Math 801
(A. Assadi)

Poincaré Section, Chaos and Stability

Control Parameters - The variation of a control parameter gives rise to drastically different qualitative behavior in some dynamical systems. Below, we briefly discuss the phenomenon of chaos which is defined to be defined to be *aperiodic bounded dynamics in a deterministic system with sensitive dependence on initial conditions*. The first studies of chaos go back to the French universalist mathematician Henri Poincaré and his famous memoirs on celestial mechanics. Study of chaotic dynamics became a great deal more widespread with the advent of digital computers. The term *chaos*, however, was coined only in the 1975, by York et al. A very useful topological construction called the Poincaré section and some well-known examples of the famous nonlinear maps. Also, we mention a few basic insights into chaotic dynamical systems, for example, that *chaos is not a random process*. In a nutshell, the property that distinguishes a chaotic solution to a (deterministic) equation is that it is highly sensitive to the initial conditions and wanders all over the place. This wandering may appear to be random, but mathematically it cannot be called random in view of the Fundamental Theorem of Existence and Uniqueness of ODE systems. Note that there could be solutions that alter sharply after small changes in initial conditions, but do not necessarily exhibit what we would call *random-like behavior*. Such trajectories are not regarded as chaotic.

The Poincaré Section

In the discussion of the Hopf bifurcation in Beltrami's textbook, the stability of the limit cycle was discussed. There is another method of exploring this property; it leads us to construct a new mathematical structure for a dynamical system, called the *Poincaré section* of a limit cycle. If the dynamical system is two-dimensional, then a Poincaré section is also called a Poincaré line, and for 3-dimensional systems, it is also known as a Poincaré plane.

First, recall that two trajectories are considered *distinct* when the starting point of one does not lie on the other. The Fundamental Theorem of Existence and Uniqueness of ODEs implies that *distinct trajectories in the phase plane* (or more generally, in the phase space of any dimension N) *cannot intersect in finite time*. This basic observation allows us to associate a well-defined collection of points $\{P_1, P_2, P_3, \ldots\}$ on a line (or more generally, in a linear subspace of dimension N-1 when the phase space of has dimension N.) The *Poincaré section* is formed by the following procedure. Take a line segment in the phase plane that intersects the limit cycle, for instance, at a right angle to its tangent line at the chosen point (similarly, for the N-dimensional phase-space, take a linear subspace of dimension N-1 that is the orthogonal complement of the tangent line to the limit cycle, that is, perpendicular to the vector field at the selected point.) The choice of a right angle is for convenience sake, and the arguments work equally well when any convenient line is chosen so long as it is NOT a tangent line of the limit cycle. Let P_0 be a point of intersection with the limit cycle. Start a trajectory from a point near to, but not on, the limit cycle. Follow the trajectory until it crosses the line of the Poincaré section at a point, say P_1. The following the flow one more time around, the intersection will be at P_2, and the next one at P_3 and so on. If the sequence of points $\{P_1, P_2, P_3, \ldots\}$ approaches P_0,

the trajectory is tending to the limit cycle. If the sequence leaves P_0, for instance as in the case for a set of points $\{Q_1, Q_2, Q_3, \ldots\}$, the trajectory is going away from the limit cycle. Thus, a *stable limit cycle* can be identified by the sequences on both sides of P_0 tending to P_0. A sequence always stays on one side of P_0 because distinct trajectories in the phase plane *cannot intersect in finite time*. It is useful to know that the effort to determine the mapping that gives the Poincaré line (or higher dimensional sections) is in general equivalent to solving the original system of ODEs. Nonetheless, the method of Poincaré section opens new possibilities for qualitative arguments that are not as transparent if we were to deduce them analytically from solution of the system.

When is a trajectory in the phase plane *chaotic*? A more basic question is: Which solutions of differential equations constitute *a chaotic trajectory*? Consider systems in which the trajectories stay in a bounded region, so that there is no possibility of the solution becoming infinite. It is reasonable to expect the conditions for *chaos* to be the following, of which, the first two are basic in any dynamical system when we wish to follow a single flow line that enters a bounded region $\mathbf{\Omega}$ and leaves it and re-enters it repeatedly, for instance because $\mathbf{\Omega}$ is rather small. (1) Distinct trajectories do not intersect; (2) the trajectories are bounded, and (3) trajectories, which are initially close, diverge in a rapid manner and widely. Not all of these conditions can be met by trajectories in the phase plane so that *chaos does not arise for the phase plane or Poincaré line.* The situation is different in *phase space* with three or more dimensions. Here the trajectories can intertwine without intersection and stay in a bounded region so that chaos is possible. A chaotic trajectory cannot return to its initial point; if it did, it would correspond to a periodic solution of the differential system. It might, however, return to near the initial point but then, on account of (c), its subsequent path would bear little or no resemblance to the path from the

initial point. The system of equations, discovered by Lorenz in late 1950s in the course of his research on dynamics of climate, is an example of a 3-dimensional chaotic dynamical system. The first applications of chaos and nonlinear dynamics to physiology and human disease is discussed by Glass and Mackey (1988).

(**Reference:** Lorentz E N (1963): Deterministic nonperiodic flow. Journal of the Atmosphairic Sciences 20: 130-141.)

```
% Values of parameters
sigma = 10;
R = 28;
beta = 8/3;
%Lorenz equation
f_1=sigma*(y-x);
f_2= -x*z + R*x-y;
f_3= x*y-beta*z;
%Linearized system
L=[-sigma, sigma,     0;
         R-z,    -1,    -x;
           y,     x, -beta];
```

Bifurcation in Discrete Dynamical Systems

Below, we look at some models of dynamic processes in biology. Simple formulas relate, for instance, the population of a species in a certain year to that of the following year. We learn to understand the consequences an equation might have through mathematical analysis, so that our formulation can be checked against biological observation. Although many of the models we examine may at first seem to be gross simplifications, their very simplicity is strength. Simple models show clearly the implications of our most basic

assumptions. We begin by focusing on modeling the way populations grow or decline over time. Since mathematical models should be driven by questions, here are a few to consider: Why do populations sometimes grow and sometimes decline? Must populations grow to such a point that they are unsustainably large and then die out? If not, must a population reach some equilibrium? If equilibrium exists, what factors are responsible for it? Is such equilibrium so delicate that any disruption might end it? What determines whether a given population follows one of these courses or another?

Consider a simple scenario in population dynamics. Cells reproduce by division; the process by which the cell nucleus divides is called *mitosis*. One way to regulate the rate of reproduction of cells is by regulating mitosis. There is a certain body of biochemical evidence that there are compounds that are tissue-specific inhibitors of mitosis (Reference: Bullough and Laurence, 1968). For simplicity, assume that the generations of cells are distinct and that the number of cells in each generation is given by y_t. Following the same, assume that for each cell in generation t, there are ξ cells in generation $t + 1$. For instance, if every cell divided in half every time step, then ξ would equal 2. The finite-difference equation describing this situation is the linear equation $N_{t+1} = \xi N_t$, which leads either to exponential growth or to decay to zero. There are other scenarios that leads to recurrence equations similar to the one mentioned above. Rather than getting into the controversy of the mechanisms of mitosis, let us suppose that a population of microorganisms, such as fungi, has the following pattern of population of growth in cycles that are all of length T. Let us refer to any biological form of seeds or dormant off-springs in the course of

reproduction as *yield*. At the end of the first growth cycle, the yield is y_1 and the fungi population succeeds in preserving a proportion of this yield in dormant form for the next growth cycle. Assume perfect germination and a negligible loss of yield. The quantity y_2 at the end of the second growth cycle will be proportional to y_1, for example through an equation $y_2 = \xi y_1$. If the fungi population undergoes the same process at the end of each growth cycle, we have a general formula $y_{n+1} = \xi y_n$, from which by induction, we conclude $y_n = \xi^n y_1$. This formula shows that the yield increases steadily from growth cycle to growth cycle when $\xi > 1$. Now consider the opposite case when $\xi < 1$, the yield decays towards zero. There is a bifurcation as ξ varies in values near $\xi = 1$ from a stable state to an unstable state. A more realistic assumption, of course, would be that ξ varies with time from growth cycle to growth cycle, and that the formula for estimating the yield would be estimated from above and below by some constants $y_{n+1} = \xi y_n,\ c \le \xi(t) \le C$. To analyze the situation in this case, we estimate a fixed multiple of cycles that we call τ by analogy with T, and we use an average value of $\xi(t)$ over the period τ, so that we could approximately repeat the same argument for every time interval τ, and make the same qualitative conclusion. We must also take into account losses due to poor environmental conditions arising, for instance, due to change of climate or invasion of predators or waste due to parasites. Further, there could be physical constraints on the size of the population and their yield due to limited quantities of the nutrients that the organisms' environment could support. For example, we could envision that in one such circumstance, we have the relation $y_{n+1} = \xi y_n - \eta y_n^2$ with both parameters estimated to be positive constants, so that η

term represents losses. The reason behind taking a quadratic power is to inhibit the growth if the yield becomes too large. Now, in the new formula, make the substitution $y_n = \xi x_n / \eta$ and compute:

$$x_{n+1} = \xi x_n (1 - x_n) .$$

Since yields are positive numbers, x_n must be restricted to lie between 0 and 1. Note that ξ cannot exceed 4. If ξ becomes greater than 4, it would generate values of x_n outside the permitted range. For example, $x_n = 0.5$ would make x_{n+1} too large. The equation $x_{n+1} = \xi x_n (1 - x_n)$ is called an *iteration scheme* in which each value is determined from its predecessor once x_1 has been fixed. If x_n tends to a limit x_0 as $n \to \infty$ then x_0 satisfies $x_0 = \xi x_0 (1 - x_0)$. The solutions of $x_0 = \xi x_0 (1 - x_0)$ are known as the *fixed points* of the iteration scheme. In this example, the fixed points are $x_0 = 0,\ x_0 = 1 - 1/\xi$. Because x_0 must lie between 0 and 1, the answer $x_0 = 1 - 1/\xi$ cannot be applied unless $\xi < 1$. Consequently, there are two fixed points when $\xi > 1$, but only one fixed point when $\xi < 1$. Note that, when $x_n \to 1 - 1/\xi$, then $y_n \to (\xi - 1)/\eta$.

Observe that, if $x_1 = 1$, $x_2 = 0$ and all subsequent x_n are zero. So the iteration goes to the fixed point $x_0 = 0$ whenever $x_1 = 1$. Accordingly, it will be sufficient to limit x to $x_1 < 1$ subsequently. You are invited to repeat the methods of last lecture I order to study the stability properties of the dynamics near a fixed point by looking at small perturbations and extracting the linear part from it. You would conclude then, that the fixed point $X_0 = 0$ is stable for $\xi > 1$ and unstable for $\xi < 1$. Moreover, the fixed point is $1 - 1/\xi$ stable for $1 < \xi < 3$ and unstable $\xi > 3$.

Global Stability Of Fixed Points

Local stability tells us whether the fixed point is approached if the initial condition is sufficiently close to the fixed point. The local stability can be assessed simply by looking at the slope of the function at the fixed point. A slightly different—and often much more difficult—question is whether a locally stable fixed point is globally stable.

For linear finite-difference equations, the answer is that a locally stable fixed point is also globally stable: Regardless of the initial condition, the iterates will eventually reach the locally stable point (i.e., the origin) from any initial condition. For nonlinear finite-difference equations, there can be more than one fixed point. When multiple fixed points are present, none of the fixed points can be globally stable! The set of initial conditions that eventually leads to a fixed point is called the basin of attraction of the fixed point. Often, the basin of attraction for fixed points in nonlinear systems can have a very complicated geometry. When there are multiple fixed points that are locally stable, the system is said to have multi-stability.

Another qualitative behavior for discrete dynamical systems generated by finite-difference equations is having *periodic cycles*. Informally, a cycle is a pattern that repeats itself. The period of a cycle is the length of time between repetitions. Fixed points are special examples of periodic cycles of period equal to one. Ore generally, in finite-difference equations, a cycle arises when $x_{t+N} = x_t$, $x_{t+k} \neq x_t$ for all $k=1, 2$,

..., N-1. For example, a cycle of period 2 exists when $x_{t+1} \neq x_t$ but $x_{t+2} = x_t$, as in the dynamical system given by the finite difference equation: $x_{t+1} = f(x_t) = 3.3(1-x_t)x_t$. To find cycles of period 2, we need to set $x_{t+2} = f(f(x_t))=f(x_{t+1})=3.3(1-x_{t+1})x_{t+1}$ equal to x_t. Note that fixed points of *f* are not cycles of period 2; rather, they have period one.

Just as in continuous dynamical systems, a fixed point can be locally stable or unstable, also a cycle can be stable or unstable. We say that a cycle is locally stable if, given that the initial condition is close to a point on the cycle, subsequent iterates approach the cycle. This concept was called *local asymptotic stability* in the continuous case. Let us consider the computation of the stability of the fixed point of the finite-difference equation $x_{t+2} = x_t$, $f(f(x_t))=f(x_{t+1})=3.3(1-x_{t+1})x_{t+1}$. Call a solution to this equation q. Stability in the continuous dynamical systems requires the value of the derivative to be negative. For discrete dynamical systems, we have a similar rule, so we need to examine the value of the following at $x_t = q$, which we compute using the chain rule:

$$\frac{df(f(x_t))}{dx_t} = [\frac{df}{dx_t}|_{x_t=f(q)}][\frac{df}{dx_t}|_{x_t=q}].$$

Thus, the stability of a fixed point of period 2 depends on the slope of the function $f(x_t)$ at both of the two points q and $f(q)$.

Let's do a numerical experiment to investigate the properties of the mappings $x_{t+1} = x_t(4-4x_t)$ that is quite similar to the quadratic map that we have studied

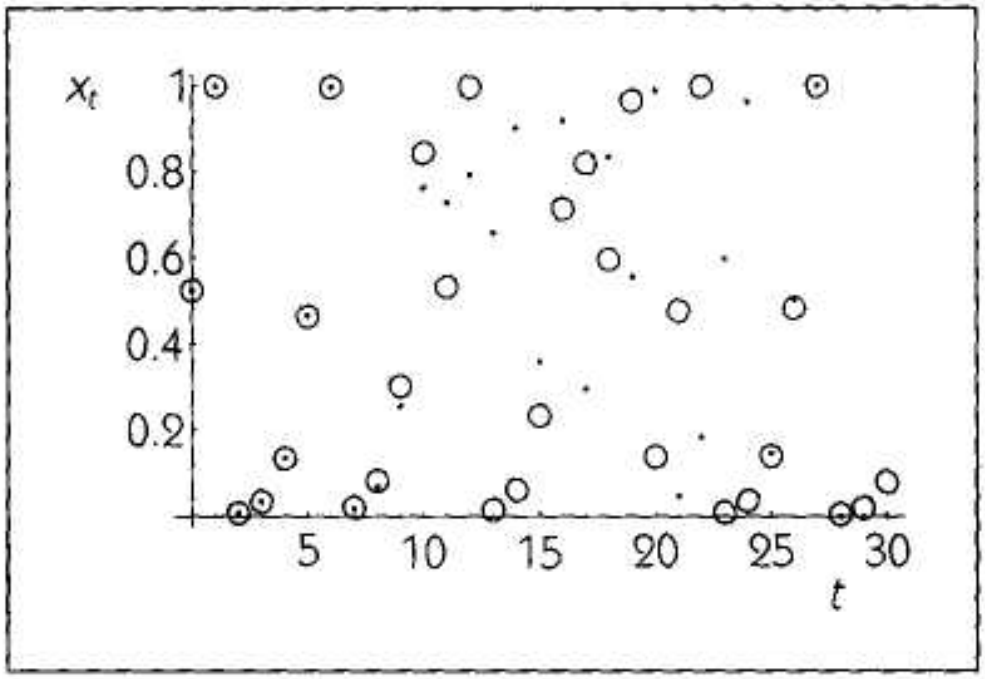
earlier. Pick an initial condition, say $x_0 = 0.523423$, and iterate. Now start over, but change the initial condition by just a little bit, to $x_0 = 0.523424$. The results are shown in the Figure below (Reference: Leon Glass and Kaplan.)

There are several important features of the dynamics illustrated in the following Figure. In fact, based on the figure we have strong evidence that this equation displays chaos—which is defined to be *aperiodic bounded dynamics in a deterministic system with sensitive dependence on initial conditions*.

Each of these terms has a specific meaning. We define the terms and explain why each of these properties appears to be satisfied by the dynamics in Figure below.

Aperiodic means that the same state is never repeated twice. Examination of the numerical values used in this graph shows this to be the case. However, in practice, by either graphically iterating or using a computer with finite precision, we eventually may return to the same value. Although a computer simulation or graphical iteration always leaves some doubt about whether behavior is periodic, the presence of very long cycles or of *aperiodic* dynamics in computer simulations is partial evidence for chaos.

Bounded means that on successive iterations the state stays in a finite range and does not approach $\pm\infty$. In the present case, as long as the initial condition x_0 is in the range $0 < x_0 < 1$, then all future iterates will also fall in this range. This is because for $0 < x_t < 1$, the minimum value of $4(1 - x_t)x_t$ is 0 and the maximum value is 1. Recall that in the linear



Below, we quote the MATLAB code for several famous maps.

```
function [x,y]=henon(n,level,a,b,x0,y0)
%Syntax: [x,y]=henon(n,level,a,b,x0,y0)
%_____________________________________
%
% Simulation of the Henon map.
%    x'=1-a*x^2+y
%    y'=b*x
%
% x and y are the simulated time series.
```

```
% n is the number of the simulated points.
% level is the noise standard deviation divided by the standard
deviation of the
%   noise-free time series. We assume Gaussian noise with zero mean.
% a is the a parameter.
% b is the b parameter.
% x0 is the initial value for x.
% y0 is the initial value for y.
%
%
% Reference:
%
% Henon M (1976):A two-dimensional mapping with a strange attractor.
% Communications in Mathematical Physics 50: 69-77
%
%
% author: Alexandros Leontitsis


if nargin<1 | isempty(n)==1
    n=500;
else
    % n must be scalar
    if sum(size(n))>2
        error('n must be scalar.');
    end
    % n must be positive
    if n<0
        error('n must be positive.');
    end
    % n must be an integer
    if round(n)-n~=0
        error('n must be an integer.');
    end
end

if nargin<2 | isempty(level)==1
    level=0;
else
    % level must be scalar
    if sum(size(level))>2
        error('level must be scalar.');
    end
    % level must be positive
    if level<0
        error('level must be positive.');
    end
end

if nargin<3 | isempty(a)==1
    a=1.4;
else
    % a must be scalar
    if sum(size(a))>2
        error('a must be scalar.');
```

```
    end
end

if nargin<4 | isempty(b)==1
    b=0.3;
else
    % b must be scalar
    if sum(size(b))>2
        error('s must be a scalar.');
    end
end

if nargin<5 | isempty(x0)==1
    x0=0.1;
else
    % x0 must be scalar
    if sum(size(x0))>2
        error('x0 must be scalar.');
    end
end

if nargin<6 | isempty(y0)==1
    y0=0.1;
else
    % y0 must be scalar
    if sum(size(y0))>2
        error('y0 must be scalar.');
    end
end

% Initialize
x(1,1)=1-a*x0^2+b*y0;
y(1,1)=b*x0;

% Simulate
for i=2:n
    x(i,1)=1-a*x(i-1,1)^2+y(i-1,1);
    y(i,1)=b*x(i-1,1);
end

% Add normal white noise
x=x+randn(n,1)*level*std(x);
y=y+randn(n,1)*level*std(y);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=mackeyglass(n,level,a,b,c,x0)
%Syntax: x=mackeyglass(n,level,a,b,c,x0)
%_____________________________________
%
% Simulation of the discretized variant of the Mackey-Glass PDE.
%    x(i+1)=x(i)+ax(i-s)/(1+x(i-s)^c)-bx(i)
%
% x is the simulated time series.
% n is the number of the simulated points.
```

```
% level is the noise standard deviation divided by the standard
deviation of the
%   noise-free time series. We assume Gaussian noise with zero mean.
% a, b, c, and s are the parameter
% x0 is the initial values vector for x.
%
% Note:
% s=length(x0)
%
%
% Reference:
%
% Mackey M C, Glass L (1977): Oscillation and Chaos in Physiological
% Control Systems. Science 177: 287-289
%
%
% code author Alexandros Leontitsis

if nargin<1 | isempty(n)==1
    n=500;
else
    % n must be scalar
    if sum(size(n))>2
        error('n must be scalar.');
    end
    % n must be positive
    if n<0
        error('n must be positive.');
    end
    % n must be an integer
    if round(n)-n~=0
        error('n must be an integer.');
    end
end

if nargin<2 | isempty(level)==1
    level=0;
else
    % level must be scalar
    if sum(size(level))>2
        error('level must be scalar.');
    end
    % level must be positive
    if level<0
        error('level must be positive.');
    end
end

if nargin<3 | isempty(a)==1
    a=0.2;
else
    % a must be scalar
    if sum(size(a))>2
        error('a must be scalar.');
    end
```

```
end

if nargin<4 | isempty(b)==1
    b=0.1;
else
    % b must be scalar
    if sum(size(b))>2
        error('b must be scalar.');
    end
end

if nargin<5 | isempty(c)==1
    c=10;
else
    % c must be scalar
    if sum(size(c))>2
        error('c must be scalar.');
    end
end

if nargin<6 | isempty(x0)==1
    x0=0.1*ones(17,1);
else
    % x0 must be either a scalar or a vector
    if max(size(x0))>2
        error('x0 must be either a scalar or a vector.');
    end
end

s=length(x0);
% n must be greater than or equal to s=length(x0)
if n<s
    error('n must be greater than or equal to s=length(x0).');
end


% Initialize
x(1,1)=x0(s)+a*x0(1)/(1+x0(1)^c)-b*x0(s);
for i=2:length(x0)
    x(i,1)=x(i-1)+a*x0(i)/(1+x0(i)^c)-b*x(i-1);
end

% Simulate
for i=s+1:n
    x(i,1)=x(i-1)+a*x(i-s)/(1+x(i-s)^c)-b*x(i-1);
end

% Add normal white noise
x=x+randn(n,1)*level*std(x);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,y]=ikeda(n,level,mu,x0,y0)
%Syntax: [x,y]=ikeda(n,level,mu,x0,y0)
%_____________________________________
```

```
%
% Simulation of the Ikeda map.
%    x'=1+mu(xcos(t)-ysin(t)
%    y'=mu(xsin(t)+ycos(t))
%
% x and y are the simulated time series.
% n is the number of the simulated points.
% level is the noise standard deviation divided by the standard
deviation of
%   the noise-free time series. We assume Gaussian noise with zero
mean.
% mu is the parameter.
% x0 is the initial value for x.
% y0 is the initial value for y.
%
%
% Reference:
%
% Ikeda K (1979): Multiple-valued stationary state and its
instability of the
% transmitted light by a ring cavity system. Optics Communications
30: 257
%
%
% author: Alexandros Leontitsis

if nargin<1 | isempty(n)==1
    n=500;
else
    % n must be scalar
    if sum(size(n))>2
        error('n must be scalar.');
    end
    % n must be positive
    if n<0
        error('n must be positive.');
    end
    % n must be an integer
    if round(n)-n~=0
        error('n must be an integer.');
    end
end

if nargin<2 | isempty(level)==1
    level=0;
else
    % level must be scalar
    if sum(size(level))>2
        error('level must be scalar.');
    end
    % level must be positive
    if level<0
        error('level must be positive.');
    end
end
```

```
if nargin<3 | isempty(a)==1
    mu=0.9;
else
    % mu must be scalar
    if sum(size(mu))>2
        error('a must be scalar.');
    end
end

if nargin<4 | isempty(x0)==1
    x0=0.1;
else
    % x0 must be scalar
    if sum(size(x0))>2
        error('x0 must be scalar.');
    end
end

if nargin<5 | isempty(y0)==1
    y0=0.1;
else
    % y0 must be scalar
    if sum(size(y0))>2
        error('y0 must be scalar.');
    end
end


% Initialize
t=0.4-6/(1+x0^2+y0^2);
x(1,1)=1+mu*(x0*cos(t)-y0*sin(t));
y(1,1)=mu*(x0*sin(t)+y0*cos(t));

% Simulate
for i=2:n
    t=0.4-6/(1+x(i-1,1)^2+y(i-1,1)^2);
    x(i,1)=1+mu*(x(i-1,1)*cos(t)-y(i-1,1)*sin(t));
    y(i,1)=mu*(x(i-1,1)*sin(t)+y(i-1,1)*cos(t));
end

% Add normal white noise
x=x+randn(n,1)*level*std(x);
y=y+randn(n,1)*level*std(y);



function x=logistic(n,level,a,x0)
%Syntax: x=logistic(n,level,a,x0)
%_________________________________
%
% Simulation of the Quadratic map.
%    x'=ax(1-x)
%
% x is the simulated time series.
```

```
% n is the number of the simulated points.
% level is the noise standard deviation divided by the standard
deviation of the
%   noise-free time series. We assume Gaussian noise with zero mean.
% a is the parameter.
% x0 is the initial value for x.
%
%
% Reference:
%
% May R M (1976): Simple mathematical models with
% very complicated dynamics. Nature 261: 459-467
% code author: Alexandros Leontitsis

if nargin<1 | isempty(n)==1
    n=500;
else
    % n must be scalar
    if sum(size(n))>2
        error('n must be scalar.');
    end
    % n must be positive
    if n<0
        error('n must be positive.');
    end
    % n must be an integer
    if round(n)-n~=0
        error('n must be an integer.');
    end
end

if nargin<2 | isempty(level)==1
    level=0;
else
    % level must be a scalar
    if sum(size(level))>2
        error('level must be scalar.');
    end
    % level must be positive
    if level<0
        error('level must be positive.');
    end
end

if nargin<3 | isempty(a)==1
    a=4;
else
    % a must be scalar
    if sum(size(a))>2
        error('a must be scalar.');
    end
end

if nargin<4 | isempty(x0)==1
    x0=0.1;
```

```
else
    % x0 must be scalar
    if sum(size(x0))>2
        error('x0 must be scalar.');
    end
end


% Initialize
x(1,1)=a*x0*(1-x0);

% Simulate
for i=2:n
    x(i,1)=a*x(i-1,1)*(1-x(i-1,1));
end

% Add normal white noise
x=x+randn(n,1)*level*std(x);



function [x,y,z]=rossler(n,level,a,b,c,x0,y0,z0,h)
%Syntax: [x,y,z]=rossler(n,level,a,b,c,x0,y0,z0,h)
%_____________________________________________
%
% Simulation of the ODE.
%    dx/dt=-y-z
%    dy/dt=x+ay
%    dz/dt=b+z(x-c)
%
% x, y, and z are the simulated time series.
% n is the number of the simulated points.
% level is the noise standard deviation divided by the standard
deviation of the
%   noise-free time series. We assume Gaussian noise with zero mean.
% a, b, and c are the parameters
% x0 is the initial value for x.
% y0 is the initial value for y.
% z0 is the initial value for z.
% h is the step size.
%
%
% Notes:
% The time is n*h.
% The integration is obtained by the Euler's method.
%
%
% Reference:
%
% Rossler O E (1976): An equation for continuous chaos.
% Physics Letters A 57: 397-398
%
% code author: Alexandros Leontitsis

if nargin<1 | isempty(n)==1
```

```
    n=500;
else
    % n must be scalar
    if sum(size(n))>2
        error('n must be scalar.');
    end
    % n must be positive
    if n<0
        error('n must be positive.');
    end
    % n must be an integer
    if round(n)-n~=0
        error('n must be an integer.');
    end
end

if nargin<2 | isempty(level)==1
    level=0;
else
    % level must be scalar
    if sum(size(level))>2
        error('level must be scalar.');
    end
    % level must be positive
    if level<0
        error('level must be positive.');
    end
end

if nargin<3 | isempty(a)==1
    a=0.2;
else
    % a must be scalar
    if sum(size(a))>2
        error('a must be scalar.');
    end
end

if nargin<4 | isempty(b)==1
    b=0.4;
else
    % b must be scalar
    if sum(size(b))>2
        error('b must be scalar.');
    end
end

if nargin<5 | isempty(c)==1
    c=5.7;
else
    % c must be scalar
    if sum(size(c))>2
        error('c must be scalar.');
    end
end
```

```
if nargin<6 | isempty(x0)==1
    x0=0.1;
else
    % x0 must be scalar
    if sum(size(x0))>2
        error('x0 must be scalar.');
    end
end

if nargin<7 | isempty(x0)==1
    y0=0.1;
else
    % y0 must be scalar
    if max(size(y0))>2
        error('y0 must be scalar.');
    end
end

if nargin<8 | isempty(z0)==1
    z0=0.1;
else
    % z0 must be scalar
    if max(size(z0))>2
        error('z0 must be scalar.');
    end
end

if nargin<9 | isempty(h)==1
   h=0.1;
else
    % h must be scalar
    if max(size(h))>2
        error('h must be scalar.');
    end
    % h must be positive
    if h<0
        error('h must be positive.');
    end
end

% Initialize
y(1,:)=[x0 y0 z0];

% Simulate
for i=2:n
    ydot(1)=-y(i-1,2)-y(i-1,3);
    ydot(2)=y(i-1,1)+a*y(i-1,2);
    ydot(3)=b+y(i-1,3)*(y(i-1,1)-c);
    y(i,:)=y(i-1,:)+h*ydot;
end

% Separate the solutions
x=y(:,1);
z=y(:,3);
```

```
y=y(:,2);

% Add normal white noise
x=x+randn(n,1)*level*std(x);
y=y+randn(n,1)*level*std(y);
z=z+randn(n,1)*level*std(z);


function [Y,T]=phasespace(x,dim,tau)
%Syntax: [Y,T]=phasespace(x,dim,tau)
%_____________________________________
%
% The phase space reconstruction of a time series x whith the Method
Of Delays
% (MOD), in embedding dimension m and for time dalay tau.
%
% Y is the trajectory matrix in the reconstructed phase space.
% T is the phase space length.
% x is the time series.
% dim is the embedding dimension.
% tau is the time delay.
%
%
% Reference: Takens F (1981): Detecting strange attractors
% in turbulence. Lecture notes in Mathematics, 898. Springer.
%
% code author: Alexandros Leontitsis

if nargin<1 | isempty(x)==1
   error('You should provide a time series.');
else
   % x must be a vector
   if min(size(x))>1
      error('Invalid time series.');
   end
   x=x(:);
   % N is the time series length
   N=length(x);
end

if nargin<2 | isempty(dim)==1
    dim=2;
else
    % dim must be scalar
    if sum(size(dim))>2
        error('dim must be scalar.');
    end
    % dim must be an integer
    if dim-round(dim)~=0
        error('dim must be an integer.');
    end
    % dim must be positive
    if dim<=0
        error('dim must be positive.');
    end
```

```
end

if nargin<3 | isempty(tau)==1
    tau=1;
else
    % tau must be scalar
    if sum(size(tau))>2
        error('tau must be scalar.');
    end
    % tau must be an integer
    if tau-round(tau)~=0
        error('tau must be an integer.');
    end
    % tau must be positive
    if tau<=0
        error('tau must be positive.');
    end
end

% Total points on phase space
T=N-(dim-1)*tau;

% Initialize the phase space
Y=zeros(T,dim);

% Phase space reconstruction with MOD
for i=1:T
   Y(i,:)=x(i+(0:dim-1)*tau)';
end
```