An Introduction to Mathematical Modeling in the Life Sciences

Final Project Report

A Chaotic Model for the Insulin-Glucose Regulatory System Reveals Key Features of Multiple Metabolic Disorders

Zhen-Yu Liu (刘振宇)   Yuanpei College   1700017853

2022.6.20

Abstract

The insulin-glucose regulatory system is one of the most studied homeostatic regulation systems in mammals. Mathematical modeling of the system could provide non-invasive methods for investigating the interaction of insulin and glucose, as well as helping the diagnosis of metabolic disorders related to this system. Here we implement a new model for this system based on the famous prey and predator models. Numerical simulation indicates that the model has various different dynamic behaviors under different conditions, including chaos. Using this model, we could mimic several common metabolic disorders related to the system and identity key parameters responsible for the initiation of these disease. We also explored the performance of this model on well-defined glucose tolerance tests, both intra-venous and oral. Our results might be helpful for better understanding of this regulatory system, as well as the mechanism and therapies for disease such as diabetes, hypoglycemia, and hyperinsulinemia.

Key words

Mathematic modeling; insulin-glucose regulatory system; chaos; diabetes.

1. Introduction

The insulin-glucose regulatory system offers one of the clearest and simplest examples of homeostatic control in the organism. As the main metabolic substrate for the organism, and the only energy source for brain tissue, glucose in the blood needs to be maintained within a narrow range (3.9-6.1 mmol/L). Abnormally low glucose level leads to anxiety, tremors, aggressiveness, obfuscation, coma and eventually death. On the other hands, excessive glucose concentrations cause microvascular damages (notably in the retina and kidney) and neural damages, leading to blindness and chronic renal insufficiency. Thus, limiting the plasma glucose concentrations in an appropriate range is crucial to the normal metabolism of the whole organism.

Physiological mechanisms of how the body controls plasma glucose level seems deceptively simple and has been well established. Essentially a single hormone (insulin), secreted by the β-cells in the pancreas, regulates glycemia. Insulin secretion is simulated by the increased blood glucose level and could subsequently promote peripheral tissue glucose uptake (mainly by the muscle and fat tissues) and decrease spontaneous glucose output from the liver. Apart from insulin, a group of others hormones, including the glucagon secreted by α-cell in the pancreas and adrenalin, contribute to rescuing the organism from hypoglycemia. However, since in clinical practice the situation of interest is normally inappropriately high glycemia, it's justified to concentrate more on the regulation by insulin.

Malfunction of the insulin-glucose system is associated with multiple metabolic disorders. Diabetes Mellitus (DM), also called diabetes, is one of the most common metabolic disorders in the human population. It was estimated that in 2015, about 415 million people, approximately 8.3% of the adult population of the world, suffer from diabetes (Yau et at, 2012) and the number is still increasing. Diabetes is classified into three types. In type 1 diabetes, the islet could not produce enough insulin, resulting in increased glucose level. For type 2 diabetes, insulin secretion isn't affected while the body is less sensitive to insulin, which also leads to increased glucose level. Type 3 diabetes a temporary situation that occurs during pregnancy. Besides diabetes, other disorders like hypoglycemia and hyperinsulinemia are also closely related to the glucose-insulin system. Understanding the dynamics of this system might help provides insights for the treatment of these metabolic disease.

Many mathematical models have been proposed to simulate the insulin-glucose system at different scale. Some models focus on insulin secretion of β-cells at cellular level, or insulin granule dynamics (Shibasaki et al, 2007). Some models aim to mimic the physiological oscillations of insulin and glucose. There are also many models designed to explain insulin-glucose dynamics after glucose tolerance test (GTT), where large dose of glucose is injected into blood (intra-venous glucose tolerance test, IVGTT) or absorbed through digestive system (oral glucose tolerance test). These GTT models are mostly top-down compartmental models, representing the observed features of the system without molecular mechanisms. However, they offered methods to estimate a set of key markers of T2DM development and are widely used in clinical diagnosis.

In our final project, we adopted a new chaotic model for the glucose-insulin regulatory system published on *Chaos, Solitons and Fractals* (Shabestari et al, 2018). This model could successfully describe the insulin and glucose oscillation under physiological conditions, it could also simulate some common metabolic disorders by optimizing specific model parameters. We implemented model in the paper and reproduced all the results provided by the author. Furthermore, we tested the predictive ability of the model on extreme non-physiological conditions, including IVGTT and OGTT. It turns out that the model had very poor performance on glucose tolerance test, probably because the large dose of glucose intake in GTT was too extremely excessive for the chaotic system.

2. Model presentation

2.1. Lotka–Volterra model for prey and predator

Alfred Lotka and Vito Volterra proposed the two-dimension ordinary differential equation model to describe the population dynamics of prey and predator in 1926, which was known as *Lotka–Volterra* model.

$$\frac{dx}{dt} = -ax + bxy$$

$$\frac{dy}{dt} = cy(1-y) - dxy \tag{1}$$

Where $x(t)$ is the population density of prey and $y(t)$ is the population density of predator. Model parameter a, b, c, d are all positive.

2.2. New model for insulin-glucose regulatory system.

Prompted by the prey and predator model, the relationship of glucose and insulin is also like prey and predator. It's reasonable to model the insulin-glucose system by making some adjustments to the *Lotka–Volterra* model. The author assumed that the derivatives of variables are cubic function of the variables themselves. The mathematical relationships for the model are formulated as follows:

$$\frac{dx}{dt} = -a_1x + a_2xy + a_3y^2 + a_4y^3 + a_5z + a_6z^2 + a_7z^3 + a_{20}$$

$$\frac{dy}{dt} = -a_8xy - a_9x^2 - a_{10}x^3 + a_{11}y(1-y) - a_{12}z - a_{13}z^2 - a_{14}z^3 + a_{21}$$

$$\frac{dz}{dt} = a_{15}y + a_{16}y^2 + a_{17}y^3 - a_{18}z - a_{19}yz \tag{2}$$

Where $x(t)$ is the concentration of insulin, $y(t)$ is the population of glucose and $z(t)$ is the population density of β-cells. There are 21 distinct model parameters, each has specific biological meaning: a_1 represents the natural reduction of insulin in the absence of glucose; a_2 shows the propagation rate of insulin in presence of glucose. a_8 represents the effect of insulin on glucose and a_{11} indicated the natural growth of glucose in the absence of insulin. These 4 parameters are determined through prey and predator model and should be positive. a_3 and a_4 show the increase rate of insulin when there is an increase in glucose concentration. a_5, a_6 and a_7 show the increase rate of insulin level secreted by β-cells and are independent from other components. a_9 and a_{10} represent the rate of glucose reduction in response to insulin secretion. a_{12}, a_{13} and a_{14} show the reduction rate of glucose concentration due to insulin secreted by β-cells. a_{15}, a_{16} and a_{17} represent the rate of increase in β-cells caused by the increase in glucose concentration. a_{18} and a_{19} show the rate of decrease in β-cells due to its current level.

The system has different behavior under different sets of parameters, Table 1 provided a set of parameters for normal condition and metabolic disorder, respectively. In this model, it is expected to

Table 1 Coefficients of proposed model.

Parameter	a_1 (normal)	a_1 (disorder)	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
Value	2.04	3	0.1	1.09	−1.08	0.03	−0.06	2.01	0.22	−3.84	−1.2
Parameter	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}
Value	0.3	1.37	−0.3	0.22	0.3	−1.35	0.5	−0.42	−0.15	−0.19	−0.56

observe periodic behavior under normal metabolic conditions and chaotic behavior under faulty status of metabolic system. This is consistent with previous studies revealing that a chaotic behavior of a system is a sign of an existing disorder in the system (Letellier et al, 2013).

3. Results

Part I Results from Paper

3.1. Time series and state space

Behavior of the system for different parameters is investigated. As mentioned above, it is expected to observe periodic orbits under normal condition and chaotic behavior under faulty status. Numerical simulation of the system under parameter sets provided in Table 1 is performed. Time series of three state variables are showed in Fig 1. Obviously, time series of insulin, glucose and β-cells are positive. Under normal condition, all variable reached periodic orbits after some time (Fig.1A) while in metabolic disorders, the time series exhibit chaotic behavior (Fig.1B). Trajectory of the system under abnormal conditions also confirmed that there are no stable points or periodic orbits (Fig.2). It has been proven in the next section that these time-series are chaotic. According to the author, these periodic and chaotic behaviors under different metabolic condition are accordant with experimental observations (Letellier et al, 2013).

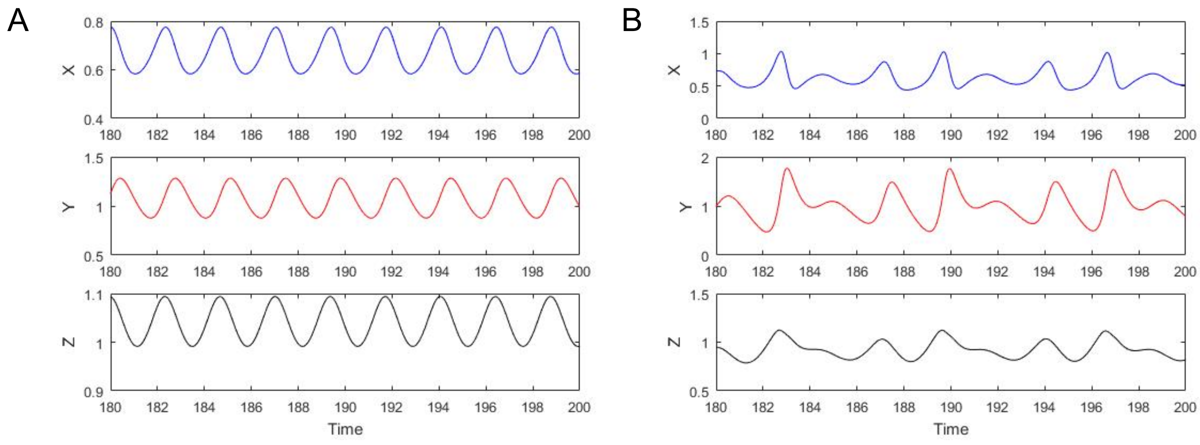


Figure 1| A) Time series of the system with initial condition $x_0 = 0.53,\ y_0 = 1.31,\ z_0 = 1.03$ and parameters for normal conditions given in Table 1. B) Time series of the system with initial condition $x_0 = 0.53,\ y_0 = 1.31,\ z_0 = 1.03$ and parameters for disorder conditions given in Table 1.

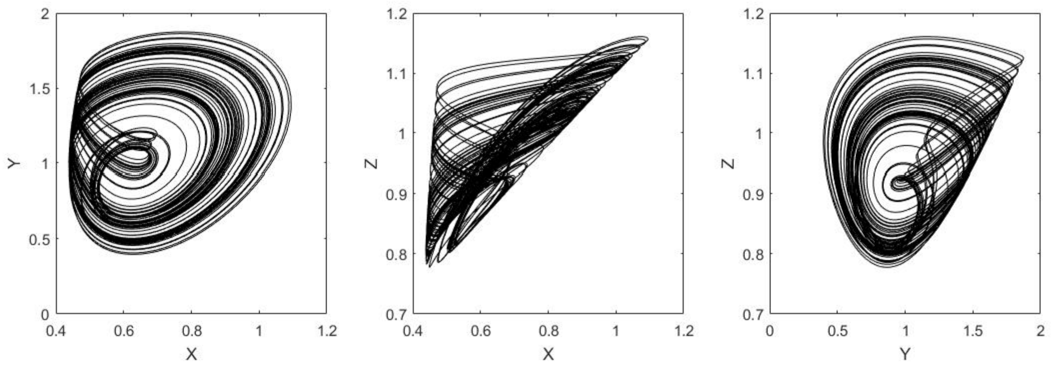


Figure 2| Different projection of the visualization of chaotic attractor of system by trajectory with initial condition $x_0 = 0.53,\ y_0 = 1.31,\ z_0 = 1.03$ and parameters for disorder conditions given in Table 1.

3.2. Stability analysis

The dynamical behavior of the ODE system can be determined by evaluation the eigenvalues of corresponding Jacobian matrix at each of the equilibrium points. The Jacobian matrix of the proposed system is given by:

$$J = \begin{bmatrix} -a_1 + a_2 y & a_2 x + 2a_3 y + 3a_4 y^2 & a_5 + 2a_6 z + 3a_7 z^2 \\ -a_8 - 2a_9 x - 3a_{10} x^2 & -a_8 + a_{11}(1 - 2y) & -a_{12} - 2a_{13} z - 3a_{14} z^2 \\ 0 & a_{15} + 2a_{16} y + 3a_{17} y^2 - a_{19} z & -a_{18} - a_{19} y \end{bmatrix} \quad (3)$$

Since all model variables have specific biological meanings and should be positive, the system has only two fixed points in the allowed state space under abnormal parameters in Table 1 (Table 2). Eigenvalue of Jacobian matrixes at both points are calculated (Table 2). From the eigenvalues we could tell that both points are saddles, the system has no stable fixed points. Similar results are observed for normal sets of parameters (data not shown).

Table 2 Equilibria and eigenvalues of the system under abnormal parameters

Equilibria (x_0, y_0, z_0)	Eigenvalues
(0.6244, 0.9354, 0.8769)	$\lambda_1 = -2.8372$, $\lambda_{2,3} = -0.5262 \pm 2.3472i$
(0.8051, 1.8148, 1.3194)	$\lambda_1 = -1.3802$, $\lambda_{2,3} = -1.7563 \pm 7.5090i$

3.3. Bifurcation diagrams and chaotic analysis

Next, we investigated the bifurcation diagrams of the system against different parameters and biological meanings of these diagrams are discussed. The chaotic behavior of the system signifies the existence of some sort of metabolic disorder. Thus, we could use bifurcation diagrams to simulate some common disorders related to the insulin-glucose regulatory system. Its noteworthy that the system has no stable fixed points for the generation of bifurcation diagrams. Alternatively, we first allow the system to settle down, and the local maxima of time series (X_{max}) are recorded for a few thousand iterations to plot the bifurcation diagrams of the system.

3.3.1. Type 2 diabetes

If the effect of insulin on glucose level is declined or cells of the body is less sensitive to insulin, the blood glucose would increase, which leads to type 2 diabetes. We expect the system to show chaotic behavior under such condition. In the model hypothesis (2), effect of insulin on glucose is represented by the parameter a_8. A decreased a_8 value simulates the reduced insulin sensitivity and higher insulin resistance. Bifurcation diagram of a_8 confirmed that the system is stable for large value of a_8 and when a_8 decreases, the system behaves chaotically (Fig.3).

3.3.2. Hypoglycemia

Existence of too much insulin in blood causes the glucose level to decrease beyond physiological range, leading to a disorder named hypoglycemia. High level of insulin is most possibly caused by inefficient degradation of insulin, which is indicated by reduced value of parameter a_1 in the model. If a_1 gets too low, it proceeds to hypoglycemia. Bifurcation diagrams of a_1 confirms that the system is stable for large value of a_1 but as a_1 decreases the system starts to act in a chaotic manner (Fig.4A).

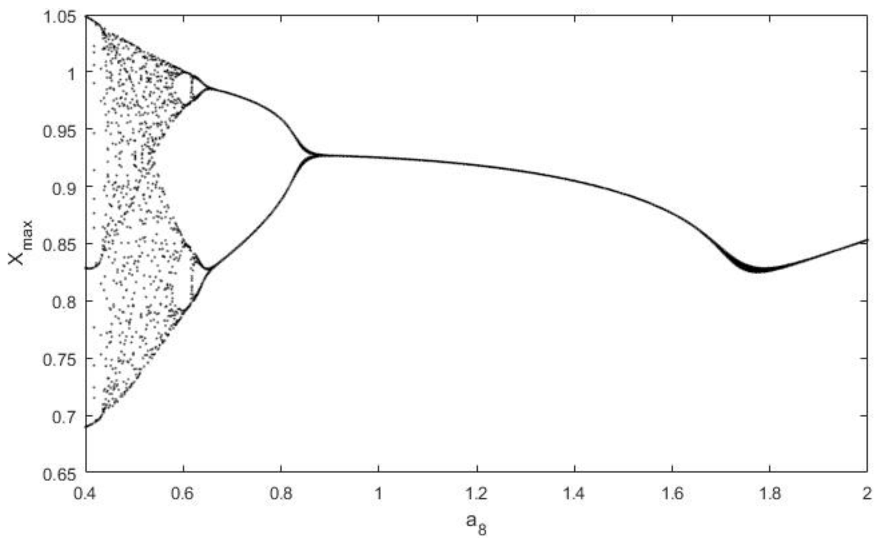


Figure 3| The system bifurcation diagram for parameter a_8

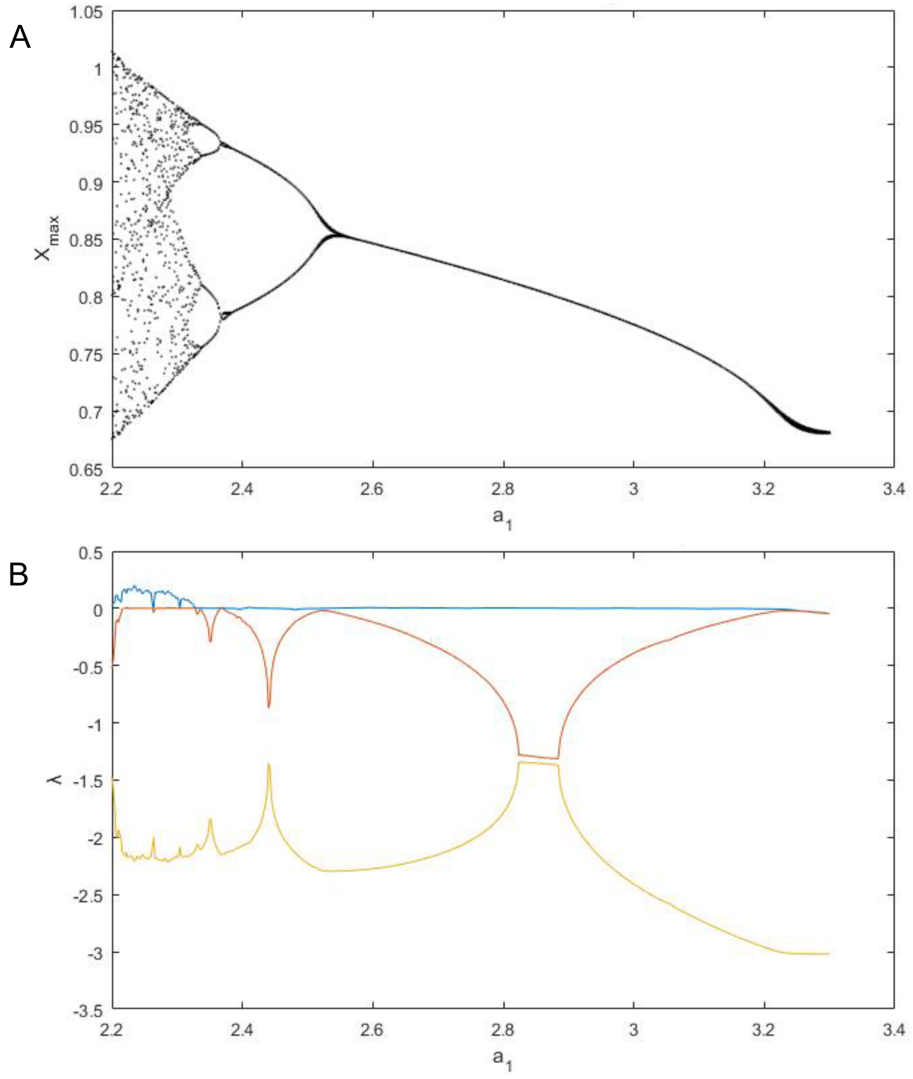


Figure 4| A) The system bifurcation diagram for parameter a_1. B) The Lyapunov exponent diagram for a_1.

To further certify the chaotic characteristics of the system, we calculate the Lyapunov exponent of the system by numerical simulation. To achieve this, we adopted MATLAB code provided by Vasiliy Govorukhin on the File Exchange platform of MathWorks MATLAB Central (Govorukhin, 2020) and make some vital changes to the code to make it feasible for our system. A positive Finite-time Lyapunov exponent is an indicator of chaotic behavior. During numerical simulation, time series of Lyapunov exponents gradually converge to fix values (Fig.S1). By plotting the fixed Lyapunov exponents against a_1 we generated the Lyapunov exponent diagram for a_1 (Fig.4B). Clearly as a_1 decreases, one of the Lyapunov exponents increased from zero to positive, confirming that the system is chaotic for small values of a_1.

3.3.3. Hyperinsulinemia

If β-cells of pancreas secrete insulin more than particular quantities in response to the continued high blood glucose levels, hyperinsulinemia occurs. This could be observed in the proposed model by adjusting the parameter a_7, which illustrates amplified rate of insulin level secreted by β-cells. Bifurcation diagram of a_7 confirmed that higher level of a_7 leads to chaos in the system (Fig.5).

3.3.4. Type 1 diabetes

Type 1 diabetes involves an autoimmune destruction of the insulin-producing β-cells in the pancreas. In the model, if the rate of increase in the population density of β-cells, represented by a_{15}, decreases, the pancreas could not secret enough insulin to control the glucose level and the system is expected to behave chaotically. This was confirmed by the bifurcation diagram of a_{15} (Fig.6).

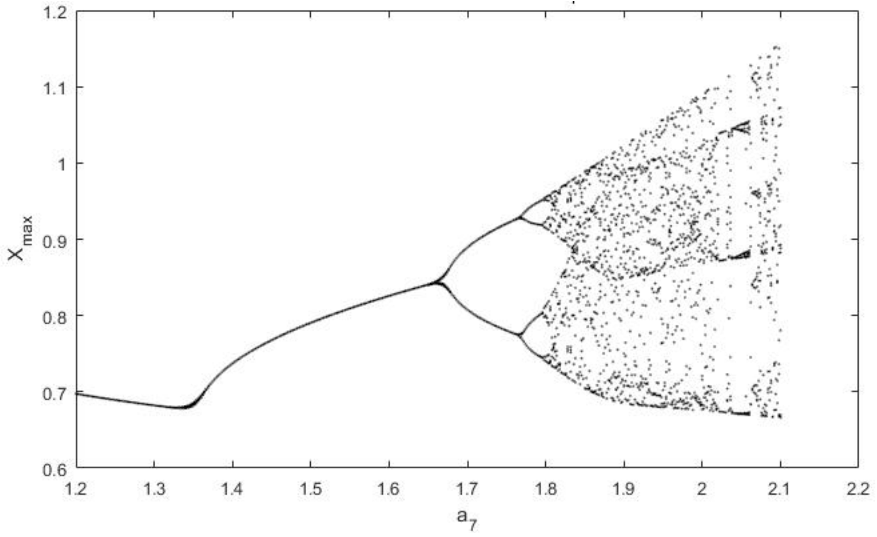


Figure 5| The system bifurcation diagram for parameter a_7

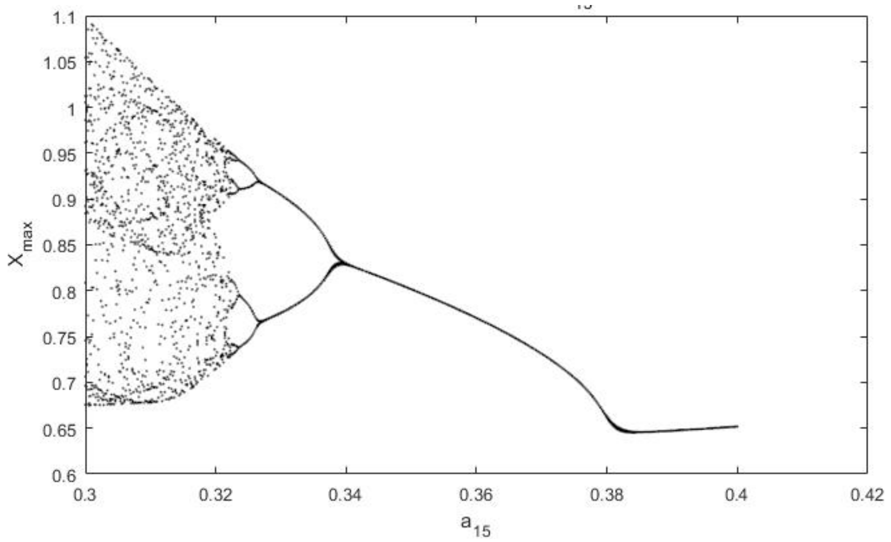


Figure 6| The system bifurcation diagram for parameter a_{15}

3.4. Sensitivity to initial condition

For chaotic dynamics, a minor variation in initial conditions may cause significantly different dynamic behavior. Therefor, a small fluctuation in the insulin concentration may result in unpredictable outcomes through time. This is confirmed by numerical simulation that the system is pretty sensitive to the initial value of x_0 (Fig.8), where a small perturbance could result in very distinct time series.

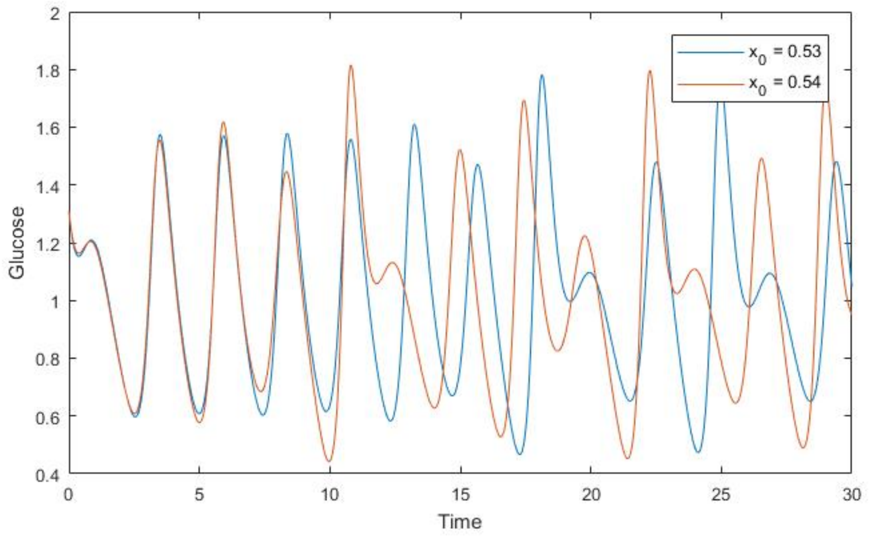


Figure 7| Chaotic evolution of proposed system showing the effect of very small difference in the initial conditions. The initial conditions are $x_0 = 0.53$ and $x'_0 = 0.54$ $(y_0 = y'_0 = 1.31,\ z_0 = z'_0 = 1.03)$.

Part II Our own results

As described above, the proposed model could describe the insulin-glucose regulatory system under physiological conditions and mild metabolic disorders. However, model performance under strong external interference remains to be tested. Glucose tolerance test (GTT) is a common interference to the system and is commonly used in clinical diagnosis. Since glucose tolerance test usually involves introducing very large amount of glucose into the system, it is expected to influence the system behavior dramatically. There have been some successful models for the system under glucose tolerance test, including the classic Minimal Model (MM, Bergman et al, 1979) and improved Single Delay Mode (SDM, Panunzi et al, 2007). These models could predict the behavior of the system after well-defined GTTs. Next, we investigated whether this model performs well for GTTs.

3.5. Model performance on intra-venous glucose tolerance test (IVGTT)

3.5.1. Model tolerance of IVGTT

IVGTT involves intra-venous injection of large amount of glucose into the blood. Typical dosage of IVGTT is 0.33 g Glucose/kg Body Weight, approximately equivalent to 24 mmol/L blood glucose level. This is 4-6 folds higher than the normal glucose level (3.9-6.1 mmol/L). Given the chaotic property of the proposed model, we speculate that the model could not tolerate such strong perturbance. After normalizing the injected glucose concentration to the scale of model variable Y, we simulated the model behavior after a typical IVGTT. As expected, the injected dose of glucose was too extreme for the model and it iterates to an obviously not reasonable state (Fig.S2). From this result, we could tell that the model could not simulated the system after a typical IVGTT.

3.5.2. *Model tolerance and phase dependency of glucose injection*

Since the 0.33 g Glucose/kg Body Weight concentration of glucose injection proves too extreme for this model, we next test the maximum level of glucose injection that the model could tolerate. Considering that the steady state of the model is periodic orbits, the maximum tolerance of glucose injection should vary with the phase of periodic orbits. And the behavior of the system after injection might also vary with the phase of the orbits.

We use the fluctuation of glucose level as the indicator of phase. First, we simulated injecting glucose at the highest concentration of glucose in the periodic orbit. By increasing the dose of injection gradually, the highest level of glucose injection allowed by the system is 3.5282 mmol/L, corresponding to 0.0483 g Glucose/kg Body Weight, which is significantly lower than the recommended dosage of IVGTT. Actually, this maximum tolerance of injection is even lower than the normal blood glucose level (3.9 mmol/L). Maximum tolerance of glucose injection at the lowest level of glucose in the periodic orbit is comparatively higher and reaches about 0.097 g Glucose/kg Body Weight. This is still much lower than the dosage of IVGTT.

We also recorded the time series of the model variables after glucose injection at both highest and lowest glucose level (Fig.8A, B). The time series showed some common feature: after injection of glucose, both the insulin and glucose level first decreases sharply while population density of β-cell soon start to grow. Then insulin concentration increase, followed by the increase of glucose. Finally concentration of insulin and glucose, as well as population density of β-cells start to decrease and the system returns to the same periodic orbit (Fig.8A, B). However, the time series of the proposed model are quite different from experimental data and predictions of previous IVGTT models (Fig.8C, Panunzi et al, 2007). Taken together, the proposed model has very poor performance on IVGTT.

3.6. Model performance on oral glucose tolerance test (OGTT)

IVGTT involves acute injection of high dosage of glucose, while is too extreme of the model to tolerate. By contrast, in oral glucose tolerance test glucose in absorbed by the digestion system and is thus much milder. We figure that the model might be able to have better performance for OGTT. However, OGTT doesn’t involve direct manipulation of blood glucose and could not be simulated by simply changing model variables. Considering that glucose dosage of oral administration is also very high, and that the absorption of glucose in the epithelium of intestinal villus is mainly through facilitated diffusion and active transport, both of which requires the involvement of carrier proteins, we conjecture that high level of glucose in the intestine should be enough to occupy all the carrier proteins, thus making the rate of glucose absorption a constant. Under this hypothesis, the model under OGTT should be:

$$\frac{dx}{dt} = -a_1x + a_2xy + a_3y^2 + a_4y^3 + a_5z + a_6z^2 + a_7z^3 + a_{20}$$

$$\frac{dy}{dt} = -a_8xy - a_9x^2 - a_{10}x^3 + a_{11}y(1-y) - a_{12}z - a_{13}z^2 - a_{14}z^3 + a_{21} + C$$

$$\frac{dz}{dt} = a_{15}y + a_{16}y^2 + a_{17}y^3 - a_{18}z - a_{19}yz \tag{4}$$

With this hypothesis we simulate the system after OGTT and record the time series. According to our results, after the oral administration of glucose, the system soon reaches a new steady state, which seems to be a period-2 orbit and stays in the orbit until end of glucose absorption (Fig.9). The period-2 orbit is featured by stronger fluctuations for all state variables, as well as lower time frequency. Then the system returns to the same period orbit as previous physiological conditions (Fig.9).

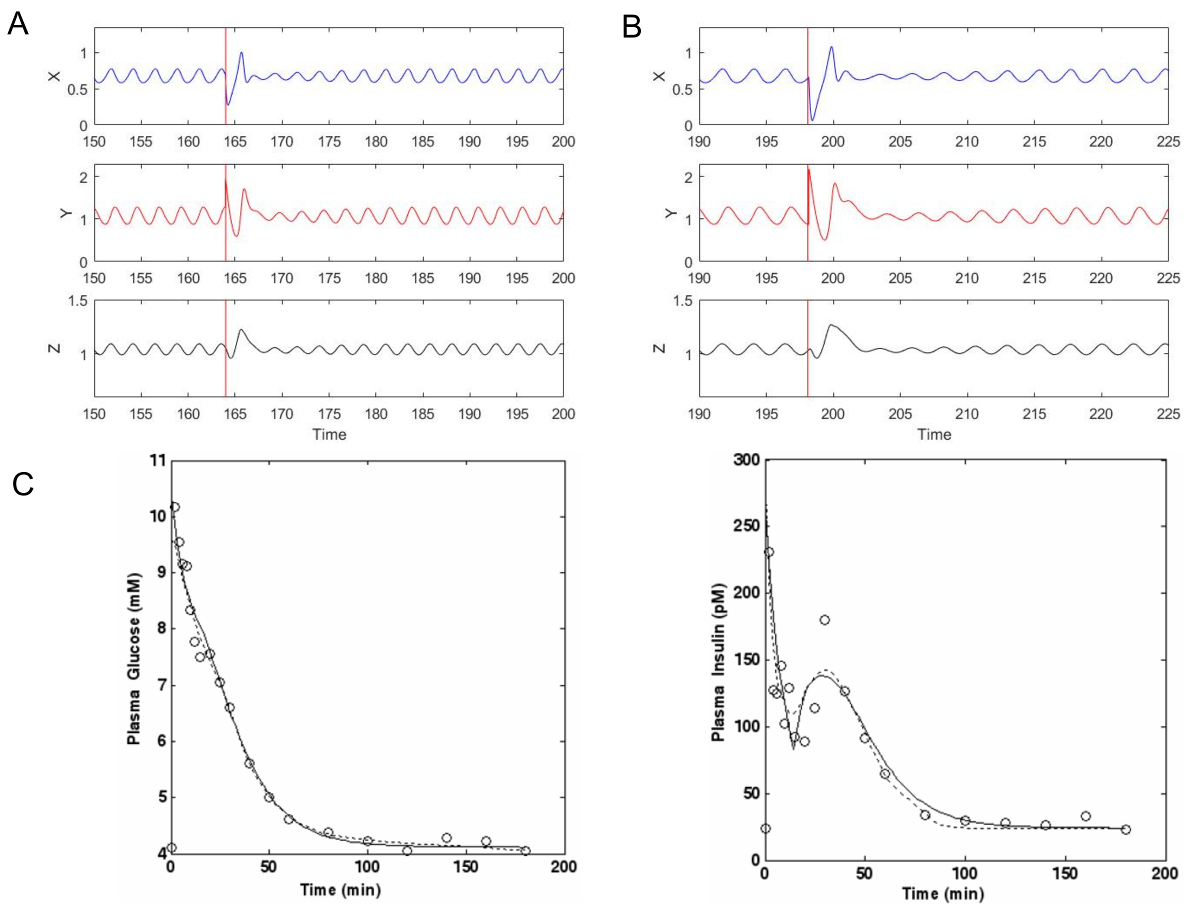


Figure 8| A) Time series after glucose injection at highest level of glucose, time of injection is indicated by red line. B) Time series after glucose injection at lowest level of glucose, time of injection is indicated by red line. C) Experimental observations of glucose and insulin level (circles) after IVGTT with the predicted time-curves from the SDM (continuous lines) and the MM (dotted lines) (Panunzi et al, 2007)

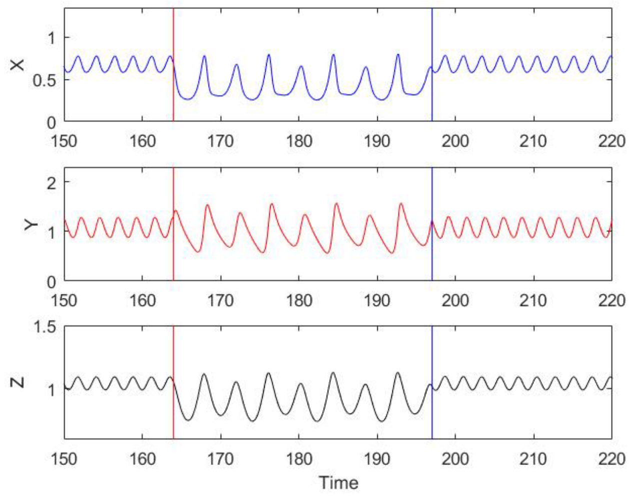


Figure 9| Time series of OGTT, time of oral administration is indicated by red line and end of absorption indicated by blue line.

4. Discussions

In the final project, we implement a new continuous nonlinear mathematical model for insulin-glucose regulatory system. The model is based on the prey and predator model and could mimic the interaction between blood glucose, insulin, and β-cells in both normal and abnormal conditions. Comparing with precious model, this new model exhibits various behaviors for different sets of parameters, including chaos, which was clinically observed in previous researches. Through analyzing model behavior in different parameters, we could reproduce several common metabolic disorders related to this system and identify key biological process relevant to the initiation and development of these diseases. Hopefully further research of this model could provide insights into the prevention and therapy of these metabolic disorders.

However, there are many apparent deficiencies of this piece of work. First, the units of three state variable, as well as the time term, are very ambiguous. And the author hasn't mentioned how the blood glucose level, or insulin concentration should be normalized to fit in the model. This makes it very difficult to apply this model to real experimental results. In my own work I assume that the fluctuation of Y in periodic orbit corresponds to the normal range of blood glucose and use this ratio to normalize blood glucose level with state variables in the model. Obviously, there are other ways of normalization such as by the absolute value of Y and blood glucose level. Different ways of normalization could result in different behaviors in the system.

Another problem of this work is that it uses a single parameter to mimic a certain type of disorder, such as a_8 for Type 2 diabetes and a_7 for hyperinsulinemia. Such inference is possibly arbitrary and lacks persuasion. Although the insulin-glucose regulatory system has been well studied, the metabolic processes involved in the initiation and development of these disease are still too complicated to be explained by a single parameter. This might work for certain situations. For example, the main cause of Type 2 diabetes is indeed impaired insulin sensitivity. But this is not always the case. Diseases are usually the results of multiple factors and there is not necessarily a single dominant factor. We think that this way of analyzing disease with the model is not persuasive enough.

To sum up, this model provides a new approach to investigate the insulin-glucose system and is able to mimic some common type of metabolic disorders. With this model we could find key parameters for each type of disorder and possibly help with the treatment of these disease. However, the universality and accuracy of the model is limited. The model still needs further optimization to have a better performance and applicability for clinical uses.

5. Contribution of each team member

This is an individual project conducted by Zhen-Yu Liu alone. Z.-Y.L. undertakes all the works including but not limited to project conceiving, literature searching and reading, code composition, model implementation, results analyzing, visualization, and manuscript writing.

6. Reference

[1] Shabestari, P.S., S. Panahi, B. Hatef, S. Jafari, and J.C. Sprott. 2018. A new chaotic model for glucose-insulin regulatory system. Chaos Solitons & Fractals. 112:44-51. This is the main paper of which this project follows.

[2] Yau, J.W., S.L. Rogers, R. Kawasaki, E.L. Lamoureux, J.W. Kowalski, T. Bek, S.J. Chen, J.M. Dekker, A. Fletcher, J. Grauslund, S. Haffner, R.F. Hamman, M.K. Ikram, T. Kayama, B.E. Klein, R. Klein, S. Krishnaiah, K. Mayurasakorn, J.P. O'Hare, T.J. Orchard, M. Porta, M. Rema, M.S. Roy, T. Sharma, J. Shaw, H. Taylor, J.M. Tielsch, R. Varma, J.J. Wang, N. Wang, S. West, L. Xu, M. Yasuda, X. Zhang, P. Mitchell, T.Y. Wong, and G. Meta-Analysis for Eye Disease Study. 2012. Global prevalence and major risk factors of diabetic retinopathy. Diabetes Care. 35:556-564.

[3] Shibasaki, T., H. Takahashi, T. Miki, Y. Sunaga, K. Matsumura, M. Yamanaka, C. Zhang, A. Tamamoto, T. Satoh, J. Miyazaki, and S. Seino. 2007. Essential role of Epac2/Rap1 signaling in regulation of insulin granule dynamics by cAMP. Proc Natl Acad Sci U S A. 104:19333-19338.

[4] Letellier, C., F. Denis, and L.A. Aguirre. 2013. What can be learned from a chaotic cancer model? Journal of Theoretical Biology. 322:7-16.

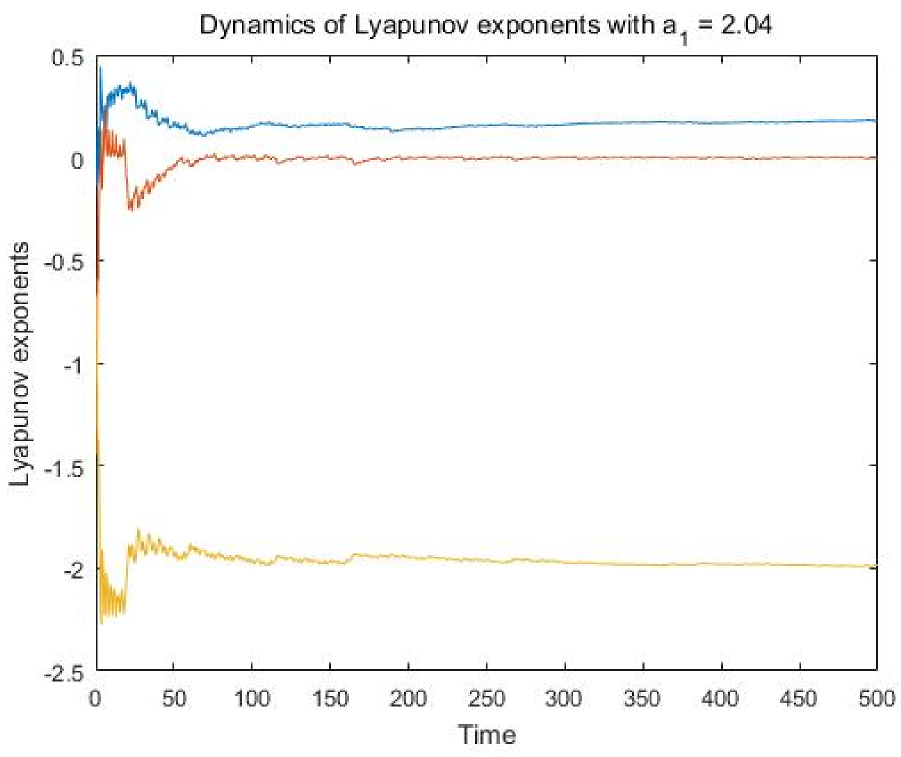
[5] Vasiliy Govorukhin (2020). Calculation Lyapunov Exponents for ODE, MATLAB Central File Exchange. (https://www.mathworks.com/matlabcentral/fileexchange/4628-calculation-lyapunov-exponents-for-ode) Retrieved June 19, 2020. This is the MATLAB code we utilize and adjust to calculate Lyapunov exponent of the system.

[6] Bergman, R.N., Y.Z. Ider, C.R. Bowden, and C. Cobelli. 1979. Quantitative estimation of insulin sensitivity. Am J Physiol. 236: E667-677. In this paper the author proposed the Minimal Model for IVGTT.

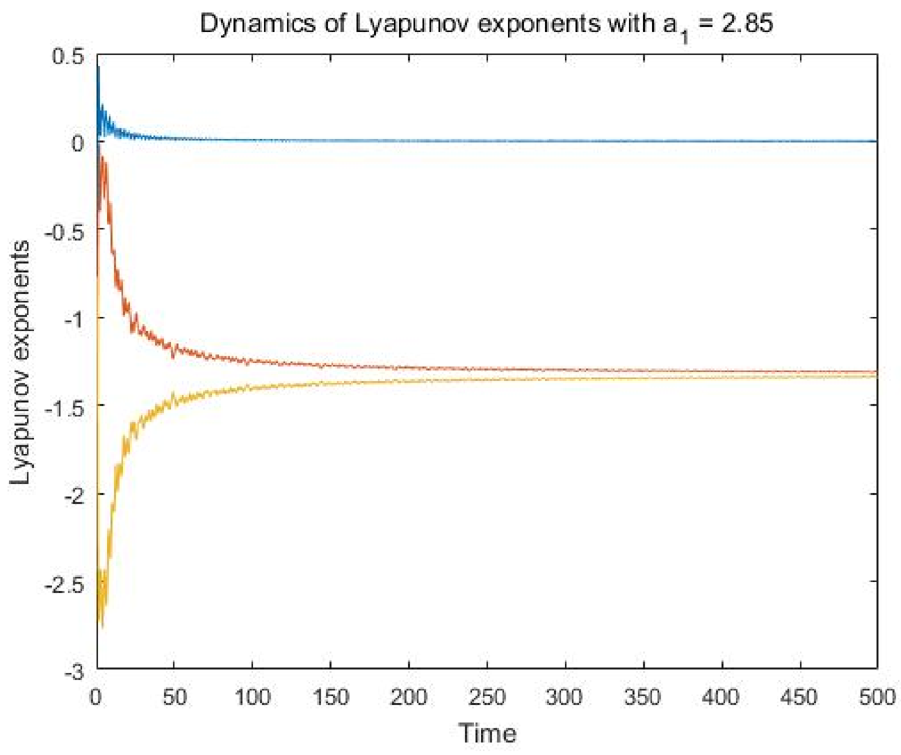
[7] Panunzi, S., P. Palumbo, and A. De Gaetano. 2007. A discrete Single Delay Model for the Intra-Venous Glucose Tolerance Test. Theor Biol Med Model. 4:35. In this paper the author proposed the Single Delay Model for IVGTT, we also cite a figure form this paper showing time series of IVGTT and predictions from Minimal Model and Single Delay Model (Fig.8C).

7. Supplementary Figure

A



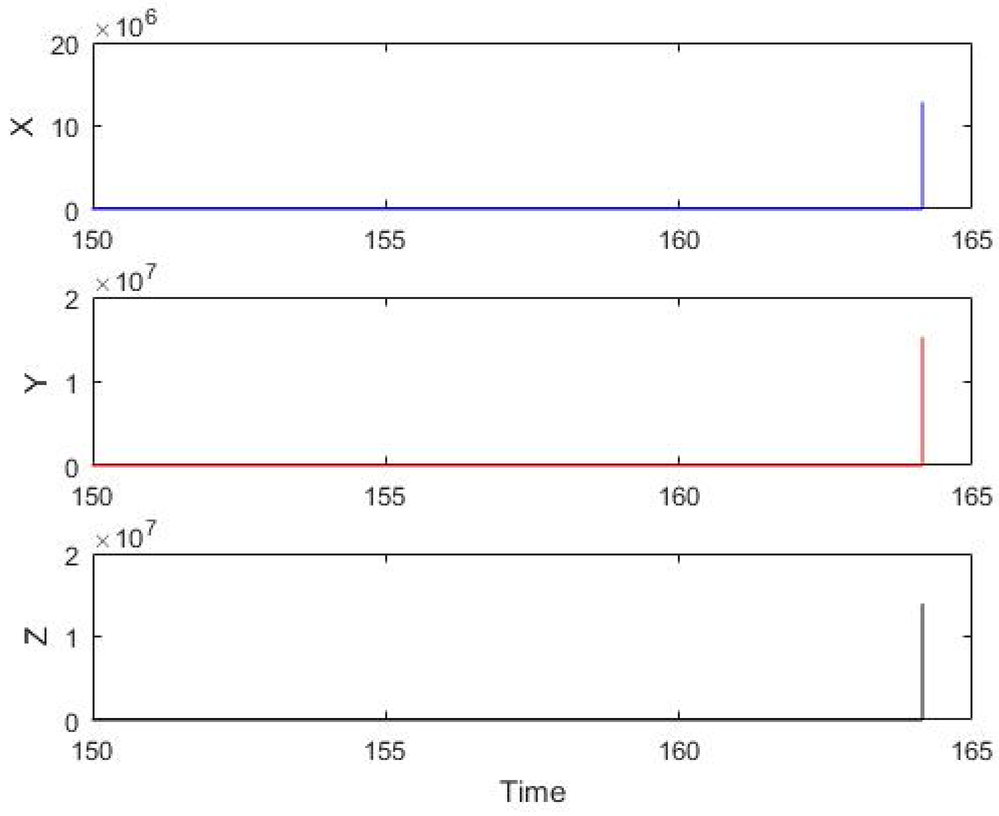
B



Supplementary Figure 1| Time Series of Lyapunov Exponents of the System.

A) Time series of there Lyapunov exponents during simulation for $a_1 = 2.04$

B) Time series of there Lyapunov exponents during simulation for $a_1 = 2.85$



Supplementary Figure 2| Time series of state variables after a typical IVGTT, the results are obviously not reasonable.

8. MATLAB Code

8.1. Initialize parameters

```
%%  Mathematical Modeling in the Life Sciences -- Final Project
% Main function: A Chaotic Model for Insulin-Glucose System
%
% @author: Zhen-Yu Liu (刘振宇)
% @student ID: 1700017853
% @time: 2020.5.20
%
%% 1. initialize parameters
p = [2.04 0.1 1.09 -1.08 0.03 -0.06 2.01 0.22 -3.84 -1.2 0.3
 1.37 -0.3 0.22 0.3 -1.35 0.5 -0.42 -0.15 -0.19 -0.56];
```

8.2. Time series and state space

First write MATLAB function to calculate derivative at certain point:

```
function Deriv = Derivative(t, loc, p)
% compute derivative for the X, Y and Z over t
    x= loc(1);
    y= loc(2);
    z= loc(3);
    Deriv = zeros(3,1);
    dxdt = -p(1)*x + p(2)*x*y + p(3)*y*y + p(4)*y*y*y + p(5)*z +
          p(6)*z*z + p(7)*z*z*z + p(20);
    dydt = -p(8)*x*y - p(9)*x*x - p(10)*x*x*x + p(11)*y*(1-y) -
          p(12)*z - p(13)*z*z - p(14)*z*z*z + p(21);
    dzdt = p(15)*y + p(16)*y*y + p(17)*y*y*y - p(18)*z - p(19)*y*z;
    Deriv = [dxdt; dydt; dzdt];
```

Time series from $x_0 = 0.53,\ y_0 = 1.31,\ z_0 = 1.03$ under normal parameters. (Fig.1A)

```
%% 2. Time series and state space
initial_loc = [0.53, 1.31, 1.03];

% numeriacl simulation for the trajectory

% a1 = 3
p(1) = 3;
options = odeset('RelTol',1e-5);
[t,loc] = ode45(@Derivative, [0, 200], initial_loc, options, p);

subplot(3,1,1)
plot(t,loc(:,1),'-b')
xlim([180,200]); ylim([0.4,0.8]);ylabel('X');
subplot(3,1,2)
plot(t,loc(:,2),'-r')
xlim([180,200]); ylim([0.5,1.5]);ylabel('Y');
subplot(3,1,3)
plot(t,loc(:,3),'-k')
xlim([180,200]); ylim([0.9,1.1]);ylabel('Z');xlabel('Time');
```

Time series from $x_0 = 0.53,\ y_0 = 1.31,\ z_0 = 1.03$ under abnormal parameters. (Fig.1B)

```
p(1) = 2.04; % a1 = 2.04
options = odeset('RelTol',1e-5);
[t,loc] = ode45(@Derivative, [0, 200], initial_loc, options, p);

subplot(3,1,1)
plot(t,loc(:,1),'-b')
xlim([180,200]); ylim([0,1.5]);ylabel('X');
subplot(3,1,2)
plot(t,loc(:,2),'-r')
xlim([180,200]); ylim([0,2]);ylabel('Y');
subplot(3,1,3)
plot(t,loc(:,3),'-k')
xlim([180,200]); ylim([0.5,1.5]);ylabel('Z');xlabel('Time');
```

Different projection of the visualization of trajectory with previous initial condition (Fig.2)

```
% visulize trajectory and attractor
plot3(loc(:,1),loc(:,2),loc(:,3))
subplot(1,3,1)
plot(loc(:,1),loc(:,2),'-k');xlabel('X');ylabel('Y');
subplot(1,3,2)
plot(loc(:,1),loc(:,3),'-k');xlabel('X');ylabel('Z');
subplot(1,3,3)
plot(loc(:,2),loc(:,3),'-k');xlabel('Y');ylabel('Z');
```

8.3. Stability analysis

First write MATLAB function to solve equilibria of the system and filter these equilibria in the state space.

```
function equi = SolveEquilibria(p)
% solve equilibrias under given parameter set p
% returns the coordinates of equilibria, saved in rows

    %create symbolic expressions
    syms x y z;
    eq1 = -p(1)*x + p(2)*x*y + p(3)*y*y + p(4)*y*y*y + p(5)*z + p(6)*z*z
          + p(7)*z*z*z + p(20);
    eq2 = -p(8)*x*y - p(9)*x*x - p(10)*x*x*x + p(11)*y*(1-y) - p(12)*z -
          p(13)*z*z - p(14)*z*z*z + p(21);
    eq3 = p(15)*y + p(16)*y*y + p(17)*y*y*y - p(18)*z - p(19)*y*z;

    % solve equilibrias using built-in solve function
    [x,y,z] = solve(eq1,eq2,eq3,'x,y,z');

    % convert symbolic expressions into complex number
    x = double(x);
    y = double(y);
    z = double(z);

    % filter reasonable solutions: positive and real
    x_select = (real(x)>0) & (imag(x)==0);
    y_select = (real(y)>0) & (imag(y)==0);
    z_select = (real(z)>0) & (imag(z)==0);

    select = x_select & y_select & z_select;

    x = x(select);
    y = y(select);
    z = z(select);

    % generate output: coordinates of equilibrias
    equi = [x,y,z];
```

Then we solve equilibria with the function above.

```
%% 3. Equilibria and Stability analysis
% solve equilibria
equi_loc = SolveEquilibria(p);    % 2 equilibria
```

Next, we write function to calculate Jacobian matrix at certain point

```
function J = Jacobian(loc, p)
% compute Jacobian at certain point under given parameters
    x= loc(1);
    y= loc(2);
    z= loc(3);
    J = zeros(3,3);
    J(1,1) = -p(1) + p(2)*y;
    J(1,2) = p(2)*x + 2*p(3)*y + 3*p(4)*y*y;
    J(1,3) = p(5) + 2*p(6)*z + 3*p(7)*z*z;
    J(2,1) = -p(8)*y - 2*p(9)*x - 3*p(10)*x*x;
    J(2,2) = -p(8)*x + p(11)*(1-2*y);
    J(2,3) = -p(12)- 2*p(13)*z - 3*p(14)*z*z;
    J(3,1) = 0;
    J(3,2) = p(15) + 2*p(16)*y + 3*p(17)*y*y - p(19)*z;
    J(3,3) = -p(18) - p(19)*y;
```

Then we calculate the Jacobian matrix at equilibria and corresponding eigenvalues.

```
% calculate Jacobians
J1 = Jacobian(equi_loc(1,:),p);
J2 = Jacobian(equi_loc(2,:),p);

% calculate eigenvalue and eigenvectors for Jacobians
[V1, D1] = eig(J1);
[V2, D2] = eig(J2);

D1 = diag(D1);
D2 = diag(D2);
```

8.4.Bifurcation diagrams

First write function to construct bifurcation diagram of the system over certain parameter:

```
function Bifurcation(which, from, to, step, initial_loc, N_peroid, p)
% construct bifurcation diagram for certain parameter
    options = odeset('RelTol',1e-5);
    figure;

    for a = from:step:to
        str=['current value of a is '  num2str(a)];
        disp(str);
        p_temp = p;
        p_temp(which) = a;
        [t_temp,loc_temp] = ode45(@Derivative, [0:0.001:100],
                                  initial_loc, options, p_temp);
        record = loc_temp(end-N_peroid+1:end,1);
        % time  = t_temp(end-N_peroid+1:end,1);
        % plot(time, record)
        record = findpeaks(record);
        % disp(size(record,1));
        plot(ones(1,size(record,2))*a, record, '.k', 'markersize', 1);
        hold on;
    end

    title(['Bifurcation Diagram of a_'  num2str(which)]);
    xlabel(['a_'  num2str(which)]);
    ylabel('X_m_a_x');
```

Then construct bifurcation diagram of a_8, a_1, a_7 and a_{15} (Fig.3, Fig.4A, Fig.5, Fig.6)

```
%% 4. Bifurcation diagrams and relevant desease
initial_loc = [0.53, 1.31, 1.03];
N_peroid = 30000;


% bifurcation diagram of a_8
Bifurcation(8, 0.4, 2, 0.002, initial_loc, N_peroid, p)
% bifurcation diagram of a_1
Bifurcation(1, 2.2, 3.3, 0.002, initial_loc, N_peroid, p)
% bifurcation diagram of a_7
Bifurcation(7, 1.2, 2.1, 0.002, initial_loc, N_peroid, p)
```

```
% bifurcation diagram of a_15
Bifurcation(15, 0.3, 0.4, 0.0002, initial_loc, N_peroid, p)
```

8.5. *Lyapunov exponents and chaotic behavior*

We use the function provided by Vasiliy Govorukhin on the File Exchange platform of MathWorks MATLAB Central (Govorukhin, 2020) to calculate Lyapunov exponents, the function is given as:

```
function [Texp,Lexp]=lyapunov(n,rhs_ext_fcn,fcn_integrator,tstart,stept,
                             tend,ystart,ioutp);
%
%    Lyapunov exponent calcullation for ODE-system.
%
%    The alogrithm employed in this m-file for determining Lyapunov
%    exponents was proposed in
%
%         A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano,
%        "Determining Lyapunov Exponents from a Time Series," Physica D,
%        Vol. 16, pp. 285-317, 1985.
%
%    For integrating ODE system can be used any MATLAB ODE-suite methods.
%    This function is a part of MATDS program - toolbox for dynamical
%    system investigation
%    See:    http://www.math.rsu.ru/mexmat/kvm/matds/
%
%    Input parameters:
%      n - number of equation
%      rhs_ext_fcn - handle of function with right hand side of
%                    extended ODE-system.
%              This function must include RHS of ODE-system coupled with
%              variational equation
%              (n items of linearized systems, see Example).
%      fcn_integrator - handle of ODE integrator function,
%                       for example: @ode45
%      tstart - start values of independent value (time t)
%      stept - step on t-variable for Gram-Schmidt renormalization
%              procedure.
%      tend - finish value of time
%      ystart - start point of trajectory of ODE system.
%      ioutp - step of print to MATLAB main window. ioutp==0 - no print,
%              if ioutp>0 then each ioutp-th point will be print.
%
%    Output parameters:
%      Texp - time values
%      Lexp - Lyapunov exponents to each time value.
%
%    Users have to write their own ODE functions for their specified
%    systems and use handle of this function as rhs_ext_fcn - parameter.
%
%    Example. Lorenz system:
%               dx/dt = sigma*(y - x)     = f1
%               dy/dt = r*x - y - x*z = f2
%               dz/dt = x*y - b*z     = f3
%
%    The Jacobian of system:
%        | -sigma  sigma  0 |
%    J = |   r-z     -1    -x |
%        |    y       x    -b |
%
%    Then, the variational equation has a form:
%
%    F = J*Y
%    where Y is a square matrix with the same dimension as J.
%    Corresponding m-file:
```

```
%        function f=lorenz_ext(t,X)
%         SIGMA = 10; R = 28; BETA = 8/3;
%         x=X(1); y=X(2); z=X(3);
%
%         Y= [X(4), X(7), X(10);
%             X(5), X(8), X(11);
%             X(6), X(9), X(12)];
%         f=zeros(9,1);
%         f(1)=SIGMA*(y-x); f(2)=-x*z+R*x-y; f(3)=x*y-BETA*z;
%
%         Jac=[-SIGMA,SIGMA,0; R-z,-1,-x; y, x,-BETA];
%
%         f(4:12)=Jac*Y;
%
%    Run Lyapunov exponent calculation:
%
%    [T,Res]=lyapunov(3,@lorenz_ext,@ode45,0,0.5,200,[0 1 0],10);
%
%    See files: lorenz_ext, run_lyap.
%
% --------------------------------------------------------------------
% Copyright (C) 2004, Govorukhin V.N.
% This file is intended for use with MATLAB and was produced for
%   MATDS-program
% http://www.math.rsu.ru/mexmat/kvm/matds/
% lyapunov.m is free software. lyapunov.m is distributed in the hope
% that it will be useful, but WITHOUT ANY WARRANTY.
%
%
%       n=number of nonlinear odes
%       n2=n*(n+1)=total number of odes
%
n1=n; n2=n1*(n1+1);
%  Number of steps
nit = round((tend-tstart)/stept);
% Memory allocation
y=zeros(n2,1); cum=zeros(n1,1); y0=y;
gsc=cum; znorm=cum;
% Initial values
y(1:n)=ystart(:);
for i=1:n1 y((n1+1)*i)=1.0; end;
t=tstart;
% Main loop
for ITERLYAP=1:nit
% Solutuion of extended ODE system
  [T,Y] = feval(fcn_integrator,rhs_ext_fcn,[t t+stept],y);


  t=t+stept;
  y=Y(size(Y,1),:);
  for i=1:n1
      for j=1:n1 y0(n1*i+j)=y(n1*j+i); end;
  end;
%
%       construct new orthonormal basis by gram-schmidt
%
  znorm(1)=0.0;
  for j=1:n1 znorm(1)=znorm(1)+y0(n1*j+1)^2; end;
  znorm(1)=sqrt(znorm(1));
  for j=1:n1 y0(n1*j+1)=y0(n1*j+1)/znorm(1); end;
  for j=2:n1
      for k=1:(j-1)
          gsc(k)=0.0;
          for l=1:n1 gsc(k)=gsc(k)+y0(n1*l+j)*y0(n1*l+k); end;
```

```
    end;
  %
  %       construct new orthonormal basis by gram-schmidt
  %
    znorm(1)=0.0;
    for j=1:n1 znorm(1)=znorm(1)+y0(n1*j+1)^2; end;
    znorm(1)=sqrt(znorm(1));
    for j=1:n1 y0(n1*j+1)=y0(n1*j+1)/znorm(1); end;
    for j=2:n1
        for k=1:(j-1)
            gsc(k)=0.0;
            for l=1:n1 gsc(k)=gsc(k)+y0(n1*l+j)*y0(n1*l+k); end;
        end;

        for k=1:n1
            for l=1:(j-1)
                y0(n1*k+j)=y0(n1*k+j)-gsc(l)*y0(n1*k+l);
            end;
        end;
        znorm(j)=0.0;
        for k=1:n1 znorm(j)=znorm(j)+y0(n1*k+j)^2; end;
        znorm(j)=sqrt(znorm(j));
        for k=1:n1 y0(n1*k+j)=y0(n1*k+j)/znorm(j); end;
    end;
  %
  %       update running vector magnitudes
  %
    for k=1:n1 cum(k)=cum(k)+log(znorm(k)); end;
  %
  %       normalize exponent
  %
    for k=1:n1
        lp(k)=cum(k)/(t-tstart);
    end;
  % Output modification
    if ITERLYAP==1
       Lexp=lp;
       Texp=t;
    else
       Lexp=[Lexp; lp];
       Texp=[Texp; t];
    end;
    if (mod(ITERLYAP,ioutp)==0)
       fprintf('t=%6.4f',t);
       for k=1:n1 fprintf(' %10.6f',lp(k)); end;
       fprintf('\n');
    end;
    for i=1:n1
        for j=1:n1
            y(n1*j+i)=y0(n1*i+j);
        end;
    end;
  end;
```

Also, we need to rewriter the supporting function to make it usable for our system.

```
  function f=lorenz_ext(t,X)
  %
  %   adapted from
  %   Vasiliy Govorukhin (2020). Calculation Lyapunov Exponents for ODE
  %   (https://www.mathworks.com/matlabcentral/fileexchange/4628-calculation-lyap
  nents-for-ode),
  %   MATLAB Central File Exchange. Retrieved June 15, 2020.
  %

  %%
      global p;
```

```
    x=X(1); y=X(2); z=X(3);
    Y= [X(4), X(7), X(10);
    X(5), X(8), X(11);
    X(6), X(9), X(12)];
    f=zeros(9,1);

    % ODE equation
    f(1) = -p(1)*x + p(2)*x*y + p(3)*y*y + p(4)*y*y*y + p(5)*z +
           p(6)*z*z + p(7)*z*z*z + p(20);
    f(2) = -p(8)*x*y - p(9)*x*x - p(10)*x*x*x + p(11)*y*(1-y) -
           p(12)*z - p(13)*z*z - p(14)*z*z*z + p(21);
    f(3) = p(15)*y + p(16)*y*y + p(17)*y*y*y - p(18)*z - p(19)*y*z;

    %Linearized system
     Jac=Jacobian(X, p);

     %Variational equation
    f(4:12)=Jac*Y;
    %Output data must be a column vector
```

With the function above, we could calculate the Lyapunov exponents of the system, we first record the dynamics of Lyapunov exponents during simulation (Fig.S1)

```
%% 5. Lyapunov Exponents and chaotic behavior

% Dynamics of Lyapunov exponents under initial parameters
p(1) = 2.04;
[T,Res]=lyapunov(3,@lorenz_ext,@ode45,0,0.5,500,initial_loc,10);
plot(T,Res);
title('Dynamics of Lyapunov exponents with a_1 = 2.04');
xlabel('Time'); ylabel('Lyapunov exponents');

p(1) = 2.85;
[T,Res]=lyapunov(3,@lorenz_ext,@ode45,0,0.5,500,initial_loc,10);
plot(T,Res);
title('Dynamics of Lyapunov exponents with a_1 = 2.85');
xlabel('Time'); ylabel('Lyapunov exponents');
```

Then we construct the Lyapunov exponent diagram for a_1 (Fig.4B).

```
% Lyapunov Exponents of peroidic orbits under different a_1 parameter
delta_a1 = 0.002;
LyapunovExp = [];

for a = 2.2:delta_a1:3.3
    p(1)=a;
    [T,Res]=lyapunov(3,@lorenz_ext,@ode45,0,0.5,300,initial_loc,0);
    disp(['current value of a1 is '  num2str(a)]);
    LyapunovExp = [LyapunovExp;Res(end, :)];
end
p(1) = 2.04;

plot(2.2:delta_a1:3.3,LyapunovExp);
title('Lyapunov exponent diagram based on a_1');
xlabel('a_1'); ylabel('¦Ë');
```

8.6. Sensitivity to initial condition

We recorded the time series for different initial condition and plot them (Fig.7)

```
%% 6. Sensitivity to initial condition
initial_loc_1 = [0.53, 1.31, 1.03];
initial_loc_2 = [0.54, 1.31, 1.03];
options = odeset('RelTol',1e-5);
[t_1,loc_1] = ode45(@Derivative, [0, 200], initial_loc_1, options, p);
[t_2,loc_2] = ode45(@Derivative, [0, 200], initial_loc_2, options, p);
```

```
plot(t_1,loc_1(:,2))
hold on
plot(t_2,loc_2(:,2))
xlim([0,30]); ylim([0.4,2]);ylabel('Glucose');xlabel('Time');
legend('x_0 = 0.53','x_0 = 0.54')
```

8.7. Performance on intravenous glucose tolerance test (IVGTT)

First, we simulate the periodic orbit of physiological condition.

```
%% 7.Performance on intravenous glucose tolerance test (IVGTT)

% physiological condition
p(1) = 3;
options = odeset('RelTol',1e-5);
[t,loc] = ode45(@Derivative, [0, 200], initial_loc, options, p);
```

Then we normalize the dosage of glucose injection to the scale of model parameter.

```
% adjust for units
Y_inject = 0.33/180.16*1000/(0.08/1.053); % unit: mmol/L
Y_max = max(loc(end-1000:end,2));
Y_min = min(loc(end-1000:end,2));
ratio = (Y_max-Y_min)/(6.1-3.9);
Y_inject = Y_inject * ratio; %Y_inject = 4.4417
```

Next, we simulate injecting glucose at the maximum value of glucose concentration (Fig.8A, FigS2)

```
% rapid injection of glucose (IVGTT) at the maximum value of Y
find(loc(:,2)==Y_max) % 4011
t_1 = t(1:4011);
loc_1 = loc(1:4011,:);


temp_loc = loc_1(end,:);
temp_loc(2) = temp_loc(2) + 0.65; % maximum tolerance for the model
[t_2,loc_2] = ode45(@Derivative, [0, 200], temp_loc, options, p);


plot(t_2,loc_2(:,2),'-b')


t_2 = t_2 + t_1(end);
t_all = [t_1(end-400:end); t_2(1:1000)];
loc_all = [loc_1(end-400:end,:); loc_2(1:1000,:)];


subplot(3,1,1)
plot(t_all,loc_all(:,1),'-b'); line([164 164],[0,1.35],'Color','red');
xlim([150,200]); ylim([0,1.35]);ylabel('X');
subplot(3,1,2)
plot(t_all,loc_all(:,2),'-r'); line([164 164],[0,2.3],'Color','red');
xlim([150,200]);  ylim([0,2.3]); ylabel('Y');
subplot(3,1,3)
plot(t_all,loc_all(:,3),'-k');line([164 164],[0.6,1.5],'Color','red');
xlim([150,200]);  ylim([0.6,1.5]); ylabel('Z');xlabel('Time');
```

Similarly, we simulate injecting glucose at the minimum value of glucose concentration in the orbit (Fig.8B).

```
% rapid injection of glucose (IVGTT) at the minumum value of Y
find(loc(:,2)==Y_min) % 4844
t_1 = t(1:4844);
loc_1 = loc(1:4844,:);


temp_loc = loc_1(end,:);
temp_loc(2) = temp_loc(2) + 1.3; % maximum tolerance for the model
[t_2,loc_2] = ode45(@Derivative, [0, 200], temp_loc, options, p);


plot(t_2,loc_2(:,2),'-b')
```

```
t_2 = t_2 + t_1(end);
t_all = [t_1(end-400:end); t_2(1:1000)];
loc_all = [loc_1(end-400:end,:); loc_2(1:1000,:)];

subplot(3,1,1)
plot(t_all,loc_all(:,1),'-b'); line([198.1 198.1],[0,1.35],'Color','red');
xlim([190,225]); ylim([0,1.35]);ylabel('X');
subplot(3,1,2)
plot(t_all,loc_all(:,2),'-r'); line([198.1 198.1],[0,2.3],'Color','red');
xlim([190,225]);  ylim([0,2.3]); ylabel('Y');
subplot(3,1,3)
plot(t_all,loc_all(:,3),'-k');line([198.1 198.1],[0.6,1.5],'Color','red');
xlim([190,225]);  ylim([0.6,1.5]); ylabel('Z');xlabel('Time');
```

8.8. Performance on oral glucose tolerance test (OGTT)

First, we simulate the periodic orbit of physiological condition.

```
%% 8.Performance on oral glucose tolerance test (OGTT)
% physiological condition
p(1) = 3;
options = odeset('RelTol',1e-5);
[t,loc] = ode45(@Derivative, [0, 200], initial_loc, options, p);
```

Then we write the function to calculate the derivative under OGTT.

```
function Deriv_OGTT = Derivative_OGTT(t, loc, p)
% compute derivative for the X, Y and Z over t
    x= loc(1);
    y= loc(2);
    z= loc(3);
    Deriv_OGTT = zeros(3,1);
    dxdt = -p(1)*x + p(2)*x*y + p(3)*y*y + p(4)*y*y*y + p(5)*z
      + p(6)*z*z + p(7)*z*z*z + p(20);
    dydt = -p(8)*x*y - p(9)*x*x - p(10)*x*x*x + p(11)*y*(1-y) -
      p(12)*z - p(13)*z*z - p(14)*z*z*z + p(21) + 0.95;
    dzdt = p(15)*y + p(16)*y*y + p(17)*y*y*y - p(18)*z - p(19)*y*z;
    Deriv_OGTT = [dxdt; dydt; dzdt];
```

Finally, we simulate the system for OGTT (Fig.9).

```
t_1 = t(1:4011);
loc_1 = loc(1:4011,:);


temp_loc = loc_1(end,:);
[t_2,loc_2] = ode45(@Derivative_OGTT, [0, 33], temp_loc, options, p);


t_2 = t_2 + t_1(end);


temp_loc = loc_2(end,:);
[t_3,loc_3] = ode45(@Derivative, [0, 40], temp_loc, options, p);


t_3 = t_3 + t_2(end);



t_all = [t_1(end-400:end); t_2; t_3];
loc_all = [loc_1(end-400:end,:); loc_2;loc_3];


subplot(3,1,1)
plot(t_all,loc_all(:,1),'-b'); line([164 164],[0,1.35],'Color','red');
line([197 197],[0,1.35],'Color','blue');
xlim([150,220]); ylim([0,1.35]);ylabel('X');
subplot(3,1,2)
plot(t_all,loc_all(:,2),'-r'); line([164 164],[0,2.3],'Color','red');
line([197 197],[0,2.3],'Color','blue');
xlim([150,220]);  ylim([0,2.3]); ylabel('Y');
```

```
subplot(3,1,3)
plot(t_all,loc_all(:,3),'-k');line([164 164],[0.6,1.5],'Color','red');
line([197 197],[0.6,1.5],'Color','blue');
xlim([150,220]);  ylim([0.6,1.5]); ylabel('Z');xlabel('Time');
```