
  for (i=0;i<length;i++)
    if (series[i] < 1.0)
      array[i]=(long)(series[i]*(double)partitions);
    else
      array[i]=partitions-1;
  free(series);

  shannon=make_cond_entropy(0);
  if (corrlength >= length)
    corrlength=length-1;

  if (!stout) {
    file=fopen(file_out,"w");
    if (verbosity&VER_INPUT)
      fprintf(stderr,"Opened %s for writing\n",file_out);
    fprintf(file,"#shannon= %e\n",shannon);
    fprintf(file,"%d %e\n",0,shannon);
    for (tau=1;tau<=corrlength;tau++) {
      fprintf(file,"%ld %e\n",tau,make_cond_entropy(tau));
      fflush(file);
    }
    fclose(file);
  }
  else {
    if (verbosity&VER_INPUT)
      fprintf(stderr,"Writing to stdout\n");
    fprintf(stdout,"#shannon= %e\n",shannon);
    fprintf(stdout,"%d %e\n",0,shannon);
    for (tau=1;tau<=corrlength;tau++) {
      fprintf(stdout,"%ld %e\n",tau,make_cond_entropy(tau));
      fflush(stdout);
    }
  }

  return 0;
}
```

APPENDIX B: FALSE NEAREST C++ CODE

This program looks for the nearest neighbors of all data points in m dimensions and iterates these neighbors one step into the future. If the ratio of the distance of the iteration and that of the nearest neighbor exceeds a given threshold the point is marked as a wrong neighbor. The output is the fraction of false neighbors for the specified embedding dimensions. Program implemented a new second criterion. If the distance to the nearest neighbor becomes smaller than the standard deviation of the data devided by the threshold, the point is omitted as mentioned before in embedding dimension part. This turns out to be a stricter criterion, but can show the effect that for increasing embedding dimensions the number of points which enter the statistics is so small, that the whole statistics is meanlingless.

In command prompt, one can call by typing false_ nearest.exe then one must call original time series data by typing data file name then create a new file name therefore, calculated false nearest dimensions collected in this new file name data in directory.

Finding embedding time of applied data of Sakarya River by using TISEAN false_nearest C++ code. Code is given below, code can be used by typing;

```
false_nearest.exe filename.dat -ofilenamefal.dat
```

```
/*Author: Rainer Hegger. Last modified: Sep 3, 1999 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <math.h>
#include "routines/tsa.h"

#define WID_STR "Determines the fraction of false nearest neighbors."

char *outfile=NULL;
char *infile=NULL;
char stdo=1;
unsigned long length=ULONG_MAX,exclude=0,theiler=0;
unsigned int column=1,delay=1,maxdim=5,mindim=1;
unsigned int verbosity=0xff;
double rt=10.0;
double eps0=1.0e-5;
double *series;
double aveps,vareps;
double varianz;

#define BOX 1024
int ibox=BOX-1;
```

```
long **box,*list;
unsigned long toolarge;

void show_options(char *progname)
{
  what_i_do(progname,WID_STR);
  fprintf(stderr," Usage: %s [options]\n",progname);
  fprintf(stderr," Options:\n");
  fprintf(stderr,"Everything not being a valid option will be interpreted"
          " as a possible"
          " datafile.\nIf no datafile is given stdin is read. Just - also"
          " means stdin\n");
  fprintf(stderr,"\t-l # of data [default: whole file]\n");
  fprintf(stderr,"\t-x # of lines to ignore [default: 0]\n");
  fprintf(stderr,"\t-c column to read [default: 1]\n");
  fprintf(stderr,"\t-m minimal embedding dimension [default: 1]\n");
  fprintf(stderr,"\t-M maximal embedding dimension [default: 5]\n");
  fprintf(stderr,"\t-d delay [default: 1]\n");
  fprintf(stderr,"\t-f escape factor [default: 10.0]\n");
  fprintf(stderr,"\t-t theiler window [default: 0]\n");
  fprintf(stderr,"\t-o output file [default: 'datafile'.fnn; without -o"
             " stdout]\n");
  fprintf(stderr,"\t-V verbosity level [default: 3]\n\t\t"
          "0='only panic messages'\n\t\t"
          "1='+ input/output messages'\n\t\t"
          "2='+ information about the current state\n");
  fprintf(stderr,"\t-h show these options\n");
  exit(0);
}

void scan_options(int n,char **in)
{
  char *out;

  if ((out=check_option(in,n,'l','u')) != NULL)
    sscanf(out,"%lu",&length);
  if ((out=check_option(in,n,'x','u')) != NULL)
    sscanf(out,"%lu",&exclude);
  if ((out=check_option(in,n,'c','u')) != NULL)
    sscanf(out,"%u",&column);
  if ((out=check_option(in,n,'m','u')) != NULL)
    sscanf(out,"%u",&mindim);
  if ((out=check_option(in,n,'M','u')) != NULL)
    sscanf(out,"%u",&maxdim);
  if ((out=check_option(in,n,'d','u')) != NULL)
    sscanf(out,"%u",&delay);
  if ((out=check_option(in,n,'f','f')) != NULL)
    sscanf(out,"%lf",&rt);
  if ((out=check_option(in,n,'t','u')) != NULL)
    sscanf(out,"%lu",&theiler);
  if ((out=check_option(in,n,'V','u')) != NULL)
    sscanf(out,"%u",&verbosity);
  if ((out=check_option(in,n,'o','o')) != NULL) {
    stdo=0;
    if (strlen(out) > 0)
      outfile=out;
  }
```

```
}

char find_nearest(long n,unsigned int dim,double eps)
{
  int x,y,x1,x2,y1,i,i1;
  long element,which= -1;
  double dx,maxdx,mindx=1.1,factor;

  x=(int)(series[n-(dim-1)*delay]/eps)&ibox;
  y=(int)(series[n]/eps)&ibox;

  for (x1=x-1;x1<=x+1;x1++) {
    x2=x1&ibox;
    for (y1=y-1;y1<=y+1;y1++) {
      element=box[x2][y1&ibox];
      while (element != -1) {
            if (labs(element-n) > theiler) {
              maxdx=fabs(series[n]-series[element]);
              for (i=1;i<dim;i++) {
                i1=i*delay;
                dx=fabs(series[n-i1]-series[element-i1]);
                if (dx > maxdx)
                  maxdx=dx;
              }
              if ((maxdx < mindx) && (maxdx > 0.0)) {
                which=element;
                mindx=maxdx;
              }
            }
            element=list[element];
      }
    }
  }

  if ((which != -1) && (mindx <= eps) && (mindx <= varianz/rt)) {
    aveps += mindx;
    vareps += mindx*mindx;
    factor=fabs(series[n+1]-series[which+1])/mindx;
    if (factor > rt)
      toolarge++;
    return 1;
  }
  return 0;
}

int main(int argc,char **argv)
{
  char stdi=0;
  FILE *file=NULL;
  double min,inter,epsilon,av;
  char *nearest,alldone;
  long i;
  unsigned int dim;
  unsigned long donesofar;

  if (scan_help(argc,argv))
    show_options(argv[0]);
  scan_options(argc,argv);
#ifndef OMIT_WHAT_I_DO
  if (verbosity&VER_INPUT)
    what_i_do(argv[0],WID_STR);
#endif

  infile=search_datafile(argc,argv,&column,verbosity);
  if (infile == NULL)
    stdi=1;

  if (outfile == NULL) {
    if (!stdi) {
      check_alloc(outfile=(char*)calloc(strlen(infile)+5,(size_t)1));
      strcpy(outfile,infile);
      strcat(outfile,".fnn");
    }
    else {
      check_alloc(outfile=(char*)calloc((size_t)10,(size_t)1));
      strcpy(outfile,"stdin.fnn");
    }
  }
  if (!stdo)
    test_outfile(outfile);

  series=(double*)get_series(infile,&length,exclude,column,verbosity);
  rescale_data(series,length,&min,&inter);
  variance(series,length,&av,&varianz);

  check_alloc(list=(long*)malloc(sizeof(long)*length));
  check_alloc(nearest=(char*)malloc(length));
  check_alloc(box=(long**)malloc(sizeof(long*)*BOX));
  for (i=0;i<BOX;i++)
    check_alloc(box[i]=(long*)malloc(sizeof(long)*BOX));

  if (!stdo) {
    file=fopen(outfile,"w");
    if (verbosity&VER_INPUT)
      fprintf(stderr,"Opened %s for writing\n",outfile);
  }
  else {
    if (verbosity&VER_INPUT)
      fprintf(stderr,"Writing to stdout\n");
  }

  for (dim=mindim;dim<=maxdim;dim++) {
    epsilon=eps0;
    toolarge=0;
    alldone=0;
    donesofar=0;
    aveps=0.0;
    vareps=0.0;
    for (i=0;i<length;i++)
      nearest[i]=0;
    if (verbosity&VER_USR1)
      fprintf(stderr,"Start for dimension=%u\n",dim);
    while (!alldone && (epsilon < 2.*varianz/rt)) {
```

```
    alldone=1;
    make_box(series,box,list,length-1,BOX,dim,delay,epsilon);
    for (i=(dim-1)*delay;i<length-1;i++)
          if (!nearest[i]) {
            nearest[i]=find_nearest(i,dim,epsilon);
            alldone &= nearest[i];
            donesofar += (unsigned long)nearest[i];
          }
    if (verbosity&VER_USR1)
          fprintf(stderr,"Found %lu up to epsilon=%e\n",donesofar,epsilon*inter);
    epsilon*=sqrt(2.0);
    if (!donesofar)
          eps0=epsilon;
  }
  if (donesofar == 0) {
    fprintf(stderr,"Not enough points found!\n");
    exit(FALSE_NEAREST_NOT_ENOUGH_POINTS);
  }
  aveps *= (1./(double)donesofar);
  vareps *= (1./(double)donesofar);
  if (stdo) {
    fprintf(stdout,"%u %e %e %e\n",dim,(double)toolarge/(double)donesofar,
                aveps,vareps);
    fflush(stdout);
  }
  else {
    fprintf(file,"%u %e %e %e\n",dim,(double)toolarge/(double)donesofar,
                aveps,vareps);
    fflush(file);
  }
}
if (!stdo)
  fclose(file);

if (infile != NULL)
  free(infile);
if (outfile != NULL)
  free(outfile);
free(series);
free(list);
free(nearest);
for (i=0;i<BOX;i++)
  free(box[i]);
free(box);
return 0; }
```

APPENDIX C: LYAPUNOV EXPONENTS C++ CODES

Lyapunov exponent are calculated in two different algorithms. One is with Kantz algorithm that is called Lyap_k.

A n-dimensional system will have n Lyapunov exponents. The Lyapunov exponents are used to study the stability of a system, e. g., a fixed point has only negative Lyapunov exponents, periodic systems have one zero and else negative Lyapunov exponents, and chaotic systems have at least one positive Lyapunov exponent.

To estimate the Lyapunov exponent of time series, several approaches were suggested, like the methods of Wolff, Kantz or Rosenstein. Here we will use the method of Rosenstein provided by the TISEAN toolbox.

Let us consider the Henon system as a typical example code,

C++ source code is,

```
a = 2;
b = 0;
x(1,1) = .91;
x(1,2) = 0;
for i = 2:10001
x(i,1) = 1 - a * x(i-1,1)^2 + b * x(i-1,2);
x(i,2) = x(i-1,1);
end x(1,:) = []; save henon.dat x -ascii -tabs
```

In order to compute the Lyapunov exponent using TISEAN we call

```
tiseanPath = 'C:\Programme\MATLAB6p5\work\Tisean\'; system([tiseanPath, 'lyap_r -s20
-o lyap.dat henon.dat']);  l = load('lyap.dat'); plot(l(:,1), l(:,2)) xlabel('Iteration'),
ylabel('log(stretching factor)')
```

Here is the TISEAN software C code below:

Lyap k code :

```
/*Author: Rainer Hegger. Last modified: Sep 3, 1999*/
#include <math.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "routines/tsa.h"

#define WID_STR "Estimates the maximal Lyapunov exponent using the Kantz\n\t\
algorithm"
```

```
#define BOX 128
const unsigned int ibox=BOX-1;

unsigned long length=ULONG_MAX;
unsigned long exclude=0;
unsigned long reference=ULONG_MAX;
unsigned int maxdim=2;
unsigned int mindim=2;
unsigned int delay=1;
unsigned int column=1;
unsigned int epscount=5;
unsigned int maxiter=50;
unsigned int window=0;
unsigned int verbosity=0xff;
double epsmin=1.e-3,epsmax=1.e-2;
char eps0set=0,eps1set=0;
char *outfile=NULL;
char *infile=NULL;

double *series,**lyap;
long box[BOX][BOX],*liste,**lfound,*found,**count;
double max,min;

void show_options(char *progname)
{
  what_i_do(progname,WID_STR);

  fprintf(stderr," Usage: %s [options]\n",progname);
  fprintf(stderr," Options:\n");
  fprintf(stderr,"Everything not being a valid option will be "
            "interpreted as a possible datafile.\nIf no datafile "
            "is given stdin is read. Just - also means stdin\n");
  fprintf(stderr,"\t-l # of data [default: whole file]\n");
  fprintf(stderr,"\t-x # of lines to be ignored [default: 0]\n");
  fprintf(stderr,"\t-c column to read [default: 1]\n");
  fprintf(stderr,"\t-M maxdim [default: 2]\n");
  fprintf(stderr,"\t-m mindim [default: 2]\n");
  fprintf(stderr,"\t-d delay [default: 1]\n");
  fprintf(stderr,"\t-r mineps [default: (data interval)/1000]\n");
  fprintf(stderr,"\t-R maxeps [default: (data interval)/100]\n");
  fprintf(stderr,"\t-# # of eps [default: 5]\n");
  fprintf(stderr,"\t-n # of reference points [default: # of data]\n");
  fprintf(stderr,"\t-s # of iterations [default: 50]\n");
  fprintf(stderr,"\t-t time window [default: 0]\n");
  fprintf(stderr,"\t-o outfile [default: 'datafile'.lyap]\n");
  fprintf(stderr,"\t-V verbosity level [default: 3]\n\t\t"
            "0='only panic messages'\n\t\t"
            "1='+ input/output messages'\n\t\t"
            "2='+ plus statistics'\n");
  fprintf(stderr,"\t-h show these options\n");
  exit(0);
}

void scan_options(int n,char **str)
{
  char *out;
  if ((out=check_option(str,n,'l','u')) != NULL)
    sscanf(out,"%lu",&length);
  if ((out=check_option(str,n,'x','u')) != NULL)
    sscanf(out,"%lu",&exclude);
  if ((out=check_option(str,n,'c','u')) != NULL)
    sscanf(out,"%u",&column);
  if ((out=check_option(str,n,'M','u')) != NULL)
    sscanf(out,"%u",&maxdim);
  if ((out=check_option(str,n,'m','u')) != NULL)
    sscanf(out,"%u",&mindim);
  if ((out=check_option(str,n,'d','u')) != NULL)
    sscanf(out,"%u",&delay);
  if ((out=check_option(str,n,'r','f')) != NULL) {
    eps0set=1;
    sscanf(out,"%lf",&epsmin);
  }
  if ((out=check_option(str,n,'R','f')) != NULL) {
    eps1set=1;
    sscanf(out,"%lf",&epsmax);
  }
  if ((out=check_option(str,n,'#','u')) != NULL)
    sscanf(out,"%u",&epscount);
  if ((out=check_option(str,n,'n','u')) != NULL)
    sscanf(out,"%lu",&reference);
  if ((out=check_option(str,n,'s','u')) != NULL)
    sscanf(out,"%u",&maxiter);
  if ((out=check_option(str,n,'t','u')) != NULL)
    sscanf(out,"%u",&window);
  if ((out=check_option(str,n,'V','u')) != NULL)
    sscanf(out,"%u",&verbosity);
  if ((out=check_option(str,n,'o','o')) != NULL)
    if (strlen(out) > 0)
      outfile=out;
}

void put_in_boxes(double eps)
{
  unsigned long i;
  long j,k;
  static unsigned long blength;

  blength=length-(maxdim-1)*delay-maxiter;

  for (i=0;i<BOX;i++)
    for (j=0;j<BOX;j++)
      box[i][j]= -1;

  for (i=0;i<blength;i++) {
    j=(long)(series[i]/eps)&ibox;
    k=(long)(series[i+delay]/eps)&ibox;
    liste[i]=box[j][k];
    box[j][k]=i;
  }
}

void lfind_neighbors(long act,double eps)
```

```
{
  unsigned int hi,k,k1;
  long i,j,i1,i2,j1,element;
  static long lwindow;
  double dx,eps2=sqr(eps);

  lwindow=(long)window;
  for (hi=0;hi<maxdim-1;hi++)
    found[hi]=0;
  i=(long)(series[act]/eps)&ibox;
  j=(long)(series[act+delay]/eps)&ibox;
  for (i1=i-1;i1<=i+1;i1++) {
    i2=i1&ibox;
    for (j1=j-1;j1<=j+1;j1++) {
      element=box[i2][j1&ibox];
      while (element != -1) {
            if ((element < (act-lwindow)) || (element > (act+lwindow))) {
              dx=sqr(series[act]-series[element]);
              if (dx <= eps2) {
                for (k=1;k<maxdim;k++) {
                  k1=k*delay;
                  dx += sqr(series[act+k1]-series[element+k1]);
                  if (dx <= eps2) {
                        k1=k-1;
                        lfound[k1][found[k1]]=element;
                        found[k1]++;
                  }
                  else
                        break;
                }
              }
            }
            element=liste[element];
      }
    }
  }
}

void iterate_points(long act)
{
  double **lfactor;
  double *dx;
  unsigned int i,j,l,l1;
  long k,element,**lcount;

  check_alloc(lfactor=(double**)malloc(sizeof(double*)*(maxdim-1)));
  check_alloc(lcount=(long**)malloc(sizeof(long*)*(maxdim-1)));
  for (i=0;i<maxdim-1;i++) {
    check_alloc(lfactor[i]=(double*)malloc(sizeof(double)*(maxiter+1)));
    check_alloc(lcount[i]=(long*)malloc(sizeof(long)*(maxiter+1)));
  }
  check_alloc(dx=(double*)malloc(sizeof(double)*(maxiter+1)));

  for (i=0;i<=maxiter;i++)
    for (j=0;j<maxdim-1;j++) {
      lfactor[j][i]=0.0;
      lcount[j][i]=0;
    }

  for (j=mindim-2;j<maxdim-1;j++) {
    for (k=0;k<found[j];k++) {
      element=lfound[j][k];
      for (i=0;i<=maxiter;i++)
	    dx[i]=sqr(series[act+i]-series[element+i]);
      for (l=1;l<j+2;l++) {
	    l1=l*delay;
	    for (i=0;i<=maxiter;i++)
	      dx[i] += sqr(series[act+i+l1]-series[element+l1+i]);
      }
      for (i=0;i<=maxiter;i++)
	    if (dx[i] > 0.0){
	      lcount[j][i]++;
	      lfactor[j][i] += dx[i];
	    }
    }
  }
  for (i=mindim-2;i<maxdim-1;i++)
    for (j=0;j<=maxiter;j++)
      if (lcount[i][j]) {
	    count[i][j]++;
	    lyap[i][j] += log(lfactor[i][j]/lcount[i][j])/2.0;
      }

  for (i=0;i<maxdim-1;i++){
    free(lfactor[i]);
    free(lcount[i]);
  }
  free(lcount);
  free(lfactor);
  free(dx);
}

int main(int argc,char **argv)
{
  char stdi=0;
  double eps_fak;
  double epsilon;
  unsigned int i,j,l;
  FILE *fout;

  if (scan_help(argc,argv))
    show_options(argv[0]);

  scan_options(argc,argv);
#ifndef OMIT_WHAT_I_DO
  if (verbosity&VER_INPUT)
    what_i_do(argv[0],WID_STR);
#endif

  infile=search_datafile(argc,argv,&column,verbosity);
  if (infile == NULL)
    stdi=1;

  if (outfile == NULL) {
```

```
  if (!stdi) {
    check_alloc(outfile=(char*)calloc(strlen(infile)+6,1));
    sprintf(outfile,"%s.lyap",infile);
  }
  else {
    check_alloc(outfile=(char*)calloc(11,1));
    sprintf(outfile,"stdin.lyap");
  }
}
test_outfile(outfile);

series=get_series(infile,&length,exclude,column,verbosity);
rescale_data(series,length,&min,&max);

if (eps0set)
  epsmin /= max;
if (eps1set)
  epsmax /= max;

if (epsmin >= epsmax) {
  epsmax=epsmin;
  epscount=1;
}

if (reference > (length-maxiter-(maxdim-1)*delay))
  reference=length-maxiter-(maxdim-1)*delay;
if ((maxiter+(maxdim-1)*delay) >= length) {
  fprintf(stderr,"Too few points to handle these parameters!\n");
  exit(LYAP_K__MAXITER_TOO_LARGE);
}

if (maxdim < 2)
  maxdim=2;
if (mindim < 2)
  mindim=2;
if (mindim > maxdim)
  maxdim=mindim;

check_alloc(liste=(long*)malloc(sizeof(long)*(length)));
check_alloc(found=(long*)malloc(sizeof(long)*(maxdim-1)));
check_alloc(lfound=(long**)malloc(sizeof(long*)*(maxdim-1)));
for (i=0;i<maxdim-1;i++)
  check_alloc(lfound[i]=(long*)malloc(sizeof(long)*(length)));
check_alloc(count=(long**)malloc(sizeof(long*)*(maxdim-1)));
for (i=0;i<maxdim-1;i++)
  check_alloc(count[i]=(long*)malloc(sizeof(long)*(maxiter+1)));
check_alloc(lyap=(double**)malloc(sizeof(double*)*(maxdim-1)));
for (i=0;i<maxdim-1;i++)
  check_alloc(lyap[i]=(double*)malloc(sizeof(double)*(maxiter+1)));

if (epscount == 1)
  eps_fak=1.0;
else
  eps_fak=pow(epsmax/epsmin,1.0/(double)(epscount-1));

fout=fopen(outfile,"w");
if (verbosity&VER_INPUT)
```

```
    fprintf(stderr,"Opened %s for writing\n",outfile);
  for (l=0;l<epscount;l++) {
    epsilon=epsmin*pow(eps_fak,(double)l);
    for (i=0;i<maxdim-1;i++)
      for (j=0;j<=maxiter;j++) {
            count[i][j]=0;
            lyap[i][j]=0.0;
      }
    put_in_boxes(epsilon);
    for (i=0;i<reference;i++) {
      lfind_neighbors(i,epsilon);
      iterate_points(i);
    }
    if (verbosity&VER_USR1)
      fprintf(stderr,"epsilon= %e\n",epsilon*max);
    for (i=mindim-2;i<maxdim-1;i++) {
      fprintf(fout,"#epsilon= %e  dim= %d\n",epsilon*max,i+2);
      for (j=0;j<=maxiter;j++)
            if (count[i][j])
              fprintf(fout,"%d %e %ld\n",j,lyap[i][j]/count[i][j],count[i][j]);
      fprintf(fout,"\n");
    }
    fflush(fout);
  }
  fclose(fout);
  return 0;}
```

For each embedding dimension and each length scale the file contains a block of data consisting of 3 columns

1. The number of the iteration
2. The logarithm of the stretching factor (the slope is the Lyapunov exponent if it is a straight line)
3. The number of points for which a neighborhood with enough points was found

The other is calculated by Rosenstein algorithm. Here is the code below:

Lyap r code:

```
/*Author: Rainer Hegger, last modified: Sep 4, 1999 */
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
#include <limits.h>
#include <string.h>
```

```
#include "routines/tsa.h"

#define WID_STR "Estimates the maximal Lyapunov exponent; Rosenstein et al."

#define NMAX 256

char *outfile=NULL;
char *infile=NULL;
char epsset=0;
double *series,*lyap;
long box[NMAX][NMAX],*list;
unsigned int dim=2,delay=1,steps=10,mindist=0;
unsigned int column=1;
unsigned int verbosity=0xff;
const unsigned int nmax=NMAX-1;
unsigned long length=ULONG_MAX,exclude=0;
long *found;
double eps0=1.e-3,eps,epsinv;

void show_options(char *progname)
{
  what_i_do(progname,WID_STR);
  fprintf(stderr," Usage: %s [options]\n",progname);
  fprintf(stderr," Options:\n");
  fprintf(stderr,"Everything not being a valid option will be interpreted"
          " as a possible"
          " datafile.\nIf no datafile is given stdin is read. Just - also"
          " means stdin\n");
  fprintf(stderr,"\t-l # of datapoints [default is whole file]\n");
  fprintf(stderr,"\t-x # of lines to be ignored [default is 0]\n");
  fprintf(stderr,"\t-c column to read[default 1]\n");
  fprintf(stderr,"\t-m embedding dimension [default 2]\n");
  fprintf(stderr,"\t-d delay  [default 1]\n");
  fprintf(stderr,"\t-t time window to omit [default 0]\n");
  fprintf(stderr,"\t-r epsilon size to start with [default "
             "(data interval)/1000]\n");
  fprintf(stderr,"\t-s # of iterations [default 10]\n");
  fprintf(stderr,"\t-o name of output file [default 'datafile'.ros]\n");
  fprintf(stderr,"\t-V verbosity level [default 3]\n\t\t"
          "0='only panic messages'\n\t\t"
          "1='+ input/output messages'\n\t\t"
          "2='+ give more detailed information about the length scales\n");
  fprintf(stderr,"\t-h show these options\n");
  fprintf(stderr,"\n");
  exit(0);
}

void scan_options(int n,char **argv)
{
  char *out;

  if ((out=check_option(argv,n,'l','u')) != NULL)
    sscanf(out,"%lu",&length);
  if ((out=check_option(argv,n,'x','u')) != NULL)
    sscanf(out,"%lu",&exclude);
  if ((out=check_option(argv,n,'c','u')) != NULL)
    sscanf(out,"%u",&column);
  if ((out=check_option(argv,n,'m','u')) != NULL)
    sscanf(out,"%u",&dim);
  if ((out=check_option(argv,n,'d','u')) != NULL)
    sscanf(out,"%u",&delay);
  if ((out=check_option(argv,n,'t','u')) != NULL)
    sscanf(out,"%u",&mindist);
  if ((out=check_option(argv,n,'r','f')) != NULL) {
    epsset=1;
    sscanf(out,"%lf",&eps0);
  }
  if ((out=check_option(argv,n,'s','u')) != NULL)
    sscanf(out,"%u",&steps);
  if ((out=check_option(argv,n,'V','u')) != NULL)
    sscanf(out,"%u",&verbosity);
  if ((out=check_option(argv,n,'o','o')) != NULL)
    if (strlen(out) > 0)
      outfile=out;
}

void put_in_boxes(void)
{
  int i,j,x,y,del;

  for (i=0;i<NMAX;i++)
    for (j=0;j<NMAX;j++)
      box[i][j]= -1;

  del=delay*(dim-1);
  for (i=0;i<length-del-steps;i++) {
    x=(int)(series[i]*epsinv)&nmax;
    y=(int)(series[i+del]*epsinv)&nmax;
    list[i]=box[x][y];
    box[x][y]=i;
  }
}

char make_iterate(long act)
{
  char ok=0;
  int x,y,i,j,i1,k,del1=dim*delay;
  long element,minelement= -1;
  double dx,mindx=1.0;

  x=(int)(series[act]*epsinv)&nmax;
  y=(int)(series[act+delay*(dim-1)]*epsinv)&nmax;
  for (i=x-1;i<=x+1;i++) {
    i1=i&nmax;
    for (j=y-1;j<=y+1;j++) {
      element=box[i1][j&nmax];
      while (element != -1) {
            if (labs(act-element) > mindist) {
              dx=0.0;
              for (k=0;k<del1;k+=delay) {
                dx += (series[act+k]-series[element+k])*
                  (series[act+k]-series[element+k]);
                if (dx > eps)
                  break;
```

```
          }
          if (k==del1) {
            if (dx < mindx) {
              ok=1;
              if (dx > 0.0) {
                    mindx=dx;
                    minelement=element;
              }
            }
          }
        }
        element=list[element];
    }
  }
}
if ((minelement != -1) ) {
  act--;
  minelement--;
  for (i=0;i<=steps;i++) {
    act++;
    minelement++;
    dx=0.0;
    for (j=0;j<del1;j+=delay) {
          dx += (series[act+j]-series[minelement+j])*
            (series[act+j]-series[minelement+j]);
    }
    if (dx > 0.0) {
          found[i]++;
          lyap[i] += log(dx);
    }
  }
}
return ok;
}

int main(int argc,char **argv)
{
  char stdi=0,*done,alldone;
  int i;
  long n;
  long maxlength;
  double min,max;
  FILE *file;

  if (scan_help(argc,argv))
    show_options(argv[0]);

  scan_options(argc,argv);
#ifndef OMIT_WHAT_I_DO
  if (verbosity&VER_INPUT)
    what_i_do(argv[0],WID_STR);
#endif

  infile=search_datafile(argc,argv,&column,verbosity);
  if (infile == NULL)
    stdi=1;
if (outfile == NULL) {
  if (!stdi) {
    check_alloc(outfile=(char*)calloc(strlen(infile)+5,(size_t)1));
    strcpy(outfile,infile);
    strcat(outfile,".ros");
  }
  else {
    check_alloc(outfile=(char*)calloc((size_t)10,(size_t)1));
    strcpy(outfile,"stdin.ros");
  }
}
test_outfile(outfile);

series=(double*)get_series(infile,&length,exclude,column,verbosity);
rescale_data(series,length,&min,&max);

if (epsset)
  eps0 /= max;

check_alloc(list=(long*)malloc(length*sizeof(long)));
check_alloc(lyap=(double*)malloc((steps+1)*sizeof(double)));
check_alloc(found=(long*)malloc((steps+1)*sizeof(long)));
check_alloc(done=(char*)malloc(length));

for (i=0;i<=steps;i++) {
  lyap[i]=0.0;
  found[i]=0;
}
for (i=0;i<length;i++)
  done[i]=0;

maxlength=length-delay*(dim-1)-steps-1-mindist;
alldone=0;
file=fopen(outfile,"w");
if (verbosity&VER_INPUT)
  fprintf(stderr,"Opened %s for writing\n",outfile);
for (eps=eps0;!alldone;eps*=1.1) {
  epsinv=1.0/eps;
  put_in_boxes();
  alldone=1;
  for (n=0;n<=maxlength;n++) {
    if (!done[n])
          done[n]=make_iterate(n);
    alldone &= done[n];
  }
  if (verbosity&VER_USR1)
    fprintf(stderr,"epsilon: %e already found: %ld\n",eps*max,found[0]);
}
for (i=0;i<=steps;i++)
  if (found[i])
    fprintf(file,"%d %e\n",i,lyap[i]/found[i]/2.0);
fclose(file);

return 0;
}
```

APPENDIX D: REDUCE AND FORTRAN CODES FOR THE WOLF ALGORITHM

In a classic and highly readable paper [A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, Physica D 16, 285-317 (1985)], Alan Wolf and collaborators described algorithms for calculating the spectrum of Lyapunov exponents from systems in which the equations are known as differential equations. as well as the largest Lyapunov exponent from an experimental time series. Included here is code ported to a standard FORTRAN 77 compiler from Wolf's Fortran code for calculating the spectrum of Lyapunov exponents for maps and flows when the equations are known. The code includes examples for many systems including the Lorenz attractor and the Maxwell Bloch system:

To generate the variational equations (3.24) the following REDUCE Code (Anthony C.. Hearn, REDUCE User’s Manual Version 3.8, Santa Monica, CA (2004)) can be used:

Since the equations of motion for different systems are included, the relevant system must be uncommented by removing the % marks in Column 1.In this case, the Maxwell Bloch equations have been uncommented.

```
off echo;
% Generates the variational equations for the Wolf system

operator aa,v,y;

% Lorenz
%s:=43/2-Sqrt(3)*Sqrt(283)/2;
%r:=50;
%b:=4;
%n:=3;
%v(1):=s*(y(2)-y(1));
%v(2):=y(1)*(r-y(3))-y(2);
%v(3):=y(1)*y(2)-b*y(3);

% Henon Heiles scaled
% n:=4;
% v(1):=y(3);
% v(2):=y(4);
% v(3):=-y(1)-2*eps*y(1)*y(2);
% v(4):=-y(2)-eps*y(1)**2+eps*y(2)**2;

% Sprottd;
%n:=3;
%v(1):=-y(2)+y(2)**2+y(3)**2;
```

```
%v(2):=y(1)+y(3)+y(3)**2;
%v(3):=;


%Rossler;
%a:=0.15;
%b:=0.20;
%cc:=10;
%n:=3;
%v(1):=-(y(2)+y(3));
%v(2):=y(1)+a*y(2);
%v(3):=b+y(3)*(y(1)-cc);

%Rossler Hyper-chaos
%a:=0.25;
%b:=3.0;
%cc:=0.05;
%d:=0.5;
%n:=4;
%v(1):=-(y(2)+y(3));
%v(2):=y(1)+a*y(2)+y(4);
%v(3):=b+y(3)*y(1);
%v(4):=cc*y(4)-d*y(3);


%SprottO
%N:=3;

%v(1):=y(2);
%v(2):=y(1)-y(3);
%v(3):=y(1)+y(1)*y(3)+b*y(2);

%Lasereqs
n:=3;
EE:=y(1);
P:=y(2);
DL:=y(3);
v(1):=-ak*EE+g*P;
v(2):=-gper*P+g*EE*DL;
v(3):=-gpar*(DL-DL0)-4*g*P*EE;



nd:=n**2+n;
for j:=1:n do
for k:=1:n do
aa(j,k):=df(v(j),y(k));

for i:=1:n do
for k:=1:n do
v(i+n*k):=for j1:=1:n sum aa(k,j1)*y(n*j1+i);


%load_package gentran;
%load_package scope;
%GENTRANLANG!* := FORTRAN$
%FORTLINELEN!* :=72;
on fort;
off period;
%off getdecs;
%off gendecs;
out"laser.fort";
write "      subroutine FCN(t,y,v)       ";
write "      implicit real*8(a-h,o-z)";
write "*     subroutine for wolf integration";
write "      dimension y(12),v(12) ";
%write "      parameter(s=16.0d0,r=45.92d0,b=4.0d0)";
write "      parameter(ak=16.0d0,g=4.92d0,gper=4.0d0,gpar=3.0d0,DL0=1.0d0)";
for i:=1:nd do write v(i):=v(i);
write "      RETURN";
write "      END";
SHUT"laser.fort";
;end;
;end;;
```

The Fortran 77 code below is adapted from Wolf. N is the number of equations in the dynamical system (3 for Maxwell Bloch, 4 for Yang-Mills Higgs, NN is the sum of N equations and N^2variational equations for the Wolf algorithm. FCN contains the dynamical system, it can be copied from the REDUCE code output. Must be declared external so that it can be passed to the integrator. In this case the Numerical Recipes Runge Kutta integrator RKQC is used. It is declared external and passed to ODEINT which is the generic integrator in WOLF's paper and references to the IMSL routine there. As coded below, the fiducial trajectory is written to the file traj1.dat and liapunov exponent data is echoed to the terminal and also written to the file liap1.dat. Older versions of both files are erased. The program is commented to describe the meaning of the input parameters Integration tolerance, number of integration steps, time per step and the number of steps where intermediate results are printed. The various parts are also shown by comments. The point at which the REDUCE Wolf output is to be inserted is also shown by comments in FCN. Sample FCN functions for Lorenz and Laser systems are given in commented outg form. The program runs in double precision.

```
      Program liapode
      implicit real*8(a-h,o-z)
*     N = # of nonlnear equations, NN= total # of equations
      PARAMETER (N =3)
      PARAMETER (NN=12)
*     FOR THE NUM-REC INTEGRATOR
```

```
      DIMENSION Y(NN),ZNORM(N),GSC(N),CUM(N),YPRIME(NN),YSCAL(NN)
*
*      INITIAL CONDITIONS FOR NONLINEAR SYSTEM
*
      EXTERNAL FCN,RKQC
      open(18, file='traj1.dat', status='unknown')
      open(19, file='liap1.dat', status='unknown')
      Y(1)=1.0 D0
      Y(2)=1.0  D0
      Y(3)=0.0  D0
*
*      INITIAL CONDITIONS FOR LINEAR SYSTEM (ORTHONORMAL FRAME)
*
      DO 10 I=N+1,NN
10     Y(I)=0
      DO 20 I=1,N
      Y((N+1)*I)=1.0 D0
20     CUM(I)=0
*
*     INTEGRATION TOLERANCE, # OF INTEGRATION STEPS,
*     TIME PER STEP, AND I/O RATE
*
      WRITE(*,*) 'TOL,NSTEP,STPSZE,IO'
      READ(*,*) TOL,NSTEP,STPSZE,IO
      STTOL=0
*
*      Initialization for integration
*
      NEQ=NN
      X=0.0 D0
      IND=1
*
      DO 100 I=1,NSTEP
      XEND=STPSZE*FLOAT(I)
*
*      Call any ODE Integrator - This is an IMSL routine
*      in the original
*      CALL DVERK(NEQ,FCN,X,Y,XEND,TOL,IND,C,NEQ,W,IER)
*      replaced by a Numerical Recipes routine
*      CALL RKQC(Y,YPRIME,NEQ,X,STPSZE,TOL,YSCAL,HDID,HNEXT,FCN)
      CALL ODEINT(Y,NEQ,X,XEND,TOL,STPSZE,STTOL,NOK,NBAD,FCN,RKQC)
      X=XEND
*
*      Construct a new orthonormal basis by Gram-Schmidt method
*
*      Normalize first vector
*
      ZNORM(1)=0.0
      DO 30 J=1,N
```

```
30   ZNORM(1)=ZNORM(1)+Y(N*J+1)**2
     ZNORM(1)=SQRT(ZNORM(1))

     DO 40 J=1,N
40   Y(N*J+1)=Y(N*J+1)/ZNORM(1)
*
*    GENERATE THE NEW ORTHONORMAL SET OF VECTORS
*
     DO 80 J=2,N
*
*    GENERATE J-1 GSR COEFFICIENTS
*
     DO 50 K=1,J-1
     GSC(K)=0.0
     DO 50 L=1,N
     GSC(K)=GSC(K)+Y(N*L+J)*Y(N*L+K)
50   CONTINUE
*
*    CONSTRUCT A NEW VECTOR
*
     DO 60 K=1,N
     DO 60 L=1,J-1
     Y(N*K+J)=Y(N*K+J)-GSC(L)*Y(N*K+L)
60   CONTINUE
*
*    CALCULATE THE VECTOR'S NORM
*
     ZNORM(J)=0.0 D0
     DO 70 K=1,N
     ZNORM(J)=ZNORM(J)+Y(N*K+J)**2
70   CONTINUE
     ZNORM(J)=SQRT(ZNORM(J))
*
*    NORMALIZE THE NEW VECTOR
*
     DO 80 K=1,N
     Y(N*K+J)=Y(N*K+J)/ZNORM(J)
80   CONTINUE
*
*    UPDATE RUNNING VECTOR MAGNITUDES
*
     DO 90 K=1,N
90   CUM(K)=CUM(K)+DLOG(ZNORM(K))/DLOG(2.0D0)
*
*    NORMALIZE EXPONENT AND PRINT EVERY IO ITERATIONS
*
      IF(MOD(I,IO).EQ.0) THEN
       WRITE(*,126) X,(CUM(K)/X,K=1,N)
        WRITE(*,126) X,(CUM(K)/X,K=1,N)
       END IF
126   Format('X= ',f12.7,' LE = ',3(F13.7,1x))
      WRITE(18,138) X, Y(1),Y(2),Y(3)
138   FORMAT(1X,F10.2, 3(1X,F14.4))
100   CONTINUE
      CLOSE (18)
      CLOSE (19)
      STOP
      END

*
*     RHS OF THE LORENZ EQUATIONS
*
      subroutine fcn(t,y,v)
      implicit real*8(a-h,o-z)
*     subroutine for wolf integration
      dimension y(12),v(12)
c   INSERT THE OUTPUT OF THE WOLF REDUCE RUN HERE. THE MAXWELL-
c   BLOCH OUTPUT IS HERE
       parameter(ak=1.75d0,g=1.06d0,gper=2.1d0,gpar=1.0d0,dl0=28.0d0)
      v(1)=y(2)*g-y(1)*ak
      v(2)=y(3)*y(1)*g-y(2)*gper
      v(3)=-y(3)*gpar-4*y(2)*y(1)*g+dl0*gpar
      v(4)=y(7)*g-y(4)*ak
      v(5)=y(8)*g-y(5)*ak
      v(6)=y(9)*g-y(6)*ak
      v(7)=y(10)*y(1)*g-y(7)*gper+y(4)*y(3)*g
      v(8)=y(11)*y(1)*g-y(8)*gper+y(5)*y(3)*g
      v(9)=y(12)*y(1)*g-y(9)*gper+y(6)*y(3)*g
      v(10)=-y(10)*gpar-4*y(7)*y(1)*g-4*y(4)*y(2)*g
      v(11)=-y(11)*gpar-4*y(8)*y(1)*g-4*y(5)*y(2)*g
      v(12)=-y(12)*gpar-4*y(9)*y(1)*g-4*y(6)*y(2)*g
c    REDUCE OUTPUT ENDS HERE
      return
      end


      SUBROUTINEODEINT(YSTART,NVAR,X1,X2,EPS,H1,HMIN,NOK,
    & NBAD,DERIVS,RKQC)
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (MAXSTP=10000,NMAX=12,TWO=2.0D0,ZERO=0.0D0,TINY=1.D-
30)
      COMMON /PATH/ KMAX,KOUNT,DXSAV,XP(200),YP(10,200)
      DIMENSION YSTART(NVAR),YSCAL(NMAX),Y(NMAX),DYDX(NMAX)
      X=X1
      H=SIGN(H1,X2-X1)
      NOK=0
      NBAD=0
      KOUNT=0
```

```
     DO 11 I=1,NVAR
       Y(I)=YSTART(I)
11     CONTINUE
     XSAV=X-DXSAV*TWO
     DO 16 NSTP=1,MAXSTP
       CALL DERIVS(X,Y,DYDX)
       DO 12 I=1,NVAR
         YSCAL(I)=ABS(Y(I))+ABS(H*DYDX(I))+TINY
12       CONTINUE
       IF(KMAX.GT.0)THEN
         IF(ABS(X-XSAV).GT.ABS(DXSAV)) THEN
           IF(KOUNT.LT.KMAX-1)THEN
             KOUNT=KOUNT+1
             XP(KOUNT)=X
             DO 13 I=1,NVAR
               YP(I,KOUNT)=Y(I)
13             CONTINUE
             XSAV=X
           ENDIF
         ENDIF
       ENDIF
       IF((X+H-X2)*(X+H-X1).GT.ZERO) H=X2-X
       CALL RKQC(Y,DYDX,NVAR,X,H,EPS,YSCAL,HDID,HNEXT,DERIVS)
       IF(HDID.EQ.H)THEN
         NOK=NOK+1
       ELSE
         NBAD=NBAD+1
       ENDIF
       IF((X-X2)*(X2-X1).GE.ZERO)THEN
         DO 14 I=1,NVAR
           YSTART(I)=Y(I)
14         CONTINUE
         IF(KMAX.NE.0)THEN
           KOUNT=KOUNT+1
           XP(KOUNT)=X
           DO 15 I=1,NVAR
             YP(I,KOUNT)=Y(I)
15           CONTINUE
         ENDIF
         RETURN
       ENDIF
       IF(ABS(HNEXT).LT.HMIN) PAUSE 'Stepsize smaller than minimum.'
       H=HNEXT
16     CONTINUE
     PAUSE 'Too many steps.'
     RETURN
     END

     SUBROUTINE RK4(Y,DYDX,N,X,H,YOUT,DERIVS)
```

```
      implicit real*8(a-h,o-z)
      PARAMETER (NMAX=12)
      DIMENSION Y(N),DYDX(N),YOUT(N),YT(NMAX),DYT(NMAX),DYM(NMAX)
      HH=H*0.5
      H6=H/6.
      XH=X+HH
      DO 11 I=1,N
        YT(I)=Y(I)+HH*DYDX(I)
11    CONTINUE
      CALL DERIVS(XH,YT,DYT)
      DO 12 I=1,N
        YT(I)=Y(I)+HH*DYT(I)
12    CONTINUE
      CALL DERIVS(XH,YT,DYM)
      DO 13 I=1,N
        YT(I)=Y(I)+H*DYM(I)
        DYM(I)=DYT(I)+DYM(I)
13    CONTINUE
      CALL DERIVS(X+H,YT,DYT)
      DO 14 I=1,N
        YOUT(I)=Y(I)+H6*(DYDX(I)+DYT(I)+2.*DYM(I))
14    CONTINUE
      RETURN
      END

      SUBROUTINE RKQC(Y,DYDX,N,X,HTRY,EPS,YSCAL,HDID,HNEXT,DERIVS)
      implicit real*8(a-h,o-z)
      PARAMETER (NMAX=12,FCOR=.0666666667d0,
     *    ONE=1.d0,SAFETY=0.9d0,ERRCON=6.d-4)
      EXTERNAL DERIVS
      DIMENSION
Y(N),DYDX(N),YSCAL(N),YTEMP(NMAX),YSAV(NMAX),DYSAV(NMAX)
      PGROW=-0.20d0
      PSHRNK=-0.25d0
      XSAV=X
      DO 11 I=1,N
        YSAV(I)=Y(I)
        DYSAV(I)=DYDX(I)
11    CONTINUE
      H=HTRY
1     HH=0.5*H
      CALL RK4(YSAV,DYSAV,N,XSAV,HH,YTEMP,DERIVS)
      X=XSAV+HH
      CALL DERIVS(X,YTEMP,DYDX)
      CALL RK4(YTEMP,DYDX,N,X,HH,Y,DERIVS)
      X=XSAV+H
      IF(X.EQ.XSAV)PAUSE 'Stepsize not significant in RKQC.'
      CALL RK4(YSAV,DYSAV,N,XSAV,H,YTEMP,DERIVS)
      ERRMAX=0.
      DO 12 I=1,N
        YTEMP(I)=Y(I)-YTEMP(I)
        ERRMAX=MAX(ERRMAX,ABS(YTEMP(I)/YSCAL(I)))
12      CONTINUE
      ERRMAX=ERRMAX/EPS
      IF(ERRMAX.GT.ONE) THEN
        H=SAFETY*H*(ERRMAX**PSHRNK)
        GOTO 1
      ELSE
        HDID=H
        IF(ERRMAX.GT.ERRCON)THEN
          HNEXT=SAFETY*H*(ERRMAX**PGROW)
        ELSE
          HNEXT=4.d0*H
        ENDIF
      ENDIF
      DO 13 I=1,N
        Y(I)=Y(I)+YTEMP(I)*FCOR
13      CONTINUE
      RETURN
      END

C     HERE ARE THE LORENZ AND LASER WOLF OUTPUTS THEY ARE
C     COMMENTED OUT.

*
*    RHS OF THE LORENZ EQUATIONS
*
*     subroutine fcn(t,y,v)
*     implicit real*8(a-h,o-z)
*    subroutine for wolf integration
*     dimension y(12),v(12)
*         COMMON/ENBYKLP/S,R,B
*         v(1)=s*(y(2)-y(1))
*     v(2)=-y(3)*y(1)-y(2)+y(1)*r
*     v(3)=-y(3)*b+y(2)*y(1)
*     v(4)=s*(y(7)-y(4))
*     v(5)=s*(y(8)-y(5))
*     v(6)=s*(y(9)-y(6))
*     v(7)=-y(10)*y(1)-y(7)-y(4)*y(3)+y(4)*r
*     v(8)=-y(11)*y(1)-y(8)-y(5)*y(3)+y(5)*r
*     v(9)=-y(12)*y(1)-y(9)-y(6)*y(3)+y(6)*r
*     v(10)=-y(10)*b+y(7)*y(1)+y(4)*y(2)
*     v(11)=-y(11)*b+y(8)*y(1)+y(5)*y(2)
*     v(12)=-y(12)*b+y(9)*y(1)+y(6)*y(2)
*     return
*     end
```

```
*
*    RHS OF THE LASER EQUATIONS
*
      subroutine fcn(t,y,v)
            implicit real*8(a-h,o-z)
*     subroutine for wolf integration
      dimension y(12),v(12)
      COMMON/ENBYKLP/AK,G,GPER,GPAR,DL0,IFLAG
*     parameter(ak=1.0d-1,g=64.92d0,gper=44.0d0,gpar=43.0d0,dl0=100.0d0)
      v(1)=y(2)*g-y(1)*ak
      v(2)=y(3)*y(1)*g-y(2)*gper
      v(3)=-y(3)*gpar-4*y(2)*y(1)*g+dl0*gpar
      v(4)=y(7)*g-y(4)*ak
      v(5)=y(8)*g-y(5)*ak
      v(6)=y(9)*g-y(6)*ak
      v(7)=y(10)*y(1)*g-y(7)*gper+y(4)*y(3)*g
      v(8)=y(11)*y(1)*g-y(8)*gper+y(5)*y(3)*g
      v(9)=y(12)*y(1)*g-y(9)*gper+y(6)*y(3)*g
      v(10)=-y(10)*gpar-4*y(7)*y(1)*g-4*y(4)*y(2)*g
      v(11)=-y(11)*gpar-4*y(8)*y(1)*g-4*y(5)*y(2)*g
      v(12)=-y(12)*gpar-4*y(9)*y(1)*g-4*y(6)*y(2)*g
      return
      end
```