

Development of Algorithm for Lorenz Equation using Different Open Source Softwares

Paul R^{1,2} and Majumder D^{1,2*}

¹Department of Physiology, West Bengal State University, Berunanpukuria, Malikapur Barasat, Kolkata, West Bengal, India

²Society for Systems Biology & Translational Research; 103, Block - C, Bangur Avenue, Kolkata - 700055, West Bengal, India

Abstract

Physiological processes are dynamic in nature and have multi-factorial influences; hence, exhibited as nonlinear and complex. Due to unavailability of data capturing technology in discrete time points, majority of physiological researches are focused on linearity; and hence problems of complexity are addressed empirically. However, in recent time there is an increasing trend to understand the physiological system in a quantitative manner across the globe. Due to unavailability of costly software, it is difficult for students to get an exposure to this global trend. In physical system complexity was first addressed by Edward Lorenz in 1963, which is now known as Lorenz equation. Here we depict the simple computational approach to represent the Lorenz equation using some freely available open source softwares, so that students by themselves can appreciate the importance of the quantitative approach of science in and able to represent the complex and nonlinear dynamical behaviour of different physiological systems.

Keywords: Mathematical model; Algorithm; Complexity; Quantitative science

Introduction

Complexity in physical system was first addressed by Edward Lorenz [1] which is now known as Lorenz equations. For this he developed a simplified mathematical model of Ordinary Differential Equation (ODE) of atmospheric convection. These coupled equations have chaotic solutions for certain parameter values and initial conditions. In particular, the Lorenz attractor is a set of chaotic solutions of the Lorenz system which, when plotted, resemble a butterfly or figure eight. Lorenz equations are follows:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dx}{dt} = X(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

In the above equations, σ , ρ , β are the system parameters; x , y , z are the system state and t is time.

One can easily assume that σ , ρ and β are positive. Lorenz used the values $\sigma=10$, $\rho=28$, $\beta=8/3$. The system exhibits chaotic behaviour for these values [2]. Looking for chaos in cardiac rhythm, brain and population dynamics now has a greater merit in comparison to linear statistical methods. Hence, modeling and data analysis are considered to motivate the investigation in an understanding the behaviour or nature of the biological systems in a dynamical manner, particular their chaotic features, if any. Hence such modeling approach followed by data fitting may be useful in diagnosis and prognosis (predictions about the efficacy of a therapeutic process [3]. From a technical standpoint, the Lorenz system is nonlinear, three-dimensional and deterministic [4].

Development of Algorithm for Solving Differential Equation

If the system of differential equation is a linear there are systemic methods for deriving a solution. Most of the problems biologist

encounters are non-linear and for such cases mathematical solutions rarely exist. Hence, computer simulation is often used instead. The general approach to obtaining a solution by computer is as follows:

1. Construct the set of ordinary differential equations, with one differential equation for every molecular species in the model.
2. Assign values to all the various kinetic constants and boundary species.
3. Initialize all floating molecular species to their starting concentrations.
4. Apply an integration algorithm to the set of differential equations.
5. If required, compute the fluxes from the computed species concentrations.
6. Plot the results of the solution.

Step 4 is obviously the key to the procedure and there exist a great variety of integration algorithms. Other than educational purposes, it is rarely a modeller can write their own integration computer code because many libraries and applications exist that incorporate excellent integration methods. An integration algorithm approximates the behaviour of a continuous system on a digital computer. Since digital computers can only operate in discrete time system, the algorithms convert the continuous system into a discrete time system. In practice, a particular step size, h is chosen, and solution points are at the discrete points up to some upper time limit. The approximation generated

***Corresponding author:** Majumder D, Department of Physiology, West Bengal State University, Berunanpukuria, Malikapur Barasat, Kolkata, West Bengal, India, Tel: +91-33-25241977; E-mail: durjoy@rocketmail.com

Received September 02, 2017; **Accepted** September 11, 2017; **Published** September 13, 2017

Citation: Paul R, Majumder D (2017) Development of Algorithm for Lorenz Equation using Different Open Source Softwares. J Comput Sci Syst Biol 10: 087-092. doi:10.4172/jcsb.1000255

Copyright: © 2017 Paul R, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

by the simplest methods is dependent on the step size and in general smaller the step size more accurate the solution. It is not possible to continually reduce the step size in the hope of increasing the accuracy of the solution. For one thing, the algorithm will soon reach the limits of the precision of the computer and secondly, smaller the step size the longer it will take to compute the solution. There is therefore often a trade-off between accuracy and computation time [5].

There are several methods to develop an algorithm for increasing accuracy of a solution. We often write models of biochemical reactions in the form of ordinary differential equations. These equations describe the instantaneous rate of change of each species in the model. For example, consider the simple possible model, the first order irreversible degradation of molecular species, S into product P:



The differential equation for this simple reaction is given by a familiar form:

$$\frac{ds}{dt} = -k_1 S \quad (2)$$

To solve this equation several computational methods can be used so that the pattern of changes of S in time can be derived. Let us first consider the simplest method, Euler method which depicts an easiest way to solve a differential equation. This method uses the rate of change of S to predict the concentration at some future point in time. At time t_1 the rate of change in S is computed from the differential equation using the known concentration of S at t_1 . The rate of change in S to predict the over a time interval h , using the relation $\frac{ds}{dt}$. The current time, is incremented by the time step, h and the procedure repeated again, this time starting at t_2 . This method can be summarized by the following two equations which represent one step in an iteration that repeats until the final point is reached:

$$y(t+h) = y(t) + h \frac{dy(t)}{dt}$$

$$t_{n+1} = t_n + h \quad (3)$$

At every iteration, there will be an error between the change in S we predict and what the change in S should have been. This error is called the truncation error and will accumulate at each iteration step. If the step size is too large, this error can make the method numerically unstable resulting in wild swings in the solution. Figure 1 suggests that larger the step size larger the truncation error. This

would seem to suggest that the smaller the step size more accurate the solution will be. This is indeed the case, up to a point. If the step size becomes too small, then there is the risk that round off error will propagate at each step into the solution. In addition, if the step size is too small it will require a large number of iterations to simulate even a small time period. The final choice for the step size is therefore a compromise between accuracy and effort. A theoretical analysis of error propagation in the Euler method indicates the error accumulated over the entire integration period (global error) is proportional to the step size. Therefore, halving the step size will reduce the global error by half. This means that to achieve even modest accuracy, small step sizes are necessary. As a result, the method is rarely used in practice. The advantage of the Euler method is that it is very easy to implement in computer code or even on a spread sheet.

```

n = Number of state variables
y_i = i-th variable
Set timeEnd
currentTime = 0
h = stepSize
initialize y_i at current time
while currentTime < timeEnd do
  for i = 1 to n do
    dy_i = f_i(y)
  for i = 1 to n do
    y_i(t+h) = y_i(t) + h dy_i
  currentTime = currentTime + h
end while
    
```

Algorithm 1: Euler integration method, $f_i(y)$ represents differential equation from the systems of ordinary differential equations.

Euler method can also be used to solve systems of differential equations. In this case all the rates of change are computed first followed by the application of the equation (3). As in all numerical integration methods, the computation must start with an initial for the state variables at time zero. The algorithm is described using pseudo-code in Algorithm 1.

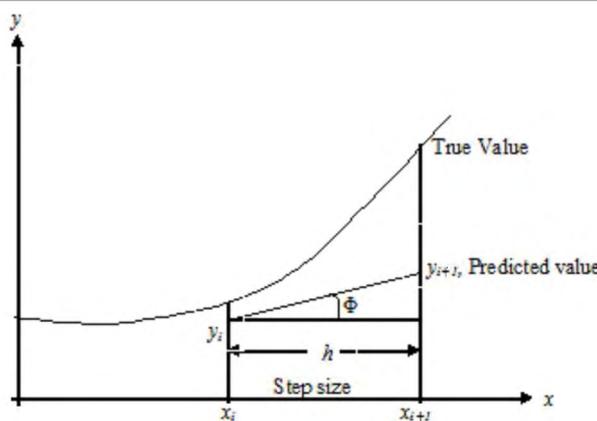


Figure 1: Graphical representation of Euler's method.

Euler method, though simple to implement, tends to be used in practice because it requires small step sizes to achieve reasonable accuracy. In addition, the small step size makes the Euler Method computationally slow. An example of Euler method is shown in Algorithm 2, by which Prey-Predator Population Dynamics can be observed. Outputs are shown Figure 2.

```
%forward Euler
K(1)=5.0; %initial Krill population
a=1.0; %(a-bW) is the intrinsic growth rate of the Krill
b=0.5;
W(1)=1.0;%initial whale population
c=0.75; %(-c+dK) is the intrinsic growth rate of whales
d=0.25;
tinit=0.0;
tfinal=50;
n=5000;%no. of time steps
dt=(tfinal-tinit)/n;%time step size
%T=[tinit:dt:tfinal]; %create vector of discrete solution times
%Execute forward Euler to solve at each time step
for i=1:n
K(i+1)=K(i)+dt*K(i)*(a-b*W(i));
W(i+1)=W(i)+dt*W(i)*(-c+d*K(i));
end;
%Plot Results...
%S1=sprintf('IC: W0=%g, K0=%g',W(1),K(1));
figure(1);
plot(K,W);
title('Phase Plane Plot');
xlabel('Krill');
ylabel('Whales');
%legend(S1,0)
%grid;
init=1;
figure(2);
%clg;
plot((init:tfinal),K((init:tfinal)), 'r', (init:tfinal),W((init:tfinal)), 'b-');
legend('Krill','Whales');
xlabel('time');
ylabel('whales and krill');
```

Algorithm 2: Application of Forward Euler method in MatLab/FreeMat.

Modification of Euler method-Heun method

A simple modification however can be made to the Euler Method to significantly improve its performance. This approach can be found under a number of headings, including the modified Euler Method or Heun or the improved Euler Method. The modification involves improving the estimate of the slope by averaging two derivatives, one at the initial point and another at the end point. In order to calculate the derivative at the end point, the first derivative must be used to predict

the end point which is then corrected by averaged slope. This method is very simple example of predictor-corrector method. This method can be summarized by the following equations:

$$y(t+h) = y(t) + h \frac{dy(t)}{dt} \quad (4)$$

$$y(t+h) = y(t) + \frac{h}{2} \left(\frac{dy(t)}{dt} + \frac{dy(t+h)}{dt} \right) \quad (5)$$

$$t_{n+h} = t_n + h \quad (6)$$

A theoretical analysis of error in the propagation of Heun method show that it is a second order method, that is if the step size is reduced by a factor of 2, the global error reduced by factor of 4. However, to achieve this improvement, two evaluations of the derivatives are required per iteration, compared to only one for Euler method. Like the Euler method Heun method is also easy to implement.

```
n = Number of state variables
y= ith variable
Set timeEnd
currentTime = 0
h = stepSize
initialize yi at current time
while currentTime < timeEnd do
for i = 1 to n do
ai = (y)
bi = fi (y + h a)
for i = 1 to n do
yi(t+h) = yi(t) +  $\frac{h}{2}$ (ai + bi)
currentTime = currentTime + h
end while
```

Algorithm 3: Heun Integration Method. $f_i(y)$ is the i^{th} ordinary differential equation.

The Runge-Kutta methods

The Heun method described in the previous section is sometimes called RK2 method where RK2 stands for second order Runge-Kutta method. The Runge-Kutta methods are a family of methods developed around the German mathematicians Runge and Kutta. In addition to the 2nd order Heun method, there have 3rd, 4th, and even 5th order Runge-Kutta methods. For hand coded numerical methods, the 4th order Runge-Kutta algorithm (often called RK4) is the most popular among modelers. The algorithm often is a little more complicated in that it involves the evaluation and weighted averaging four slopes. In terms of global error, however, RK4 is considerably better than Euler or Heun method and has a global error of the order of four. This means that halving the step size will reduce the global error by a factor of 1/16. Another way of looking at this is that the step size can be increased up to 16 fold over the Euler method and still have the same global error. This method can be summarized by the following equations which have been simplified by removing the dependence on time:

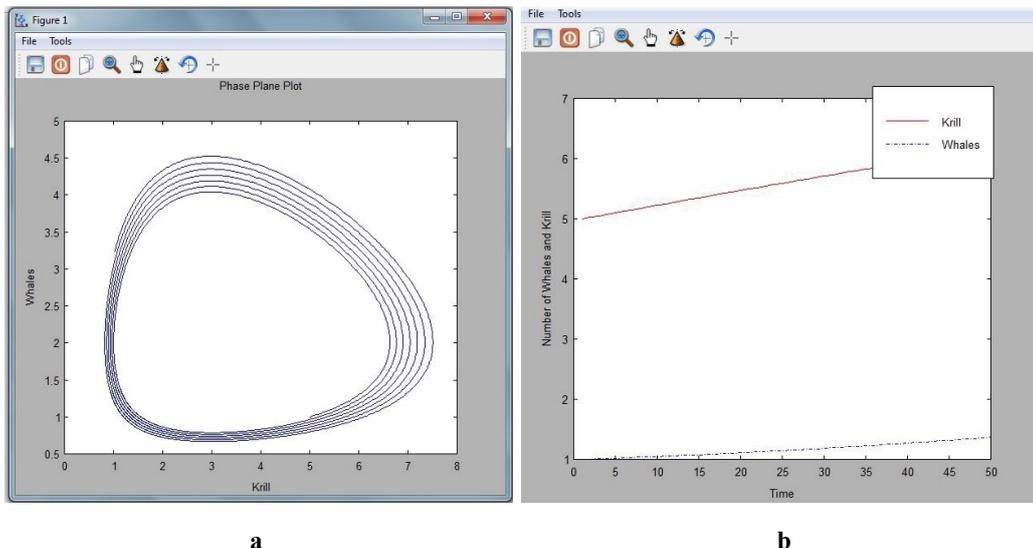


Figure 2: Prey-predator dynamics by Forward Euler Algorithm, representation through FreeMat. In (a) Phase Plane plot and in (b) dynamics of two species.

$$K_1 = hf(y_n)$$

$$K_2 = hf(y_n + \frac{k_1}{2})$$

$$K_3 = hf(y_n + \frac{k_2}{2})$$

$$K_4 = hf(y_n + k_3)$$

$$t_{n+1} = t_n + h$$

Materials and Methods

There are different software's to solve systems of differential equation. For this purpose, MatLab is the most popular software tool among engineers is available commercially. It has a built in powerful language for numerical analysis. Other examples include Octave, FreeMat and Scilab – all are freely available open source software that can be used to solve different ordinary differential equations. We have developed our code in MatLab, FreeMat and Octave and a comparison is made to understand how an algorithm for 4th and 5th order Runge – Kutta method is used through open source software for representing Lorenz equation (Table 1).

Results

Simulation run with the algorithm for Lorenz equation gives following graphical plots in MatLab (Figure 3), FreeMat (Figure 4) and Octave (Figure 5) software respectively.

Discussion

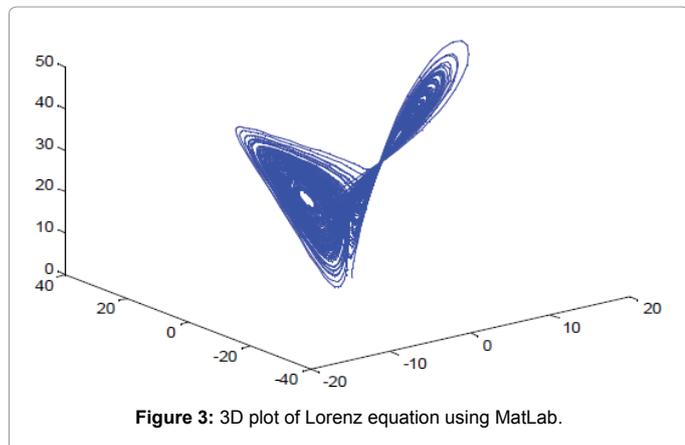
Conventionally conclusions and inferences about physiological systems are based on empirical assessment by making comparison between controls versus experimental or normal versus disease group and many of the physiological principles are set with the experimentation and forceful perturbation of the isolated system and, different conclusions and principles are set with one-shot static data. As a result, observations in the isolated systems are extrapolated towards the behavior of the natural physiological system, thus establish

MatLab	FreeMat	Octave
clear all	clear all	clear;
clc	clc	clc;
sigma=10;	sigma=10;	sigma=10;
beta=8/3;	beta=8/3;	function wdot =
rho=28;	rho=28;	f(w,t)
f = @(t,w) [-sigma*w(1) +	f = @(t,w) [-sigma*w(1) +	wdot(1)=10*(w(2)-
sigma*w(2); rho*w(1) - w(2)	sigma*w(2); rho*w(1) - w(2)	w(1));
- w(1)*w(3); -beta*w(3) +	- w(1)*w(3); -beta*w(3) +	wdot(2)=-
w(1)*w(2);	w(1)*w(2);	w(1)*w(3)+28*w(1)-
%f is the set of	%f is the set of	w(2);
differential equations and	differential equations and	wdot(3)=w(1)*w(2)-
'w' is an array containing	'w' is an array containing	8*w(3)/3;
values of x,y, and z	values of x,y, and z	endfunction
variables.	variables.	t =
%t is the time variable	%t is the time variable	linspace(0,10,100)'
[t,w] = ode45(f,[0 100],[1	[t,w] = ode45(f,[0 100],[1	: % 0 50 100
1 1]);%'ode45' uses	1 1]);%'ode45' uses	w = Isode("f",[2;
adaptive Runge-Kutte	adaptive Runge-Kutte	3; 9],t); %5 7 9
method	method	figure(1);
of 4th and 5th order to	of 4th and 5th order to	plot3(w(:,1),w(:,2)
solve differential	solve differential	plot3(w(:,1),w(:,2)
equations	equations	,w(:,3));
plot3(w(:,1),w(:,2),w(:,3))	plot3(w(:,1),w(:,2),w(:,3))	hold
%'plot3' is the command to	%'plot3' is the command to	figure(2);
make 3D plot	make 3D plot	plot(w(:,1));

Table 1: Code for Lorenz equation in MatLab, FreeMat and Octave.

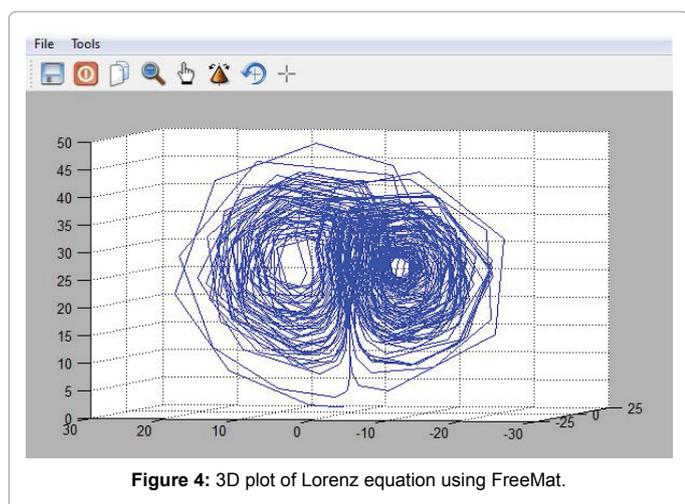
different physiological principles. Chaos theory established that all natural system is the output of multiple interactions of different components along with different feedbacks and delays; hence has an extraordinary sensitive to internal conditions which makes them inherently unpredictable in the long run. Lorenz showed that a change

in a digit in the 6th decimal make a drastic change in systems dynamics in long-run. Hence, for prediction of physiological systems, accurate measurements of initial parametric values are important, however, sensing and measurement technology becomes the limitation. This may be true for physiological system, a natural system.



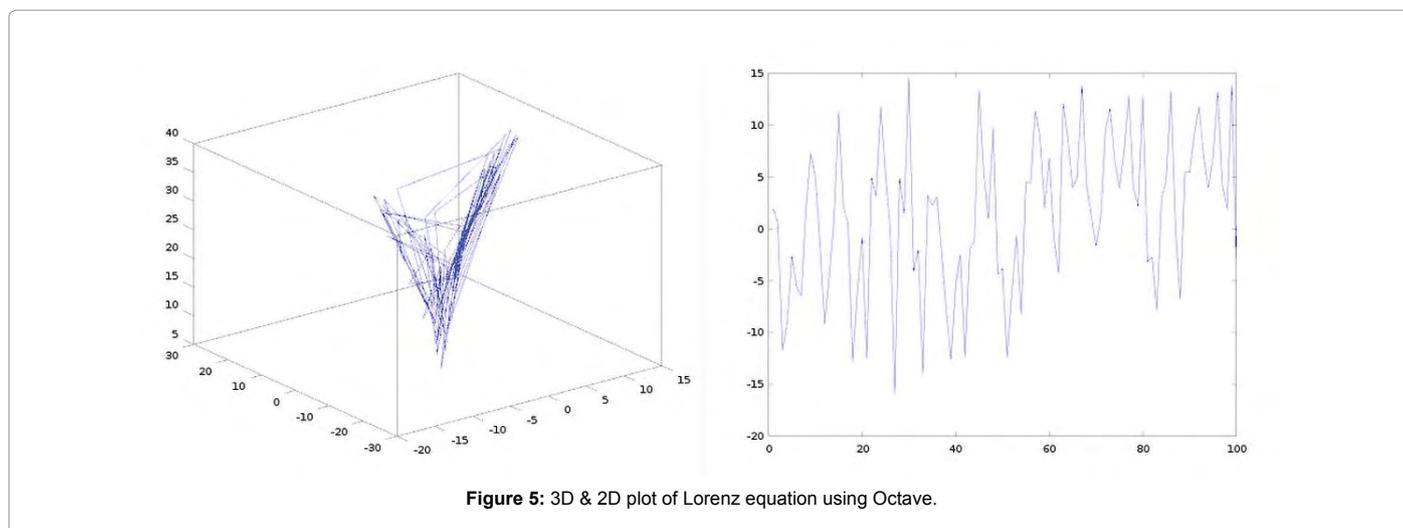
Another characteristic of chaotic systems is order without periodicity. Though chaotic systems operates under some set rule, but such behaviour may occur due to feedbacks and time time-delay associated with a process makes it unpredictable. Due to these two features behaviour of the physiological systems appears to be complex. Chaotic system also shows order and chaos depending on the situation. When the system becomes increasingly unstable, an attractor draws the stress and the system splits and return to order. This process is called bifurcation. Bifurcation results in new possibilities that keep the system alive and random.

We mention some important features of heart beating and brain function with respect to chaos. For details interested readers may consult paper “Human beings as chaotic systems” by Crystal Ives [6]. Though apparently it seems that heart beats periodically in resting condition but sensitive instrument reveal that there is small variability in the interval between beats. Such results due to delay in transmission of signal from SA node to other parts of heart and respiratory system also influences its activity.



Similarly, neuron doctrine states that the physiological basis of behavior depends on the activity of individual neurons which is triggered by stimulus. So far brain activity is explained as a local network as the “chemical point-to-point switch board”. Chaos theory strongly opposes neuron doctrine and urges for holistic analysis for the brain functioning. Small changes in a neuronal activity make a bifurcation hence employ of newer nerve cells; hence there is a large deviation in brain activity with the progress of time - this may be exhibited with the chaotic attractor. “A theory exists that learning takes place when a new stimulus leads to the emergence of an unpatterned, increasingly chaotic state in the brain”.

Another important aspect is in the definition of disease state. In the field of pathology there is a concept that disorder caused disease. But now health is viewed as chaos. Arnold Mandell, a Psychiatrist



and dynamicists wonders, “Is it possible that mathematical pathology, chaos, is health? And the mathematical health, which is... predictability and differentiability... is disease?” [7]. A nonlinear system has the adaptability that helps to adjust possesses the characteristic in the system disorder. “From seizures to leukemia, disease is finally being recognized for what it is: an acute attack of order.” So, studying the dynamics becomes important and to address of human problem, confinement within the subject nomenclature becomes irrelevant.

However, experimental biologists and physiologists consider the associated conceptual changes as an addition to the existing knowledge, and hence unable to appreciate the newer dimension. Paul Rapp, a neuroscientist at Medical College of Pennsylvania comment on the plotting of EEG waves on phase-space diagram can be noted: “For the first time we are able to see changes in the geometry of EEG activity that occur as the result of human cognitive activity... I expected to see something very boring that did not significantly change as the subject began to think. The moment these structures flooded onto the screen and began to rotate, I knew I was seeing something very extraordinary”.

Conclusion

Codes for Lorenz equation was developed in MatLab, FreeMat (Open source software), Octave (Open source software) for simulation to appreciate the complexity of a dynamical system. The simulation plots suggest that a nonlinear system and simulation study reveals

that dynamical pattern of a complex system is dependent on the initial parametric values of the systems variables. Hence, for getting a system prediction of a multifactorial complex system, accurate quantification of the parametric values of different variables is important. With the algorithm and code developed here would help physiologists to understand and appreciate the essence of measurement accuracy in different physiological experiments and the power of inferences through experiment [7,8].

References

1. Lorenz EN (1963) Deterministic non periodic flow. *Journal of the Atmospheric Sciences* 20: 130-141.
2. Hirsch MW, Smale S, Devaney RL (2003) *Differential Equations, Dynamical Systems and An Introduction to Chaos*. Chapter 14, 2nd Edn. Pure and Applied Mathematics Series, Elsevier Academic Press, 327-358.
3. Lesne A (2006) *Chaos in biology*. Chapter 18: Modeling Biological Systems 99: 413-428.
4. Sparrow C (1982) *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors*, Springer.
5. Systems Biology Organization, *Control Theory For Biologists*, draft 0.81.
6. Ives C (2016) *Student Papers*. Department of Physics, Oregon State University, USA.
7. Gleick J (1987) *Chaos: Making a new science*. Penguin Books, New York, NY, USA.
8. Briggs J (1992) *Fractals: the patterns of chaos*. Touchstone, Simon and Schuster Inc., New York, USA.