5-2015

Determination of Chaos in Different Dynamical Systems

Sherli Koshy-Chenthittayil
Clemson University

Determination of Chaos in Different Dynamical Systems

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mathematics

by
Sherli Koshy-Chenthittayil
May 2015

Accepted by:
Dr. Elena Dimitrova, Committee Chair
Dr. Eleanor Jenkins
Dr. Brian Dean

Abstract

It has been widely observed that most deterministic dynamical systems go into chaos for some values of their parameters. There are many ways to measure chaos. One popular way uses Lyapunov exponents. The objective of this thesis is to find the parameter values for a system that determines chaos via the Lyapunov exponents.The paper by Wolf et.al.,[2] proposed the frequently used choice of calculating such exponents using Gram-Schmidt orthonormalization process. The work in this thesis centered on coding and verifying the algorithm in [2], as well as using the code to investigate three biological models [7],[6] and [1] to find parameters/initial conditions to give chaos. Finally it also considers as future work choosing appropriate sampling algorithms to better understand the parameter space for which we may obtain chaos.

Acknowledgments

I would like to first acknowledge God Almighty for his grace and kindness. I would also like to thank my family for their support. Next I am really grateful to Dr.Elena Dimitrova for helping me understand chaos theory, Dr.Lea Jenkins for all the help with the Lyapunov calculator code and Dr.Brian Dean for help with the sampling algorithms and subsequent coding. A special thank you to Dr.Oleg Yordanov for all his help in rescaling the systems. Thank you all so much.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

It is an indisputable fact that chaos exists not just in theory. The objective of this thesis is to find the parameter values for a system that determines chaos via Lyapunov exponents. Before we delve into chaos, let us go through the background needed for it.

1.1 Background information

- **Dynamical systems** A dynamical system consists of a set of possible states, together with a rule determining the present state based on the previous state [3]. For example consider a simple dynamical system given by $x_{n+1} = 2x_n$. Here the variable n stands for time and x_n denotes the population at time n.

- **Deterministic Dynamical Systems** A deterministic dynamical system is one in which the present state is determined **uniquely** from the past states. In our previous example, the present population is completely determined by the previous one.

  If randomness occurs in the prediction of the new state, then the system is no longer deterministic but a *random* or *stochastic* process. An example of such a process is flipping a fair coin to determine if it will rain or not. A coin has no predictive power over rain.

Types of Dynamical Systems

- *Discrete-time Dynamical Systems*: If the rule is applied at discrete times, the system is called a discrete-time dynamical system. Our example is a discrete system.

- *Continuous-time Dynamical Systems*: It is essentially the limit of discrete system with smaller and smaller updating times. In this case, the governing rule will become a set of differential equations. Instead of expressing the current state as a function of the previous state, the differential equation expresses the **rate of change** of the current state as a function of the previous state [3]. We will be considering continuous dynamical systems with ordinary differential equations.

As we all know, an ordinary differential equation is one in which the solutions are functions of an independent variable. In our case the independent variable will be time denoted by t. Such equations come in two types:

- An *autonomous differential equation* is one in which t does not appear explicitly. An example for this would be the equation of pendulum given by:

$$\frac{dx}{dt} = -\sin x.$$

- A *nonautonomous differential equation* is one where t explicitly appears. The equation of the forced damped pendulum:

$$(1+c)\frac{dx}{dt} = -\sin x + \rho \sin t$$

is an example for such an equation.

Any nonautonomous system can be transformed into an autonomous system by introducing a new variable y and setting it to be equal to t. This conversion requires an additional differential equation. For the above example the autonomous version would be:

$$\begin{aligned}(1+c)\frac{dx}{dt} &= -\sin x + \rho \sin y \\ \frac{dy}{dt} &= 1\end{aligned}$$

1.2 Behavior of the dynamical systems

We shall describe the behavior of the dynamical systems in terms of equilibrium solutions, limit cycles and chaos.

- **Equilibrium solutions:** A constant solution of the autonomous differential equation $\frac{dx}{dt} = f(x)$ is called an *equilibrium* of the equation[3]. In other words, it is a solution which satisfies $f(x) = 0$. The solutions either converge to the equilibrium or diverge away from it.

- **Periodic orbits:** If there exists a $T > 0$ such that $F(t+T, v_0) = F(t, v_0), \forall t$ and if v_0 is not an equilibrium, then the solution $F(t, v_0)$ is called a *periodic orbit* or *cycle.* Here $F(t, v_0)$ denotes the value of the solution at time t with initial value v_0. Also the periodic orbit traces out a simple closed curve.

- **Chaotic orbit:** An orbit that exhibits an unstable behavior that is not itself fixed or periodic is called a *chaotic orbit.* At any point in such an orbit, there are points arbitrarily near that will move away from the point during further iteration. In terms of solutions, it means they are very sensitive to small perturbations in the initial conditions and almost all of them do not appear to be either periodic or converge to equilibrium solutions.

For autonomous differential equations on the real line, bounded solutions must converge to an equilibrium. For planar autonomous systems, solutions that are bounded may instead converge to periodic orbits or cycles. In this case solutions cannot be chaotic. There is no such restriction in three-dimensional cases. These results follow from the Poincaré-Bendixson Theorem. A classic three-dimensional system which displays stable equilibria and chaotic behavior for different values of a parameter is the Lorenz model given below:

$$\begin{aligned}
\dot{x} &= \sigma(y - x) \\
\dot{y} &= x(\rho - z) - y \\
\dot{z} &= xy - \beta z.
\end{aligned}$$

For $\sigma = 10, \beta = 8/3$, Lorenz found that the system behaved chaotically for $\rho \geq 24.74$. The chaotic attractor is shown below:

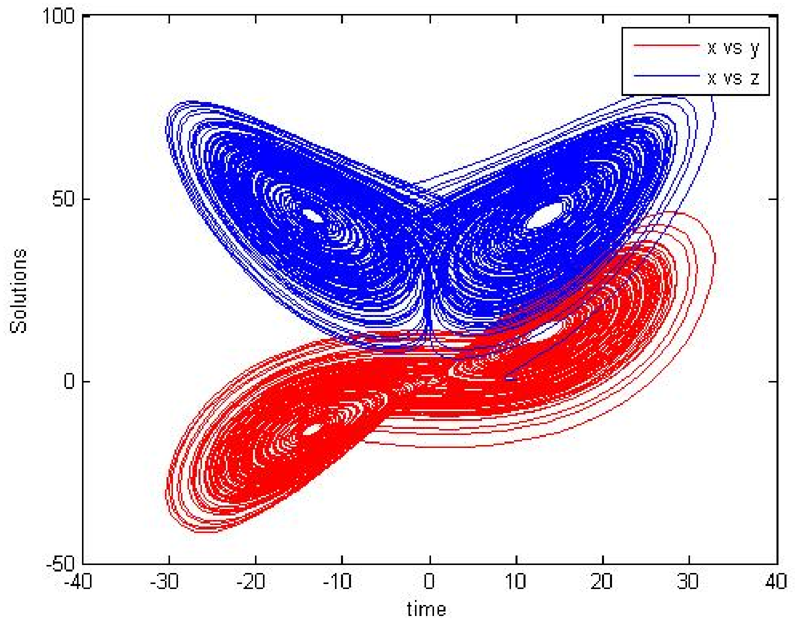


Figure 1.1: Lorenz attractor

This figure depicts the orbit of a single set of initial conditions. This is a numerically observed attractor since the choice of almost any initial condition in a neighborhood of the chosen set results in a similar figure [3].
A chaotic attractor can be dissipative (volume-decreasing), locally unstable (orbits do not settle down to stationary, periodic, or quasiperiodic motion) or stable at large scale(i.e. they get trapped in a strange attractor).

In the next chapter, we discuss ways to measure chaos using the Lyapunov spectrum. In the third chapter, we go on to evaluate the spectrum for three continuous-time dynamical systems based on biological populations. The final chapter considers as future work choosing appropriate sampling algorithms to better understand the parameter space for which we may obtain chaos.

Chapter 2

Evaluation of Lyapunov Spectrum

2.1 Definitions

A usual measure of chaos is finding the Lyapunov spectrum of the system. If at least one of the Lyapunov exponents is positive then the bounded aperiodic orbit is said to be chaotic [3]. As the systems investigated in this thesis are continuous, the definition of the exponent will be given in terms of such a dynamical system.[2] Consider a continuous dynamical system in an n-dimensional phase space. We are observing the long term behavior of an *infinitesimal n-sphere* (i.e. sphere of very small radius) of initial conditions. Due to the locally deforming nature of the flow, the sphere eventually becomes an *n-ellipsoid.* The Lyapunov exponent is calculated for each dimension and it is dependent on the length of the principal axis of the ellipsoid. It is given by:

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} log_2 \frac{p_i(t)}{p_i(0)} \tag{2.1}$$

where $p_i(t)$ denotes the length of the ellipsoidal principal axis at time t and $p_i(0)$

denotes its length at time $t = 0$.
The exponents are generally given in decreasing order, i.e. $\lambda_1 > \lambda_2 > \cdots > \lambda_n$.

The exponents give us an idea of whether a specific direction in the phase space is contracting or expanding. An expanding direction signifies a positive exponent and contracting a negative one. As the orientation of the ellipsoid is varying continuously, we cannot speak of a definite direction with respect to the exponent. For a dissipative dynamical system, we will have at least one negative Lyapunov exponent. If the exponent is positive, we wouldn't expect a bounded attractor unless some folding of widely separated trajectories takes place. So for that particular direction, the system goes through a repeated stretching and folding processes. As a result of this, we cannot predict the long-term behavior of the system given the initial conditions which is the very definition of chaos.

For a one-dimensional system, the Lyapunov spectrum clearly consists of one value. For a discrete dynamical system, it is positive for a chaotic regime, zero for a marginally stable orbit and negative for a periodic orbit [2]. For a continuous one-dimensional dynamical system, the Lyapunov exponent will always be negative. For a continuous three-dimensional system which is dissipative (i.e volume decreasing), the possible spectra are as follows:
$(+, 0, -)$ denotes a strange attractor, $(0, 0, -)$ denotes a two-torus, $(0, -, -)$ for a limit cycle and finally $(-, -, -)$ for a fixed point.
This can be extended to n-dimensions. The magnitude of the Lyapunov exponent computes the attractor's dynamics;i.e it tells us the number of orbits after which we cannot predict the future behavior of the initial condition [2].

2.2 Procedure for calculation of Lyapunov Exponents

The definition of Lyapunov exponents requires us to define principal axes with initial conditions. These axes need to evolve with the equations of the system. The issue is we cannot guarantee the condition of small separations for times on the order of hundreds of orbital periods needed for convergence in a chaotic system. To overcome this, the authors of [2] use a phase space together with a tangent space approach. A *fiducial trajectory* (center of the sphere) is obtained by the action of the non-linear system on some initial conditions. Now to obtain the trajectories of points on the surface of the sphere, we consider the action of the *linearized* system on points very close to the fiducial trajectory. In fact, the principal axes are defined by the evolution via the linearized equations of an initially orthonormal vector frame anchored to the fiducial trajectory. [2].

To define the trajectories on the points of the sphere we need the concept of a *linearized system* or *variational equations.* Consider a dynamical system of the form $\vec{x}' = \vec{F}(\vec{x})$,where $\vec{x} = (x_1, x_2, \ldots, x_n)$, $\vec{F} = (f_1, f_2, \ldots, f_n)$. It's easy to generate the state-space trajectory $\phi(\vec{x}_0)$ by using any numerical ODE solver. But what happens if there are small perturbations in $\vec{x}$? The formal way to describe how these perturbations react is with the use of partial derivatives. For instance, $\dfrac{\partial f_1}{\partial x_2}$ is how much the slope of the first variable (f_1) changes if you perturb the second variable x_2. [5]

Consider the Lorenz system:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(\rho - z) - y$$
$$\dot{z} = xy - \beta z$$

To set up the linearized system corresponding to the above equations we would need the Jacobian of the right-hand side which is given by:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix}$$

where f_i is the right-hand side of the i^{th} differential equation. For a n-dimensional system we would have an $n \times n$ matrix.
For the Lorenz system the Jacobian is

$$\begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - z & -1 & -x \\ y & x & -\beta \end{bmatrix}$$

To set up the variational equations we would need to describe the variations. For this consider the following matrix:

$$[\delta] = \begin{bmatrix} \delta_{x1} & \delta_{y1} & \delta_{z1} \\ \delta_{x2} & \delta_{y2} & \delta_{z2} \\ \delta_{x3} & \delta_{y3} & \delta_{z3} \end{bmatrix}$$

where δ_{xi} is the component of the x variation that came from the i^{th} equation.

The column sums of this matrix are the lengths of the x,y, and z coordinates of the evolved variation. The rows are the coordinates of the vectors into which the original x,y, and z components of the variation have evolved.
The linearized equations are:

$$\begin{bmatrix} \dot{\delta}_{x1} & \dot{\delta}_{y1} & \dot{\delta}_{z1} \\ \dot{\delta}_{x2} & \dot{\delta}_{y2} & \dot{\delta}_{z2} \\ \dot{\delta}_{x3} & \dot{\delta}_{y3} & \dot{\delta}_{z3} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} \begin{bmatrix} \delta_{x1} & \delta_{y1} & \delta_{z1} \\ \delta_{x2} & \delta_{y2} & \delta_{z2} \\ \delta_{x3} & \delta_{y3} & \delta_{z3} \end{bmatrix}$$

For the Lorenz system, it would be:

$$\begin{bmatrix} \dot{\delta}_{x1} & \dot{\delta}_{y1} & \dot{\delta}_{z1} \\ \dot{\delta}_{x2} & \dot{\delta}_{y2} & \dot{\delta}_{z2} \\ \dot{\delta}_{x3} & \dot{\delta}_{y3} & \dot{\delta}_{z3} \end{bmatrix} = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - z & -1 & -x \\ y & x & -\beta \end{bmatrix} \begin{bmatrix} \delta_{x1} & \delta_{y1} & \delta_{z1} \\ \delta_{x2} & \delta_{y2} & \delta_{z2} \\ \delta_{x3} & \delta_{y3} & \delta_{z3} \end{bmatrix}$$

So now in addition to the original system of n non-linear equations we will have an additional n^2 linearized equations. The system now has $n + n^2 = n(n+1)$ equations.

To implement the procedure mentioned initially for creating the fiducial trajectory we solve the new system of $n(n+1)$ differential equations with any numerical ode algorithm, e.g., Runge-Kutta 4 for some initial conditions and a time range $[tstart, tstart + ts]$ where $tstart$ denotes the initial time and ts denotes the time step.

In a chaotic system, each vector tends to fall along the local direction of most rapid growth. In addition, the finite precision arithmetic of computing, the collapse towards a common direction causes the tangent space orientation of all axis vectors to become indistinguishable. To overcome this, Wolf et.al.,[2] use repeated Gram-Schmidt re-orthonormalization(GSR) procedure on the vector frame.

Let the linearized equations act on the initial frame of orthonormal vectors to give a set of vectors $\{v_1, v_2, \ldots, v_n\}$. In other words, after we solve the system of $n(n+1)$ equations, consider the components corresponding to the variational equations. Then GSR provides the following orthonormal set $\{v'_1, v'_2, \ldots, v'_n\}$:

$$\begin{aligned}
v'_1 &= \frac{v_1}{\|v_1\|}, \\
v'_2 &= \frac{v_2 - \langle v_2, v'_1\rangle v'_1}{\|v_2 - \langle v_2, v'_1\rangle v'_1\|} \\
&\vdots \\
v'_n &= \frac{v_n - \langle v_n, v'_{n-1}\rangle v'_{n-1} - \cdots - \langle v_n, v'_1\rangle v'_1}{\|v_n - \langle v_n, v'_{n-1}\rangle v'_{n-1} - \cdots - \langle v_n, v'_1\rangle v'_1\|}
\end{aligned}$$

where $\langle , \rangle$ denotes the Euclidean inner-product. The orthonormal set thus obtained now serves as the new initial conditions for our linearized system. We then solve the system again now with these new initial conditions and a new time-range $[tstart, tstart + ts]$ where $tstart$ has now been changed to $tstart + ts$ and ts denotes the time step. This procedure is repeated n times.

It is seen that GSR never affects the direction of the first vector in a system, so this vector tends to seek out the most rapidly growing direction in the tangent space[2]. The length of vector v_1 is proportional to $2^{\lambda_1 t}$ so in this way we can obtain the first

Lyapunov exponent λ_1. According to the construction of v_2', we are changing the direction of v_2. Because v_2's direction is being changed, it is not free to chase after the most rapidly growing direction nor the second most. Also note that the vectors v_1' and v_2' span the same subspace as v_1 and v_2. The area defined by the vectors v_1 and v_2 is proportional to $2^{(\lambda_1+\lambda_2)t}$. As v_1' and v_2' are orthogonal, we may determine λ_2 directly from the mean rate of growth of the projection of vector v_2 on vector v_2' [2].

Extending this line of thought to n-dimensions, we conclude that the subspace spanned by the n vectors is not affected by the GSR process. The long-term evolution of the $n-$volume defined by these vectors is proportional to $2^{\sum_{i=1}^{n} \lambda_i t}$. Projection of the evolved vectors onto the new orthonormal frame correctly updates the rates of growth of each of the principal axes in turn, providing estimates of the Lyapunov exponents.

The code from the Wolf paper [2] was verified on standard systems, like Lorenz and Rossler.

System	Equations	Parameters	Initial Conditions	Lyapunov Spectrum (in [2])	Lyapunov Spectrum obtained
Lorenz	$\dot{x} = \sigma(y - x)$ $\dot{y} = x(\rho - z) - y$ $\dot{z} = xy - \beta z$	$\sigma = 10.0,$ $\rho = 45.92,$ $\beta = 4.0$	$x = 10.0,$ $y = 1,$ $z = 0$	$\lambda_1 = 2.16,$ $\lambda_2 = 0.00,$ $\lambda_1 = -32.4$	$\lambda_1 = 2.1676,$ $\lambda_2 = 0.0001,$ $\lambda_3 = -32.4644$
Rossler	$\dot{x} = -(y + z)$ $\dot{y} = x + ay$ $\dot{z} = b + z(x - c)$	$a = 0.15,$ $b = 0.20,$ $c = 10.0$	$x = 10.0,$ $y = 1,$ $z = 0$	$\lambda_1 = 0.13,$ $\lambda_2 = 0.00,$ $\lambda_3 = -14.1$	$\lambda_1 = 0.1309,$ $\lambda_2 = 0.0013,$ $\lambda_3 = -14.1669$

Table 2.1: Lyapunov Spectrum in [2] vs Lyapunov Spectrum obtained through the MATLAB code

Chapter 3

Systems under consideration

In this chapter we shall consider different dynamical systems and study some parameters in each system which may give chaos. The systems considered are all motivated by biological experiments.

3.1 Kot System

3.1.1 The Unforced System

The first system we consider was analyzed by Kot,Sayler and Schulz [7]. It is a double-monod system with a prey (protozoan) and a predator (bacteria). The system initially analyzed in the work did not consider a forced inflowing nutrient and did not exhibit chaotic behavior. The unforced nutrient system [7] is given by:

$$\frac{dS}{dt} = D\left[S_i - S\right] - \frac{\mu_1}{Y_1}\frac{SH}{K_1 + S} \tag{3.1}$$
$$\frac{dH}{dt} = \mu_1\frac{SH}{K_1 + S} - DH - \frac{\mu_2}{Y_2}\frac{HP}{K_2 + H} \tag{3.2}$$
$$\frac{dP}{dt} = \mu_2\frac{HP}{K_2 + H} - DP \tag{3.3}$$

where

1. S represents the concentration of limiting substrate.
2. H represent the concentration of the prey.
3. P represents the predator concentration.
4. D is the dilution rate.
5. μ_1 and μ_2 represent the maximum specific growth rate of the prey and predator respectively.
6. Y_1 is the yield of the prey per unit mass of substrate. Similarly Y_2 is the biomass yield of the predator per unit mass of prey

For ease of calculations, the authors of [7] re-scaled the concentrations by the inflowing substrate concentrations, the prey by its yield constant and the predator by its yield constants i.e.

$$x = \frac{S}{S_i}, y = \frac{H}{Y_1 S_i}, z = \frac{P}{Y_1 Y_2 S_i}, \tau = D * t$$

The resulting re-scaled equations are as follows:

$$\frac{dx}{d\tau} = 1 - x - \frac{Axy}{a+x} \tag{3.4}$$
$$\frac{dy}{d\tau} = \frac{Axy}{a+x} - y - \frac{Byz}{b+y} \tag{3.5}$$
$$\frac{dz}{d\tau} = \frac{Byz}{b+y} - z \tag{3.6}$$

Here $A = \dfrac{\mu_1}{D}$, $a = \dfrac{K_1}{S_i}$, $B = \dfrac{\mu_2}{D}$, and $b = \dfrac{K_2}{Y_1 S_i}$.

The following parameters were used for the calculations:

$D = 0.1, S_i = 115$

	Y_i	μ_i h^{-1}	K_i mg/l
Prey	0.4	0.5	8
Predator	0.6	0.2	9

Table 3.1: Values of parameters for microbial model presented in Kot, et.al. [7]

As mentioned before, on analysis of the model, the authors Kot et.al.,[7] observed equilibrium points and no chaos. Upon calculation of Lyapunov spectrum we obtain (0,-,-) which agrees with the nature of the model. The zero exponent is due to the system being autonomous.

3.1.2 The Forced System

In this case they consider when the nutrient is being forced into the system out of phase with internal substrate. The equations used to model that system were as follows [7]

$$\begin{aligned}
\frac{dS}{dt} &= D\left[S_i\left(1+\epsilon\sin\left(\frac{2\pi}{T}t\right)\right)-S\right]-\frac{\mu_1}{Y_1}\frac{SH}{K_1+S}\\
\frac{dH}{dt} &= \mu_1\frac{SH}{K_1+S}-DH-\frac{\mu_2}{Y_2}\frac{HP}{K_2+H}\\
\frac{dP}{dt} &= \mu_2\frac{HP}{K_2+H}-DP
\end{aligned}$$

where the parameters and variables are as in the unforced model (Eqn. 3.1).

For ease of calculations, the authors of [7] again re-scaled the concentrations as they did in the unforced model. The resulting re-scaled equations are as follows:

$$\frac{dx}{d\tau} = 1+\epsilon\sin(\omega\tau)-x-\frac{Axy}{a+x} \tag{3.7}$$

$$\frac{dy}{d\tau} = \frac{Axy}{a+x}-y-\frac{Byz}{b+y} \tag{3.8}$$

$$\frac{dz}{d\tau} = \frac{Byz}{b+y}-z \tag{3.9}$$

where $\omega = \dfrac{2\pi}{DT}$.

The parameters A, a, B, b were calculated as before. The values in Table 3.1 were applied to this model as well.

3.1.3 Results of Simulation

In [7], the authors vary the value of ω to observe the behavior of the model. The choice of this parameter was because it drives the periodic forcing of the inflowing substrate, which in turn causes chaos for certain values of ω. For $\omega = \frac{5\pi}{6}$ and $\epsilon = 0.6$ they observed chaotic behavior which was simulated below:

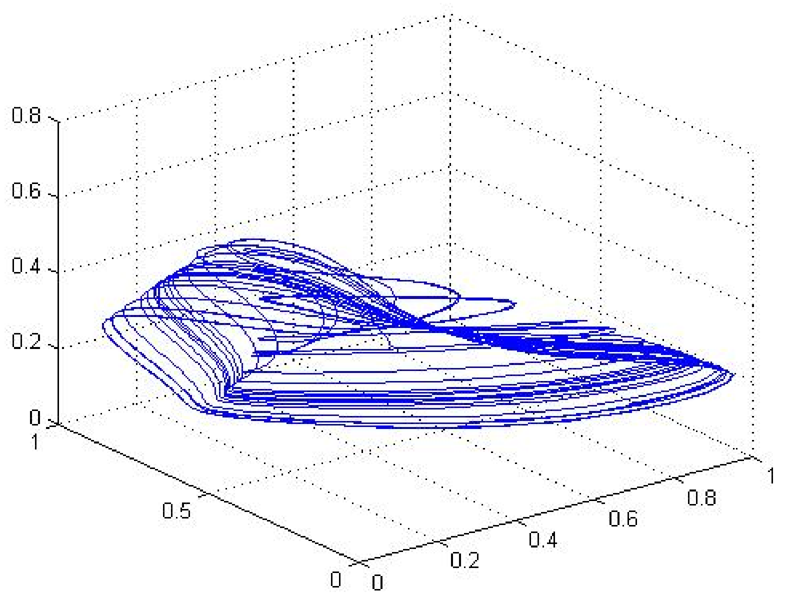


Figure 3.1: Manifold plot of forced model in [7] when $\omega = \frac{5\pi}{6}$ and $\epsilon = 0.6$

Accordingly, varying ω over a range of $[0, 6\pi]$ indicated that the system goes in and out of chaos. As ϵ was the coefficient of the sinusoidal term in Eqn.(3.9), it was also varied in a range of $[0, 1]$ to observe the dynamics.

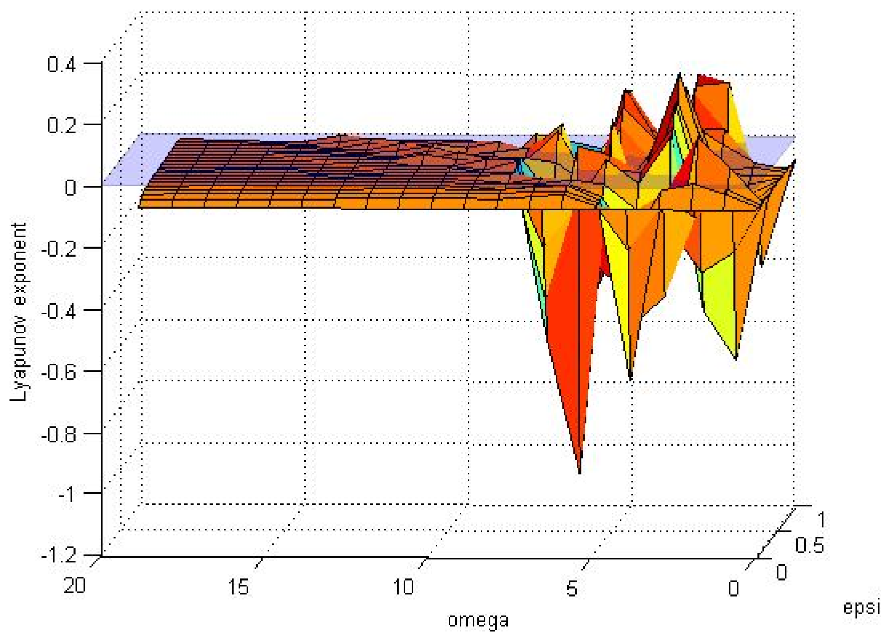


Figure 3.2: 3-D plot depicting chaos and non-chaos with changes in ϵ and ω

In the above figure, the maximum Lyapunov exponent was plotted against ω and ϵ. A positive maximum exponent depicts chaos, a negative maximum exponent depicts a fixed point and if the maximum exponent is zero it could mean either a two-torus or a limit cycle.

Another set of parameters which seemed worthwhile to investigate were the dilution rate D and the inflowing substrate concentration S_i. These two parameters can be controlled by the chemostat's operator. Varying them would give us an idea of whether the system exhibits chaos or not.

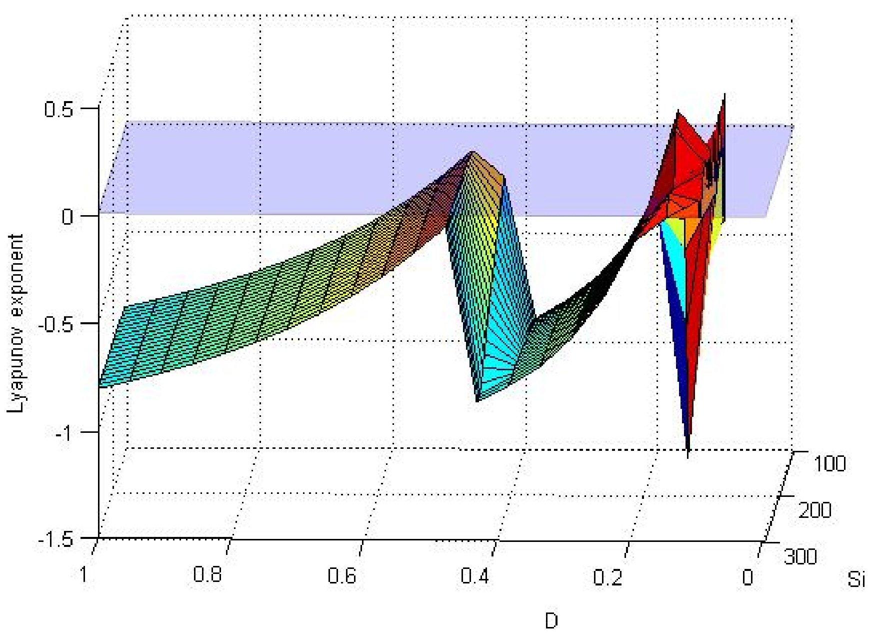


Figure 3.3: 3-D plot depicting chaos and non-chaos with changes in D and S_i

As can be seen in both the figures, the parameters ϵ,ω,D,S_i are interesting parameters which can be investigated further to understand the model's behavior.

3.2 Kravchenko System

Nikolay S.Strigul and Lev V.Kravchenko in their paper [6] consider a model based on Monod kinetics. The variables under consideration are the concentration of PGPR (aerobic non-nitrogen fixing bacteria), resident micro-organisms, oxygen and soluble substrate.

The mathematical model consists of four non-linear ordinary differential equations which are as follows:

$$\begin{aligned}
\frac{dX}{dT} &= X\left(\mu_X\left[S,P,N\right] + F\left[Z\right] - \alpha X - D_1\right)\\
\frac{dZ}{dT} &= Z\left(\mu_Z\left[S,P,N\right] + F\left[X\right] - \beta Z - D_2\right)\\
\frac{dS}{dT} &= W(t) + L - D_S(S - S_0) - \frac{X\mu_X\left[S,P,N\right]}{Y_{XS}} - \frac{Z\mu_Z\left[S,P,N\right]}{Y_{ZS}}\\
\frac{dP}{dT} &= D_P(P_0 - P) - \frac{X\mu_X\left[S,P,N\right]}{Y_{XP}} - \frac{Z\mu_Z\left[S,P,N\right]}{Y_{ZP}}
\end{aligned}$$

A brief description of the parameters and variables are as follows [6]

- X stands for the concentration of PGPR
- Z is the concentration of micro-organisms
- S denotes concentration of soluble organic compounds.
- P is the amount of molecular oxygen.
- T stands for time
- $\mu_X\left[S,P,N\right]$ is the specific growth-rate which is given by the Monod formula for several limiting resources and the formula is

$$\mu_X\left[S,P,N\right]=\mu_{mX}\frac{S}{S+\theta K_{SX}}\frac{P}{P+K_{PX}}\frac{N}{N+\theta K_{NX}}$$

where μ_{mX}, maximal specific growth rate, K_{SX}, K_{PX}, K_{NX} stand for the affinity constants for the organic substrate, molecular oxygen and mineral nitrogen compounds respectively and θ soil's water content.

- $\mu_Z\left[S,P,N\right]$ is also derived in the same manner as above. The form is

$$\begin{aligned}\mu_Z\left[S,P,N\right]=\mu_{mZ_1}&\frac{S}{S+\theta K_{SZ_1}}\frac{P}{P+K_{PZ_1}}\frac{N}{N+\theta K_{NZ_1}}+\\ \mu_{mZ_2}&\frac{S}{S+\theta K_{SZ_2}}\frac{K_{PZ_2}}{P+K_{PZ_2}}\frac{N}{N+\theta K_{NZ_2}}\end{aligned}$$

where μ_{mZ_1},μ_{mZ_2} are the growth rates for the aerobic and anaerobic parts of the microbes, $K_{SZ_1}, K_{SZ_2}, K_{PZ_1}, K_{PZ_2}, K_{NZ_1}, K_{NZ_2}$ like before are affinity constants for the organic substrate, oxygen and nitrogen with 1 and 2 denoting the aerobic and anaerobic parts, respectively.

- $F(Z) = H_1Z, G(X) = H_2X$ stand for the inter-species interaction between the microflora and PGPR.
- $\alpha X, \beta Z$ denote intra-specific interactions.
- D_1, D_2 are the death coefficients.
- $W(t)$ is a root exudation function which is maximum during day time(i.e. first 12 hours) and minimum during night time(i.e. last 12 hours). For our simulation purposes $W(t)$ has been estimated using a Fourier series calculation.
- L is the rate of decomposition of insoluble carbon compounds.
- $Y_{XS}, Y_{ZS}, Y_{XP}, Y_{ZP}$ are growth yield constants for the substrate and oxygen.

- D_S is the rate of diffusion of soluble carbon from the rhizosphere.
- D_P stands for the diffusion rate of oxygen into the rhizosphere.
- S_0 is the substrate concentration outside the rhizosphere.
- P_0 is the concentration of oxygen in the surrounding area.

The authors developed the model to match experimental results. For the parameters mentioned in the paper there was no chaos detected which can be seen by the time-series portrait of the solutions.[6]

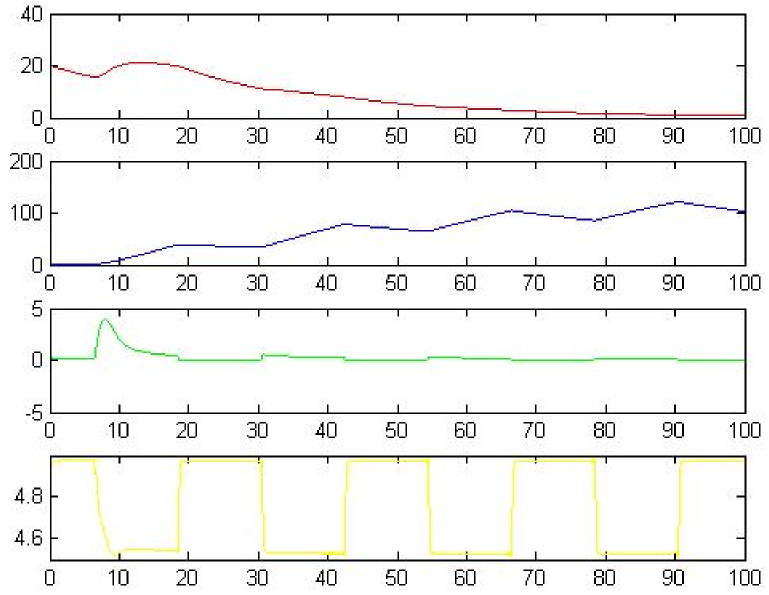


Figure 3.4: Time series plot of the solutions to the system in [6]

3.2.1 Simulation Results

Further analysis of the model in [6] indicates possibly chaos. Based on that observation the initial conditions of the PGPR and micro-organism concentration were varied in the ranges $[0.1, 60]$ and $[0.1, 10]$ respectively. For those values, the maximum Lyapunov exponent was calculated.

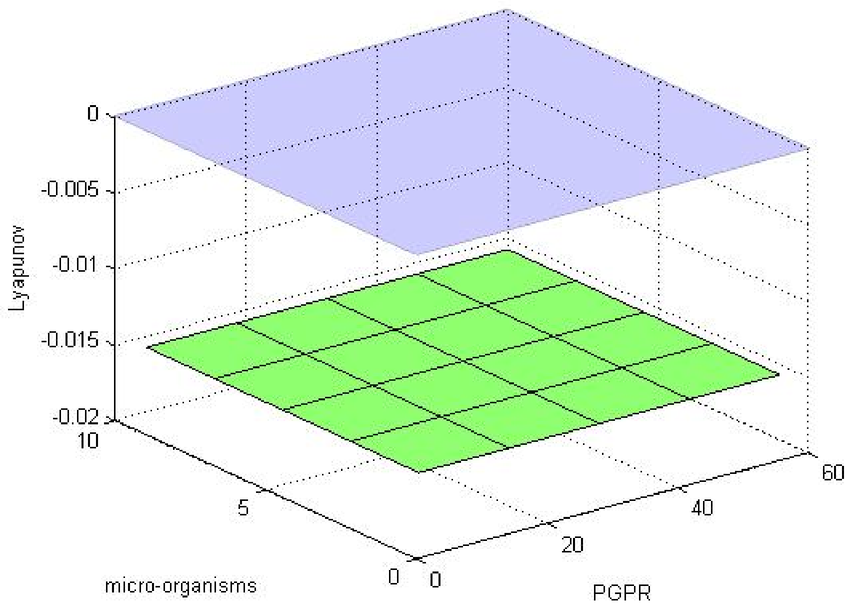


Figure 3.5: 3-D plot of the Lyapunov exponent when X and Z were varied. (The purple denotes the z-plane at 0)

As can be seen in the figure, the maximum Lyapunov exponent stays negative and thus no chaos is seen.

Another choice was the parameter $K_{SX} \in [0.1, 60]$ and the micro-organism concentration. It was seen that though the exponent was still negative, it was closer to zero.

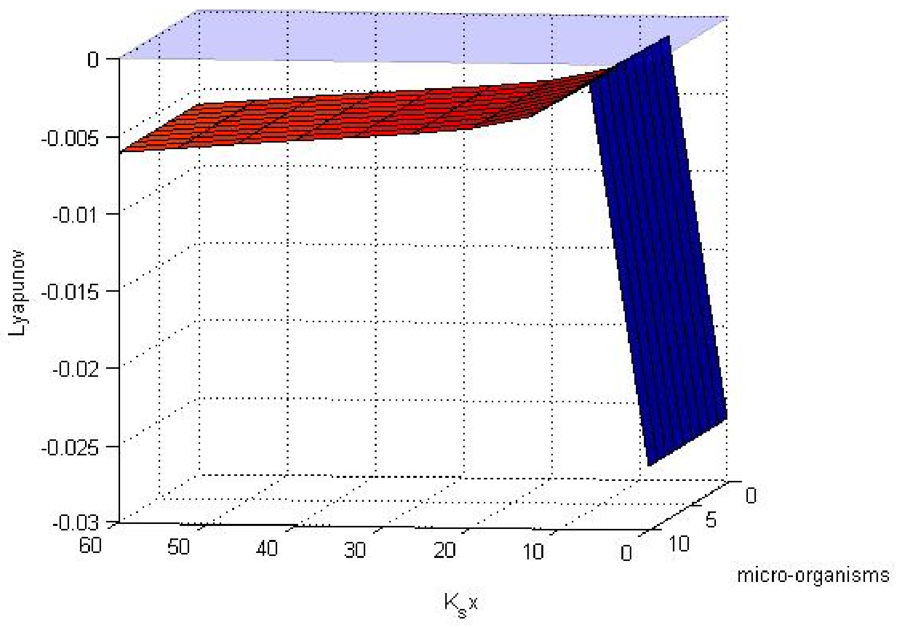


Figure 3.6: 3-D plot of the Lyapunov exponent when K_{SX} and Z were varied

For efficiency we could re-scale the model. Using the transformations $S = \theta K_{SX}s; P = K_{PX}p; N = \theta K_{NX}n; X = Y_{SX}x; Z = Y_{ZS}z; T = \dfrac{1}{\mu_{mX}}t$ we get the following equations:

$$
\begin{aligned}
\frac{dx}{dt} &= x\left[\mu_x(s,p,n) - h_1 z - \alpha_s x - d_1\right] \\
\frac{dz}{dt} &= z\left[\mu_z(s,p,n) - h_2 x - \beta_s z - d_2\right] \\
\frac{ds}{dt} &= \nu_s w(t/\mu_{mX}) + l - d_s(s - s_0) - \frac{x}{y_{xs}}\mu_x(s,p,n) - \frac{z}{y_{zs}}\mu_z(s,p,n) \\
\frac{dp}{dt} &= d_p(p - p_0) - \frac{x}{y_{xp}}\mu_x(s,p,n) - \frac{z}{y_{zp}}\mu_z(s,p,n)
\end{aligned}
\tag{3.10}
$$

where $\mu_x(s,p,n) = \dfrac{s}{s+1}\cdot\dfrac{p}{p+1}\cdot\dfrac{n}{n+1}$

$\mu_z(s,p,n) = \dfrac{\mu_{mZ_1}}{\mu_{mX}}\cdot\dfrac{s}{s+k_{sz_1}}\cdot\dfrac{p}{p+k_{pz_1}}\cdot\dfrac{n}{n+k_{nz_1}} + \dfrac{\mu_{mZ_2}}{\mu_{mX}}\cdot\dfrac{s}{s+k_{sz_2}}\cdot\dfrac{p}{p+k_{pz_2}}\cdot\dfrac{n}{n+k_{nz_2}}$

$$h_1 = \frac{Y_{ZS}H_1}{\mu_{mX}},\ h_2 = \frac{Y_{XS}H_2}{\mu_{mX}},\ \alpha_s = \frac{Y_{XS}\alpha}{\mu_{mX}},\ \beta_s = \frac{Y_{ZS}\beta}{\mu_{mX}},$$

$$
\begin{aligned}
&d_1 = \frac{D_1}{\mu_{mX}},\ d_2 = \frac{D_2}{\mu_{mX}},\ d_s = \frac{D_s}{\mu_{mX}},\ d_p = \frac{D_p}{\mu_{mX}},\\
&\nu_s = \frac{1}{\mu_{mX}\vartheta K_{SX}},\ l = \frac{L}{\mu_{mX}\vartheta K_{SX}},\ s_0 = \frac{S_0}{K_{SX}},\ p_0 = \frac{P_0}{K_{PX}},\\
&k_{sz_1} = \frac{K_{PZ_1}}{K_{SX}},\ k_{sz_2} = \frac{K_{PZ_2}}{K_{SX}},\quad y_{xs} = y_{zs} = \vartheta K_{SX},\\
&k_{pz_1} = \frac{K_{PZ_1}}{K_{PX}},\ k_{pz_2} = \frac{K_{PZ_2}}{K_{PX}},\quad y_{xp} = \frac{K_{PX}Y_{XP}}{Y_{XS}},\ y_{zp} = \frac{K_{PX}Y_{ZP}}{Y_{ZS}},\\
&k_{nz_1} = \frac{K_{NZ_1}}{K_{NX}},\ k_{nz_2} = \frac{K_{NZ_2}}{K_{NX}}
\end{aligned}
$$

It was noticed that on changing just two parameters or initial conditions chaos was not obtained. This gives rise to the question that we might need to investigate more parameters to get a clear idea of whether chaos truly exists in this model.

3.3 System based on Becks paper [1]

In a paper by Becks et.al.,[1], experimental data on a chemostat experiment including 2 preys, a predator and a nutrient was observed and chaotic states were observed for varying levels of the dilution parameter. The model given below was motivated by the results of that paper.

We describe the data using a similar kind of kinetics as described in the previous two models. The general description of the system is given by

$$\frac{dR}{dt} = R\left[\mu_{NR}\left(\frac{N}{K_{NR}+N}\right) - \delta_R\right] - \frac{\mu_{PR}}{Y_{PR}}\left(\frac{R}{K_{PR}+R}\right)P - DR \tag{3.11}$$

$$\frac{dC}{dt} = C\left[\mu_{NC}\left(\frac{N}{K_{NC}+N}\right) - \delta_C\right] - \frac{\mu_{PC}}{Y_{PC}}\left(\frac{C}{K_{PC}+C}\right)P - DC \tag{3.12}$$

$$\frac{dP}{dt} = P\left[\mu_{PR}\left(\frac{R}{K_{PR}+R}\right) + \mu_{PC}\left(\frac{C}{K_{PC}+C}\right) - \delta_P\right] - DP \tag{3.13}$$

$$\frac{dN}{dt} = DN_0 - R\left[\frac{\mu_{NR}}{Y_{NR}}\left(\frac{N}{K_{NR}+N}\right)\right] - C\left[\frac{\mu_{NC}}{Y_{NC}}\left(\frac{N}{K_{NC}+N}\right)\right] - DN. \tag{3.14}$$

The variables R and C represent the prey species of rods and cocci, respectively. We let P represent the predator species, and N represents a nutrient source to the system. The parameter D represents dilution rate for input of nutrients to the system.

The parameters in the model determine the feeding habits of the predator and prey, as well as death and growth rates for each species. These parameters may be used to specify particular behaviors of the organisms, e.g., growth rates due to feeding on nutrient sources rather than prey.

- μ_{N*} is defined as maximum growth rates for the associated species based on consumption of nutrients and

- μ_{P*} denotes the maximum growth rates for predator based on consumption of the associated prey species.

- K_{N*} is the half saturation constant for the species on the nutrient
- K_{P*} is the half saturation constant for the predator on the associated species. Note these latter constants may determine a "preference" for one prey over the other.
- Y_{N*} denotes yield coefficients for the species on the nutrient
- Y_{P*} denotes the yield coefficients for the predator associated with the prey species.
- Death rates for each species are given by δ_*

The model equations and system parameters were estimated by Molz as part of personal correspondence. The intent was to derive a mathematical model whose dynamics closely resembled the experimental dynamics seen in Becks, et.al. [1].

Specific values of the parameters used for the model are provided in Table 3.2.

	Species		
Parameter	R	C	P
μ_{N*}	12 / day	6 / day	
μ_{P*}	2.2 / day	2.2 / day	
K_{N*}	8e-6 gm/cc	8e-6 gm/cc	
K_{P*}	1e-6 gm/cc	1e-6 gm/cc	
Y_{N*}	0.1 gm R / gm N	0.1 gm C / gm N	
Y_{P*}	0.12 gm P / gm R	0.12 gm P / gm C	
δ_*	0.5 / day	0.25 / day	0.08 / day

Table 3.2: Table of values for model equations (3.11) - (3.14) used in the numerical simulations.

3.3.1 Results of Simulation

In [1], the authors found that on varying only the dilution rate, the system went into and out of chaos. So the dilution rate D is a suitable choice to study the dynamics of the system. Another choice is the initial nutrient concentration. They were varied in the ranges $[0.3, 2]$ and $[0.1, 1]$ respectively and the maximum Lyapunov exponent was calculated.

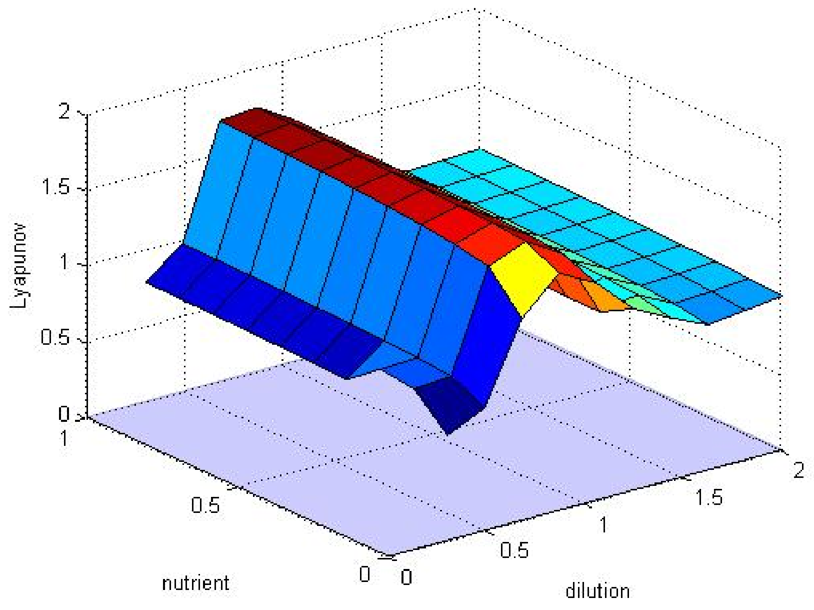


Figure 3.7: 3-D plot depicting chaos and non-chaos with changes in D and N

We observe that the system is in chaos for those choices of the parameters.

Another choice was the dilution rate and concentration of the predator population. It was again observed that the system stays in chaos.

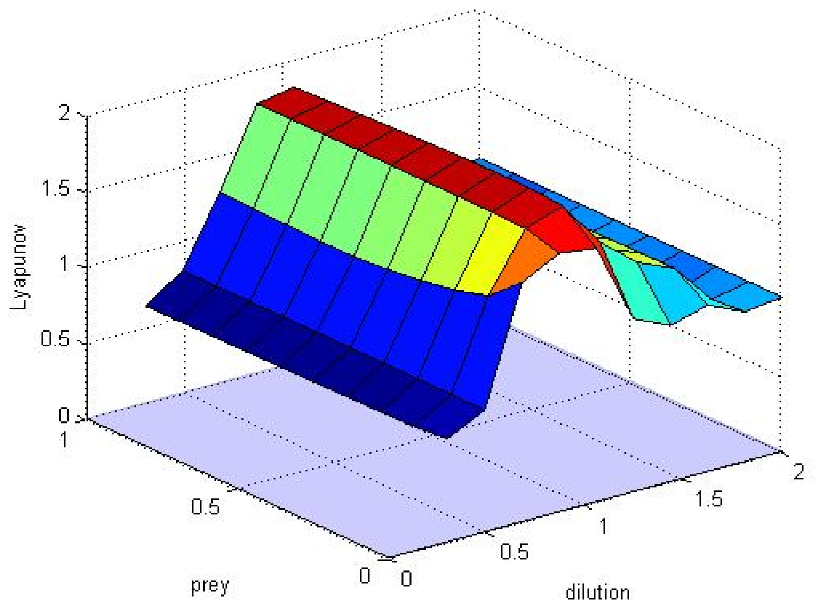


Figure 3.8: 3-D plot depicting chaos and non-chaos with changes in D and P

Chapter 4

Metropolis-Hastings Algorithm

In all the models that were investigated, a common thread was the choice of parameters that lead into chaos. They were chosen based on their significance to the model. But the simulations could only investigate at most two at a time. The entire parameter space was not investigated. For some of the models that involved exploring a 28-dimensional space. An efficient way of doing this would be to use the Metropolis-Hastings Algorithm. All the notations that follow are taken from Chib and Greenberg 1995 [4]

4.1 The Algorithm

Our objective is to generate those parameter values which would give us a positive Lyapunov exponent. Thus, we need an efficient algorithm for accepting or rejecting a possible combination of parameters. Suppose we have a starting point x and we wish to move to another point y in the space. We do so by introducing a probability that the move is made. If the move is not made we return to our previous point. Thus transitions from x to y are given by:

$$q(x,y)\alpha(x,y)$$

where $q(x,y)$ is the *candidate generating density* and $\alpha(x,y)$ is the *probability of move.* We assume both x and y are generated from a probability distribution $\pi(*)$. Now we define the probability of the move as follows:

$$\alpha(x,y) = \begin{cases} \min\left[\dfrac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right] & \text{if } \pi(x)q(x,y) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.1}$$

The choice of the candidate generating density is ours. It could be based on the Lyapunov exponents. The algorithm is as follows:

- Initialize with x^0.
- For $j = 1, 2, \ldots, N$.
  - Generate y from $q(x^j, *)$ and u from Uniform(0,1).
  - Set

$$x^{j+1} = \begin{cases} y & \text{if } u \leq \alpha(x^j, y) \\ x^j & \text{otherwise} \end{cases} \tag{4.2}$$

  - Return the values $\left\{x^1, x^2, \ldots, x^N\right\}$.

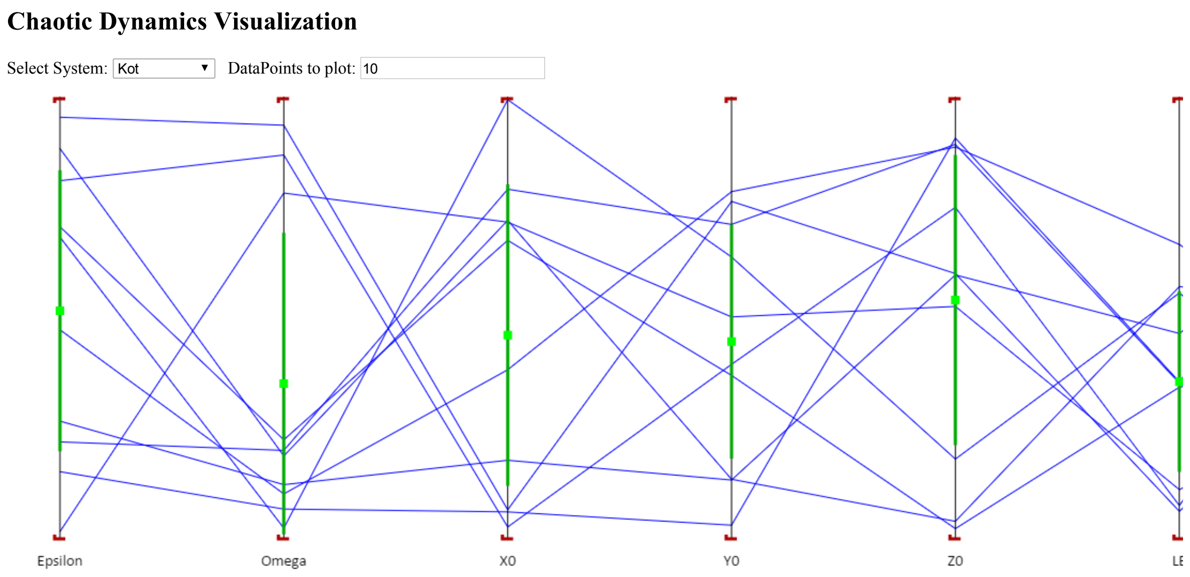
It should be noted that we do not need knowledge of the normalizing constant(it is called so since $\pi(x)$ and $\pi(y)$ are constant for given values of x and y) $\pi(*)$ because it appears in both the numerator and denominator. Also we are assuming that $q(x,y)$ need not be $q(y,x)$ i.e., the candidate generating density need not be symmetric.

Chapter 5

Conclusions

Most deterministic dynamical systems go into chaos for some values of their parameters. There are many ways to measure chaos. One popular way uses Lyapunov exponents. The paper by Wolf et.al.,[2] proposed the frequently used choice of calculating such exponents using Gram-Schmidt orthonormalization process. The work in this thesis centered on coding and verifying the algorithm, as well as using the code to investigate three biological models to find parameters/initial conditions to give chaos. It was also noted that the Metropolis-Hastings algorithm can be used as an effective way of investigating the parameter space to obtain chaotic behavior. This can be done by using the Metropolis-Hastings sampler to move from one point in the paramater space to another in an effective way.

A possible way to depict higher-dimensional results would be to use the parallel-coordinates plot. This would help us to understand the interaction between the parameters and the initial conditions. Implementation of all of these components could help in analyzing further models. An example of such a plot for the Kot system is given below:

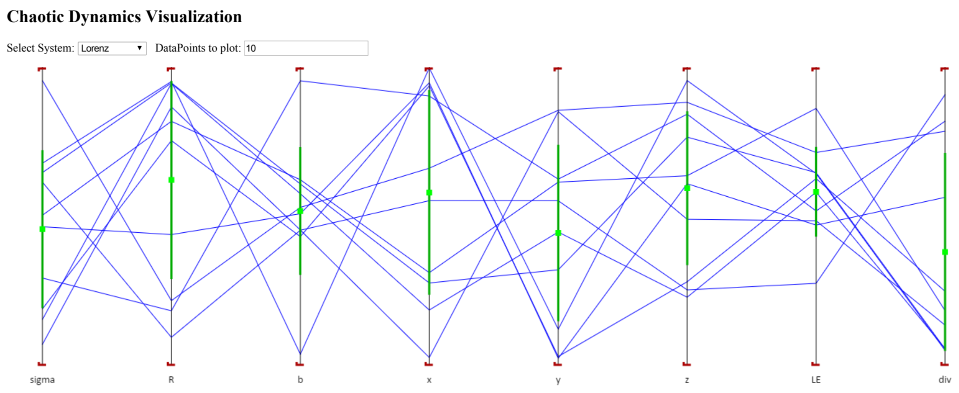

Chaotic Dynamics Visualization

Select System: Kot DataPoints to plot: 10

Figure 5.1: Parallel Coordinates Plot of ϵ, ω, initial values of the variables x, y, z of the Forced System in [7] and the Maximum Lyapunov Exponent

The figure helps us understand the values of the parameters for which we get a positive Lyapunov exponent and thus chaos.

Another example for the Lorenz system with all the parameters varied is shown below:


Chaotic Dynamics Visualization

Select System: Lorenz DataPoints to plot: 10

Figure 5.2: Parallel Coordinates Plot of ρ, β, σ, initial values of the variables x, y, z of the Lorenz system and the Maximum Lyapunov Exponent

Further mathematical analysis of the models may give us a proper starting value for the Metropolis sampling so that we don't take a shot in the dark. Another possible avenue of research would be to consider if chaos is a proper indicator of the health of the system. In other words, if we can properly convert a dynamical system into a optimization/game-theoretic model, we can analyze if chaotic behavior is an optimal solution.

To conclude, the calculation of Lyapunov exponents and sampling of parameter space would give us a better understanding of some models which may be difficult to analyze otherwise.

Appendices

Appendix A MATLAB code for determining Lyapunov Spectrum

This appendix includes the MATLAB code for determining the Lyapunov Spectrum. It is based on the paper by Wolf et al [2].

```
%Code from Wolf paper
%Before passing f make sure f=@system where system is function
%corresponding to your system

function lyapOut = myLyap(f,p,Initial,t,ts)%f is the ode system,
%p is the parameter set, init is the initial conditions,
%t is the time interval for the ode solver, ts is the timestep

%N = number of nonlinear equations,
%NN = Total number of equations

N=length(Initial); %length(Initial) gives us the size of original system
NN=N*(N+1);

% initialize arrays

Y=zeros(NN,1);
CUM=zeros(N,1);
GSC= zeros(N,1);
```

```
znorm=zeros(N,1);

y0 = Y;


lyap = zeros(N,1);

S=zeros(N,1);


len = round((t(2)-t(1))/ts);


for i = 1:N

    Y(i,1) = Initial(i);

end


%Initial Conditions for linear system(Orthonormal frame)


for i =1:N

    Y((N+1)*i,1) = 1.0;

end;


tstart = t(1);


for iterLyap=1:len

    [tvals,y] = ode45(@(t,y)(f(t,y,p)), [tstart,tstart+ts], Y);


    Y = y(size(y,1),:)';


    for i = 1:N
    for j = 1:N

        y0(N*i+j,1) = Y(N*i+j,1);

    end

end


tstart=tstart+ts;


%Construct a new orthonormal basis by Gram-Schmidt Method

%Normalize first vector

znorm(1,1)=0.0;

for j=1:N

    znorm(1,1)=znorm(1,1) + y0(N*j+1,1)^2;

end;

znorm(1,1)=sqrt(znorm(1,1));

for j=1:N

    y0(N*j+1,1) = y0(N*j+1,1)/znorm(1,1);

end;


%Generate the new orthnormal set of vectors

for j=2:N


    %Generate j-1 GSR coefficients

    for k= 1:j-1

        GSC(k,1) =0.0;

        for l=1:N

            GSC(k,1) = GSC(k,1) + y0(N*l+j,1)*y0(N*l+k,1);
        end;
    end;

    %Construct a new vector
    for k=1:N
        for l=1:j-1
            y0(N*k+j,1) = y0(N*k+j,1) - GSC(l,1)*y0(N*k+l,1);
        end;
    end

    %calculate the vector's norm
    znorm(j,1) =0.0;
    for k=1:N
        znorm(j,1)= znorm(j,1) + y0(N*k+j,1)^2;
    end;
    znorm(j,1) =sqrt(znorm(j,1));

    %normalize the new vector
    for k=1:N
        y0(N*k+j,1) = y0(N*k+j,1)/znorm(j,1);
    end;
end;

%update running vector magnitudes
for k=1:N
    CUM(k,1) = CUM(k,1) + log(znorm(k,1))/log(2.0);
    end;


    %normalize exponent and print every 10 iterations

    %if (rem(i,10)== 0)

    for k=1:N

        lyap(1,k) = CUM(k,1)/(tstart-t(1));

    end;


    if iterLyap == 1

        lyapExp = lyap;

    else

        lyapExp = [lyapExp; lyap];

    end


    for i = 1:N

        for j = 1:N

           Y(N*j+i,1) = y0(N*j+i,1);

        end

    end


    lyapOut = lyap(1,1:N);


end
```

Appendix B MATLAB code for the mathematical models

B.1 The Kot system

```
%Kot Equations
function YPRIME= kotfn1(t,x,p)
YPRIME = zeros(12,1);

    D = p(1);si=p(2);mu1=p(3);mu2=p(4);y1=p(5);y2=p(6);k1=p(7);k2=p(8);eps=p(9);om= p
    A = mu1/D ;
    a = k1/si ;
    B = mu2/D ;
    b = k2/y1/si ;
YPRIME(1,1) = 1 + eps*sin(om*t) - x(1,1) - ...
(A*x(1,1)*x(2,1)/(a+x(1,1)));

YPRIME(2,1) = (A*x(1,1)*x(2,1))/(a+x(1,1)) - ...
x(2,1) -( B*x(2,1)*x(3,1))/( b+x(2,1) );

YPRIME(3,1) = (B*x(2,1)*x(3,1))/(b+x(2,1)) - x(3,1) ;



% Copies of linearized equations of motion
```

```
    for j=0:2
        YPRIME(4+j,1) = x(4+j,1)*(-1-(A*x(2,1))/(a+x(1,1)) + ...
        (A*x(2,1)*x(1,1))/(a+x(1,1))^2) - ...
        x(7+j,1)*((A*x(1,1))/(a+x(1,1)));
        YPRIME(7+j,1) = x(4+j,1)*((A*x(2,1))/(a+x(1,1))-...
        (A*x(1,1)*x(2,1))/((a+x(1,1))^2))+...
        ((A*x(1,1))/(a+x(1,1))-1-(B*x(3,1))/(b+x(2,1))+...
        B*x(2,1)*x(3,1)/(b+x(2,1))^2)*x(7+j,1)-...
        ((B*x(2,1))/(b+x(2,1)))*x(10+j,1);
        YPRIME(10+j,1) = (B*x(3,1)/(b+x(2,1))-...
        (B*x(3,1)*x(2,1))/(b+x(2,1))^2)*x(7+j,1)+...
        ((B*x(2,1))/(b+x(2,1))-1)*x(10+j,1);
    end;
    end
```

B.1.1 The Kravchenko system

```
%Kravchenko Equations
function xp= Krav_original(t,x,p)
xp = zeros(20,1);

H1 = p(1) ;  % original
H2 = p(2) ;   % original

alpha = p(3);  % original
beta = p(4) ;  % original
```

```
D1 = p(5) ;

D2 = p(6);

R = p(7) ;

r = p(8) ;


DS = 2*R*1e-6/(R-r)^2/(R+r) ;


DP = 2*R*1e-3/(R-r)^2/(R+r) ;

% DS = p(7);

%

% DP = p(8);


L = p(9) ;

S0 = p(10);


P0 = p(11); % original


KSZ1 = p(12); % original


KSZ2 = p(13); % original


YXS = p(14);

YZS = p(15);

KPZ1 = p(16) ;
```

```
KPZ2 = p(17);

YXP = p(18);


YZP = p(19);

KNZ1 = p(20);

KNZ2 = p(21);

MUZ1 = p(22);

MUZ2 = p(23);

MUX = p(24);

N = p(25);

theta = p(26);

KSX = p(27);

KPX = p(28);

KNX = p(29);


%Defining the function Mu_x(s,p,n)

Mu_x = MUX*((x(3,1)/(x(3,1)+theta*KSX))*...

  (x(4,1)/(x(4,1)+KPX))*(N/(N+theta*KNX)));


%Defining the functions Mu_z1(s,p,n) and Mu_z2(s,p,n)

Mu_z1 = (MUZ1)*((x(3,1)/(x(3,1)+KSZ1*theta))*...

(x(4,1)/(x(4,1)+KPZ1))*(N/(N+KNZ1*theta)));

Mu_z2 = (MUZ2)*((x(3,1)/(x(3,1)+KSZ2*theta))*...

(KPZ2/(x(4,1)+KPZ2))*(N/(N+KNZ2*theta)));


%Evaluating the root exudation function
```

```
f = 4;

for n =1:100

    f = f + 8/n/pi*( sin(18.5*n*pi/12) - sin(6.5*n*pi/12) )*cos(n*pi*t/12);

    f = f + 8/n/pi*( cos(6.5*n*pi/12) - cos(18.5*n*pi/12) )*sin(n*pi*t/12);

end

%Original System of equations

xp(1,1) = x(1,1)*(Mu_x+H1*x(2,1)-alpha*x(1,1)-D1) ;


xp(2,1) = x(2,1)*(Mu_z1 + Mu_z2+H2*x(1,1)-beta*x(2,1)-D2);


xp(3,1) = f+L - DS*(x(3,1)-S0)-((x(1,1)/YXS)*Mu_x)-(x(2,1)/YZS)*(Mu_z1+Mu_z2);


xp(4,1) = DP*(P0-x(4,1)) -((x(1,1)/YXP)*Mu_x)-(x(2,1)/YZP)*(Mu_z1+Mu_z2);



% Copies of linearized equations of motion

for j=0:3

    xp(5+j,1) = (Mu_x-2*alpha*x(1,1)+H1*x(2,1)-D1)*x(5+j,1)+H1*x(1,1)*x(9+j,1)+...

        (x(1,1)*((Mu_x*theta*KSX)/(x(3,1)*(x(3,1)+theta*KSX))))*x(13+j,1)+...

    (x(1,1)*((Mu_x*KPX)/(x(4,1)*(x(4,1)+KPX))))*x(17+j,1);

    xp(9+j,1) = H2*x(2,1)*x(5+j,1)+(Mu_z1+Mu_z2-2*beta*x(2,1)+H2*x(1,1)-D2)*x(9+j,1)

        x(2,1)*(((theta*KSZ1*Mu_z1)/(x(3,1)*(x(3,1)+KSZ1*theta))+...

        (((KSZ2*Mu_z2*theta)/(x(3,1)*(x(3,1)+KSZ2*theta)))))*x(13+j,1)+...

        x(2,1)*(((KPZ1*Mu_z1)/(x(4,1)*(x(4,1)+KPZ1)))-...

        (((Mu_z2)/((x(4,1)+KPZ2)))))*x(17+j,1);

    xp(13+j,1) = -(Mu_x/YXS)*x(5+j,1)-((Mu_z1+Mu_z2)/YZS)*x(9+j,1)-...
```

```
        (DS+x(1,1)*(Mu_x*theta*KSX)/(YXS*x(3,1)*(x(3,1)+theta*KSX))+...
        (x(2,1)/YZS)*(((theta*KSZ1*Mu_z1)/(x(3,1)*(x(3,1)+theta*KSZ1))+...
        (((theta*KSZ2*Mu_z2)/(x(3,1)*(x(3,1)+KSZ2*theta))))))*x(13+j,1)-...
        (x(1,1)*(KPX*Mu_x)/(YXS*x(4,1)*(x(4,1)+KPX))+...
        (x(2,1)/YZS)*(((KPZ1*Mu_z1)/(x(4,1)*(x(4,1)+KPZ1)))-...
        (((Mu_z2)/((x(4,1)+KPZ2))))))*x(17+j,1);
    xp(17+j,1) = -(Mu_x/YXP)*x(5+j,1)-((Mu_z1+Mu_z2)/YZP)*x(9+j,1)-...
        (x(1,1)*(Mu_x*theta*KSX)/(YXP*x(3,1)*(x(3,1)+theta*KSX))+...
        (x(2,1)/YZP)*(((theta*KSZ1*Mu_z1)/(x(3,1)*(x(3,1)+KSZ1*theta))+...
        (((theta*KSZ2*Mu_z2)/(x(3,1)*(x(3,1)+KSZ2*theta))))))*x(13+j,1)-...
        (DP+x(1,1)*(KPX*Mu_x)/(YXP*x(4,1)*(x(4,1)+KPX))+(x(2,1)/YZP)*...
        (((KPZ1*Mu_z1)/(x(4,1)*(x(4,1)+KPZ1)))-...
        (((Mu_z2)/((x(4,1)+KPZ2))))))*x(17+j,1);
end
end
```

B.2 The Becks System

```
%Becks Equations
function xp= Becks(t,x,p)
xp = zeros(20,1);
% growth rates (1/sec)
% values below are in 1/day; divide by 24*3600 to get seconds
munr = p(1) ;  % original
munc = p(2) ;   % original
```

```
mupr = p(3);  % original

mupc = p(4) ;  % original


% mass (g)

mr = p(5) ;

mc = p(6);

mp = p(7);

%

% half-saturation constants (g/cc)

knr = p(8);

knc = p(9) ;

kpr = p(10) ;

kpc = p(11);

Kpr = kpr/mr;

Kpc = kpc/mc;


% death rates (1/sec)

% values below are in 1/day; divide by 24*3600 to get seconds

dr = p(12); % original


dc = p(13); % original


dp = p(14); % original


% initial nutrient (g/cc)
```

```
N0 = p(15); % original

% N0 = 2.3e-5 ;


%

% yield coefficients

ypr = p(16);

ypc = p(17) ;

ynr = p(18);

ync = p(19);

%

% dilution rate

% values below are in 1/day; divide by 24*3600 to get seconds


D = p(20);  % original

xp(1,1) = x(1,1)*(munr*x(4,1)/(x(4,1)+knr) - dr) -...

          mupr/ypr*mp/mr*x(1,1)/(Kpr+x(1,1))*x(3,1) - D*x(1,1) ;


xp(2,1) = x(2,1)*(munc*x(4,1)/(x(4,1)+knc) - dc) - ...

          mupc/ypc*mp/mc*x(2,1)/(Kpc+x(2,1))*x(3,1) - D*x(2,1) ;


xp(3,1) = x(3,1)*(mupr*x(1,1)/(Kpr+x(1,1)) + ...

  mupc*x(2,1)/(Kpc+x(2,1)) - dp) - D*x(3,1) ;


xp(4,1) = D*N0 - x(1,1)*munr*mr/ynr*x(4,1)/(knr+x(4,1)) - ...

  x(2,1)*munc*mc/ync*x(4,1)/(knc+x(4,1)) - D*x(4,1);
```

```
% Copies of linearized equations of motion
for j=0:3
    xp(5+j,1) = ((munr*x(4,1))/(x(4,1)+knr)-dr-mupr/ypr*mp/mr*x(3,1)/(Kpr+x(1,1))+...
     mupr/ypr*mp/mr*x(1,1)/((Kpr+x(1,1))^2)*x(3,1))*x(5+j,1)-...
    (mupr/ypr*mp/mr*x(1,1)/(Kpr+x(1,1)))*x(13+j,1)+...
    x(1,1)*x(17+j,1)*(munr/(x(4,1)+knr)-munr*x(4,1)/(x(4,1)+knr)^2);
    xp(9+j,1) = x(9+j,1)*(munc*x(4,1)/(x(4,1)+knc) - dc-...
    mupc/ypc*mp/mc*x(3,1)/(Kpc+x(2,1))+...
    mupc/ypc*mp/mc*x(2,1)/(Kpc+x(2,1))^2-D)-...
    (mupc/ypc*mp/mc*x(2,1)/(Kpc+x(2,1)))*x(13+j,1)+...
    (x(2,1)*(munc/(x(4,1)+knc)-munc*x(4,1)/(x(4,1)+knc)^2))*x(17+j,1);
    xp(13+j,1) = x(3,1)*x(5+j,1)*((mupr/(Kpr+x(1,1))-mupr*x(1,1)/(Kpr+x(1,1))^2)) + .
    x(3,1)*x(9+j,1)*((mupc/(Kpc+x(1,1))-mupc*x(1,1)/(Kpc+x(1,1))^2))+...
    x(17+j,1)*(mupr*x(1,1)/(Kpr+x(1,1)) + mupc*x(2,1)/(Kpc+x(2,1)) - dp-D)  ;
    xp(17+j,1) = -x(5+j,1)*(munr*mr/ynr*x(4,1)/(knr+x(4,1))) - ...
    x(9+j,1)*munc*mc/ync*x(4,1)/(knc+x(4,1)) +...
    x(17+j,1)*(- x(1,1)*munr*mr/ynr/(knr+x(4,1))+ ...
    x(1,1)*munr*mr/ynr*x(4,1)/(knr+x(4,1))^2- ...
    x(2,1)*munc*mc/ync/(knc+x(4,1))+ x(2,1)*munc*mc/ync*x(4,1)/(knc+x(4,1))^2-D);
end;
end
```

Bibliography

[1] Lutz Becks, Frank M.Hilker, Horst Malchow, Kalus Jürgens, Hartmut Arndt. Experimental demonstration of chaos in a microbial food web. *Nature03627*, 435, 2005.

[2] Alan Wolf, Jack B.Swift, Harry L.Swinney, John A.Vastano. Determining lyapunov exponents from a time series. *Physica*, Volume 16D:Pg.285–317, 1985.

[3] Kathleen T.Alligood, Tim D.Sauer, James A.Yorke. *Chaos: An Introduction to Dynamical Systems.* Springer-Verlag New York Inc, 1997.

[4] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 1995.

[5] Liz Bradley , Dept.of Computer Science , Uty . of Colorado. The variational equation notes for a course.

[6] Nikolay S.Strigul, Lev V.Kravchenko. Mathematical modeling of PGPR inoculation into the rhizosphere. *Environmental Modeling and Software*, 21:1158–1171, 2006.

[7] Mark Kot, Gary S.Sayler, Terry W.Schultz. Complex dynamics in a model microbial system. *Bulletin of Mathematical Biology*, Vol No.54(No.4):Pg.619–648, 1992.