U.U.D.M. Project Report 2014:27

Comparison Between Deterministic and Stochastic Biological Simulation

Eric Yiqing Wang

Examensarbete i matematik, 15 hp
Handledare och examinator: Ingemar Kaj
Augusti 2014

Department of Mathematics
Uppsala University

Contents

1 Introduction **5**

1.1 Model description 5

1.2 History 7

1.3 Thesis Structure 7

2 Theoretical background **7**

2.1 Introduction to Stochastic Method 7

2.1.1 Monte Carlo Method 7

2.1.2 Markov Process 8

2.2 The Stochastic Simulation Algorithm (SSA) 8

3 Comparison **10**

3.1 Deterministic Method 10

3.2 Stochastic Method 13

4 Discussion and Conclusion **18**

5 Appendix **20**

Acknowledgment

Here I would like to sincerely thanks my supervisor Professor Ingemar Kaj,for the knowledge given from him and his patient guidance to this project.

Abstrcat:

Simulation in biological systems has been developed rapidly in recent years. In this thesis, we implement mathematic modeling of Oscillating Gene and study the mechanism behind that.

The approach we are using is straight forward solving ODE equations in deterministic method and Gillespies algorithm for stochastic simulation, respectively. The main difference between the two realization is the occurrence of steady state with some significance parameters.

1 Introduction

1.1 Model description

Many People notice that they naturally experience regularly tiredness and awakeness through out the day. That helps us maintain enough sleep at night as make up for the hours we awake. The internal circadian clock inside our body ,as well as other organisms, regulate the daily behavior accordingly. In molecular biology, an oscillating gene is a gene that is expressed in a rhythmic pattern or in periodic cycles[2]. Oscillating genes are usually circadian and can be identified by periodic changes in the state of an organism circadian rhythms, controlled by oscillating genes, have a period of approximately 24 hours.

Nowdays research have shown that these clocks share many common features among species. For example, quoting Jose Vilar and others wrote in a scientific article describing this:

> The main characteristic is that the presence of intracellular transcription regulation networks with a set of clock elements that give rise to stable oscillations in gene expression. A positive element activates genes coupled to the circadian clock. It simultaneously promotes the expression of a negative element, which in turn represses the positive element. The cycle completes itself upon degradation of the negative element and reexpression of the positive element.

The biological systems we use in this thesis , is one with minimize effect of stochastic noise on circadian noise, based on the common positive and negative control elements found experimentally[1][3].

There are two important genes involved in the circadian clock model. They are an activator A and a repressor R, which are transcribed into mRNA and subsequently translated into protein. The activator A binds to the A and R promoters, which increase their transcription rate. Thus, A acts as the positive element in transcription, whereas R act as the negative element by sequestering the activator[1].

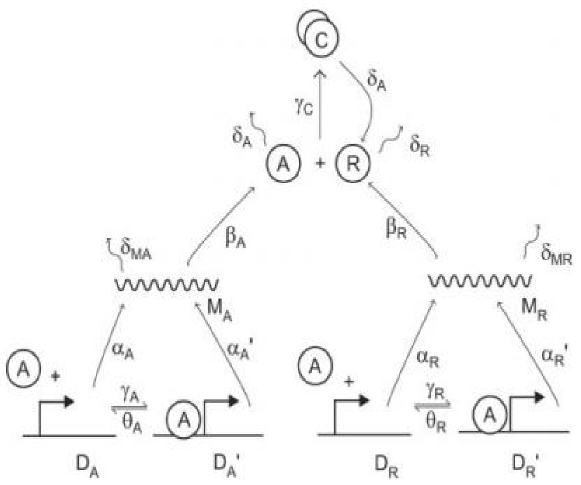


Figure 1. Biochemical network of the circadian oscillator model.

The Circadian oscillator model is given by the following set of reation rate equations:

$$\begin{cases} \frac{dD_A}{dt} = \theta_A D'_A - \gamma_A D_A A \\ \frac{dD_R}{dt} = \theta_R D'_R - \gamma_R D_R A \\ \frac{dD'_A}{dt} = \gamma_A D_A A - \theta_A D'_A \\ \frac{dD'_R}{dt} = \gamma_R D_R A - \theta_R D'_R \\ \frac{dM_A}{dt} = \alpha'_A D'_A + \alpha_A D_A - \delta_{M_A} M_A \\ \frac{dA}{dt} = \beta_A M_A + \theta_A D'_A + \theta_R D'_R - A(\gamma_A D_A + \gamma_R D_R + \gamma_C R + \delta_A) \\ \frac{dM_R}{dt} = \alpha'_R D'_R + \alpha_R D_R - \delta_{M_R} M_R \\ \frac{dR}{dt} = \beta_R M_R + \gamma_C AR + \delta_A C - \delta_R R \\ \frac{dC}{dt} = \gamma_C AR - \delta_A C \end{cases}$$

The activator A and repressor R are denoted by D_A and D_R, respectively. Meanwhile, the same genes bound to A and R are denoted as D'_A and D'_R, accordingly. The initial condition are$D_A = D_R = 1$,and$D'_A = D'_R = M_A = M_R = A = R = C = 0$. Beside the variables, the others are parameters with value $\alpha_A = 50, \alpha'_A = 500, \alpha_R = 0.01, \alpha'_R = 50, \beta_A = 50, \beta_R = 5, \delta_{M_A} = 10, \delta_{M_R} = 0.5, \delta_A = 1, \delta_R = 0.2, \gamma_A = 1, \gamma_R = 1, \gamma_C = 2, \theta_A = 50$ and $\theta_R = 100$.

1.2 History

As a interesting fact that the very first person in history who observed and described the oscillating gene is not a scientist but one of the generals of Alexander the Great in the fourth century B.C , named Androstenes. He recorded that a tamarind tree would open its leaves during the day and close them at night.[6] A couple of hundreds years later in 1729, French geophysicist Jean-Jacques dOrtous de Mairan noted that the rhythms of a plant have a 24 hours pattern even when placed in constant darkness where the sunlight cannot be reached. This was the first indication of circadian oscillation.[7] However, the developmental mechanism regulating this process is still unkown. Extensive experiments was done by Auguste Forel and Ingeborg Belling to see whether this rhythm was due to an endogenuous clock. Ron Konopka and Seymour Benzer isolated the first clock mutant in the early 1970s and mapped the period gene , the first discovered genetic component of a circadian clock.[8]

1.3 Thesis Structure

In the rest of the thesis, theoretical background and knowledge preparation is include, which will be useful in this project. Deterministic and Stochastic simulation will be done in Section 3, In Section 4, we will discuss the result with arguments .

The reference part contains article referenced earlier in the project. The appendix contains MATLAB code used for simulation.

2 Theoretical background

2.1 Introduction to Stochastic Method

2.1.1 Monte Carlo Method

There is not a exact definition of Monte Carlo method, but it follows a specific pattern: First define a domain of possible inputs; secondly generate a random input from a probability inside the domain from step 1; thirdly perform a deterministic computation on the input; fourthly ,return to step 2 and repeat. Finally, observe the result and analysis to get the answer. It is a very basic algorithm in the field of scientific computing.

There are two important theorems which Monte Carlo method is heavily rely on: namely Law of Large Numbers(LLN) and the Central Limit Theorem(CLT)

LLN describes the average result of performing the same experiment at large number of times should be close to the expected value. As the umber of experiments increase, it trends closer.

CLT states that if a large number of observations are generated independly and the arithmetic average of the observed values is computed. Perform this procedure many times, the computed values of the average will be distributed as normal distribution. This result promise the result from Monte Carlo method is consistent.

2.1.2 Markov Process

A discrete time Markov Chain can be seen as a stochastic version of a deterministic recursion of the for $X_n = g(X_{n-1}), n \geq 1$, where g(x) is a given function.

A discrete time stochastic process X_n with discrete state space E is called a Markov Chain if it satisfies the Markov Property:

$$P(X_n = x_n | X_0 = x_0, \cdots, X_{n-1} = x_{n-1}) = P(X_n = x_n | X_{n-1} = x_{n-1}),$$

for all $X_0, \cdots, X_n \in E$ and $n \geq 1$. That property states that given the entire past of X_n, it only cares what happened immediate past.

Theorem 2.1 (Chapman-Kolmogorov Equations)

$$P_{ij}^{((M))} = \sum P_{ik} P_{kj}^{((m-1))}, i, j \in E$$

This theorem states that a move from i to j in exactly m steps is the same as a jump from i to some state k in the first step, and then a second sequence of jumps from k to j in exactly m-1 steps[4].

2.2 The Stochastic Simulation Algorithm (SSA)

At the cellular level, a more realistic approach has to consider the intrinsic stochasticity of chemical reactions[5], this can be done by transforming the reaction rates in to probability transition rates and concentrations into numbers of molecules, then we can get the so called Chemical master equation(CME). CME is a differential equations that governs the time evolution of the probability for observing the Markov chain in a given state at a given time. The CME is generally derived using the Markov Property , by writing the Chapman-Kolmogorov equation. There is no general solution to such kind of equations. Monte Carlo simulation are seen as a normal routine, to realize the reaction as Markov Chain. The idea behind that is described as following be Jose Vilar and others as following:

"To perform a random walk through the possible states of the system, which are defined by the numbers of molecules of the different reacting species. Starting from a state with given numbers of molecules, the probability of jumping to other states with different numbers of molecules can be computed from the master equation. One can pick up a state and the jumping time according to that probability distribution, consider the resulting state as a new initial state, and repeat this procedure until some final state or time is reached."

There are several algorithms to implement this. In this thesis, we use the Stochastic Simulation Algorithms , more common known as Gillespies algorithm. There are several algorithms to implement this. In this thesis, we use the Stochastic Simulation Algorithms , more common known as Gillespie algorithm , first introduced by Gillespie in 1976. Its the standard method commonly employed to simulate continuous time markov chain models. Gillespie use the algorithm successfully to simulate the time evolution of the stochastic formulation of chemical kinetics, a process which takes into account that molecules come in whole numbers as well as the inherent degree of randomness in their dynamical behavior.[9] Now days, the algorithm is used widely in the area of computational biological. Gillespie proposed two methods, the Direct method and the First reaction method, but equivalent formulations. Generally , the algorithm can be summarized as the following step:
1. Initialization: Initialize the number of molecules in the system, reactions constants and random number generators.
2. Monte-Carlo Step: Generate random numbers to determine the next reaction to occur as well as the time interval. The probability of a given reaction to be chosen is proportional to the number of substrate molecules.
3. Updatte: Increase the time step by the randomly generated time in Step2. Update the molecule count based on the reaction that occurred. 4. Iterate:Go back to Step 2 unless the number of reactants is zero or the simulation time has been exceeded.

3 Comparison

3.1 Deterministic Method

To solve the system of ODEs numerically, each variable against time should be iteratively calculated based on discretized time steps. This means that for each equations, the variable value at each step will be calculated based on that of the former step. This method is realized in Matlab as can be seen in the appendix.

The code is composed of two parts, namely, the ODE function and the simulation . The former is to represent each equation in the systmes and put them in one vector for later calculation in one of the matlab built-in ODE solving method. On the contrast, the latter is actually the code to fulfill the simulation. It firstly defines the initial values for each variable and the time span to simulate. After that, ode45s or ode15s is employed to calculate the values of each variables iteratively. Finally, the result will be plotted in one figure.

We will use both Matlab build in solver ODE15s and ODE45s to simulate the model twice as to find out which is prior for the system.

After perfoming the calculation above, several figures will be generated for us to observe and discuss.

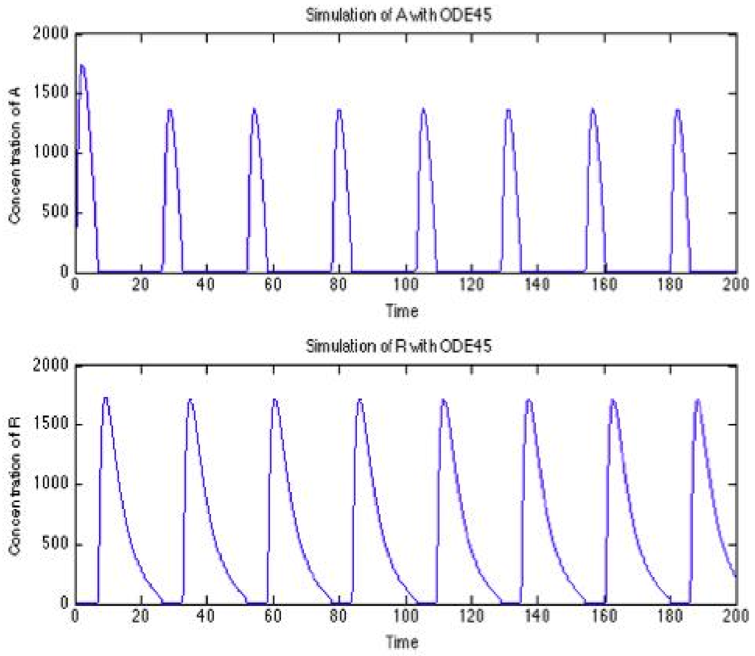


Figure 2. The simulation result of A and R within 200 hours with ode 45s

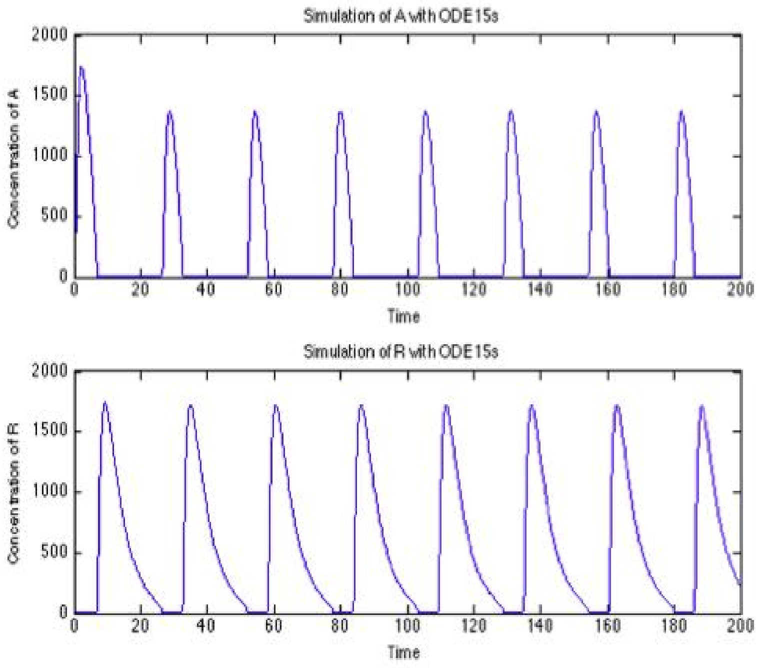


Figure 3. The simulation result of A and R within 200hours with ode15S

As can be seen from the figure, the generated results are roughly the same between ode45 and ode15s. So we can not tell which one is better ,by now. However, if we generate the CPU time spent by running each method, we can find that ode15s with 1.22s is much faster then ode45s at expense of 35.98s. Therefore, we can conclude that ode15s is superior to ode45 for this model.

To dig further why there is such big difference in time cost. We use the figure of step size against discrete point to illustrate.

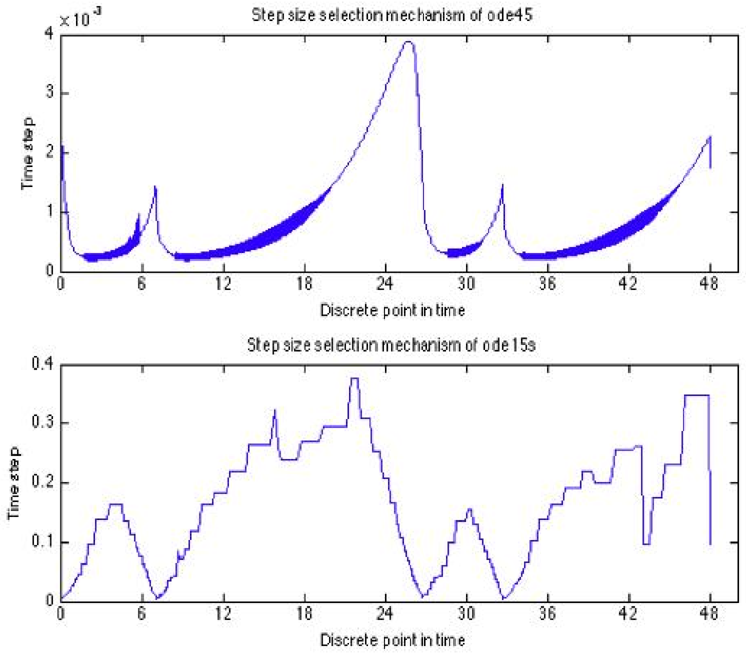


Figure 4. The figure of time step size against discrete point of time within 48 hours

As can be seen from figure 4, its quite clear that ode45s selected much smaller discretization size than ode15s. This explain the reason why ode15s is much faster than ode45. For this circadian model, the result generated by both method are very similar, ode15s beat ode45s, however, its no doubt that ode45s has much more accurate result.

3.2 Stochastic Method

The circadian model we study in this thesis can be regarded as a Contineuous Time Markov Process. The corresponding transition graph of all the possibile jump is shown as following :

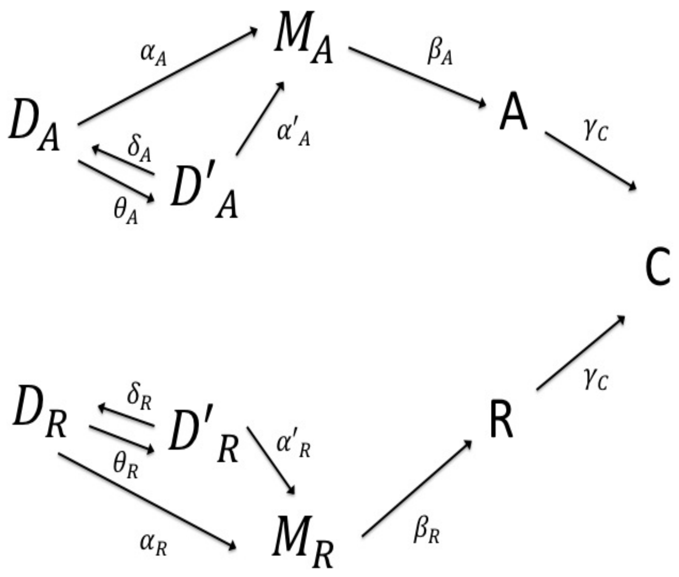


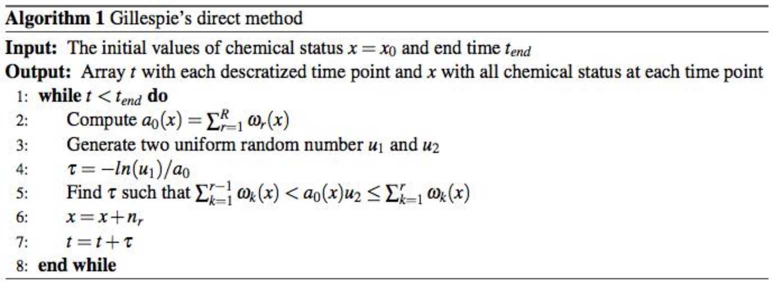
Figure 5. The transition graph of the model

And the intensity matrix describing the rate that the markov chain moves between states is :

$$
\begin{array}{c|cccccccccc}
 & D_A & D'_A & D_R & D'_R & M_A & M_R & A & R & C & \emptyset \\
\hline
D_A & -\alpha_A & 0 & 0 & 0 & \alpha_A & \cdots & \cdots & \cdots & \cdots & \cdots \\
D'_A & 0 & -\alpha'_A & 0 & 0 & \alpha'_A & \cdots & \cdots & \cdots & \cdots & \cdots \\
D_R & 0 & 0 & -\alpha_R & 0 & 0 & 0 & \alpha_R & 0 & 0 & 0 \\
D'_R & \cdots & \cdots & \cdots & -\alpha'_R & 0 & 0 & \alpha'_R & \cdots & \cdots & \cdots \\
M_A & \cdots & \cdots & \cdots & \cdots & -\beta_A & 0 & \beta_A & \cdots & \cdots & \cdots \\
M_R & \cdots & \cdots & \cdots & \cdots & \cdots & -\beta_R & \cdots & \cdots & \beta_R & 0 \\
A & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & -\gamma_C & \cdots & \cdots & \gamma_C \\
R & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & -\gamma_C & \gamma_C & 0 \\
C & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
\emptyset & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0
\end{array}
$$

Figure 6. The intensity matrix of the jump

Therefore, it has the property that each state one step in the future is solely dependent on the current status. That means status at each time step is independent. As a result, the inter event time can be sampled randomly. Moreover, which reaction should happen can also be sampled. Finally, the whole system should be updated accordingly at this time step and the time should be marching to next time step. All these steps should be placed in a loop to iteratively simulate the reactions with time going by. Gillespies algorithm we are using can be expressed as below,



Algorithm 1 Gillespie's direct method

Input: The initial values of chemical status $x = x_0$ and end time t_{end}
Output: Array t with each descratized time point and x with all chemical status at each time point

```
1: while $t < t_{end}$ do
2:     Compute $a_0(x) = \sum_{r=1}^{R} \omega_r(x)$
3:     Generate two uniform random number $u_1$ and $u_2$
4:     $\tau = -ln(u_1)/a_0$
5:     Find $\tau$ such that $\sum_{k=1}^{r-1} \omega_k(x) < a_0(x)u_2 \leq \sum_{k=1}^{r} \omega_k(x)$
6:     $x = x + n_r$
7:     $t = t + \tau$
8: end while
```

the algorithm is coded in Matlab which can be find in the appendix. We use it to simulate a 24 hours process and the reaction of the chemical R is depicted as a result.

With the stochastic model, now it is prepared to explain the strategy used to compare the stochastic method and the deterministic method. It is to test if these two methods works to simulate chemical R with the change of the value of the parameter δ_R. In this case, three values of δ_R are used for the test, namely, 0.2,0.08 and 0.01.

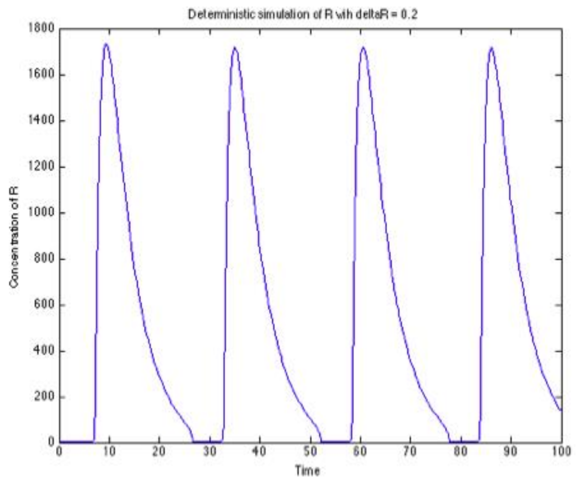


Figure 7. Deterministic simulation of R with $\delta_R = 0.2$

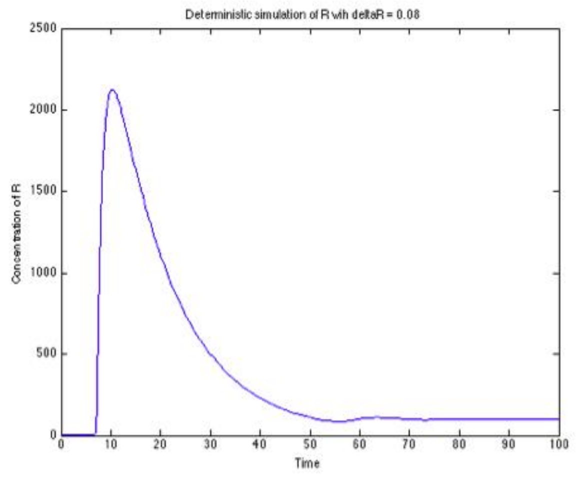


Figure 8. Deterministic simulation of R with $\delta_R = 0.08$

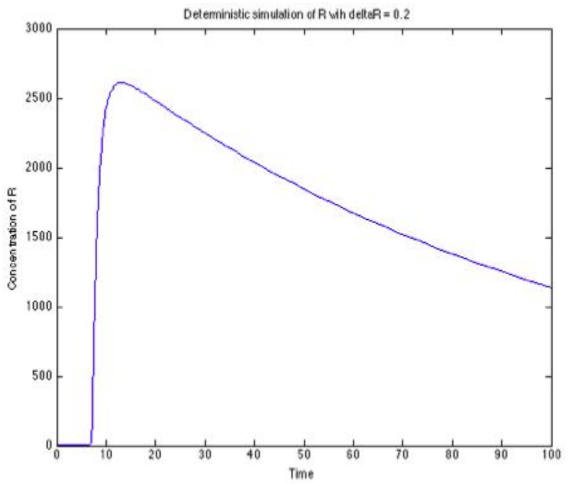


Figure 9. Deterministic simulation of R with $\delta_R = 0.01$

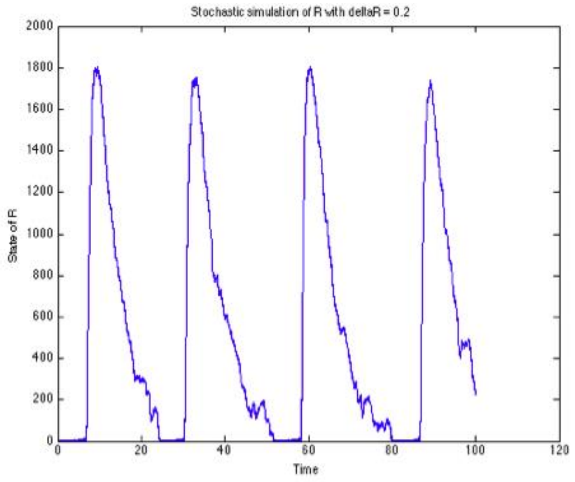


Figure 10. Stochastic Simulation of R with $\delta_R = 0.2$

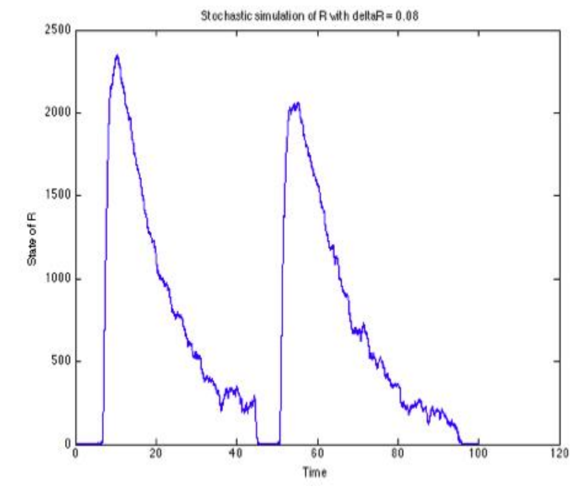


Figure 11. Stochastic Simulation of R with $\delta_R = 0.08$

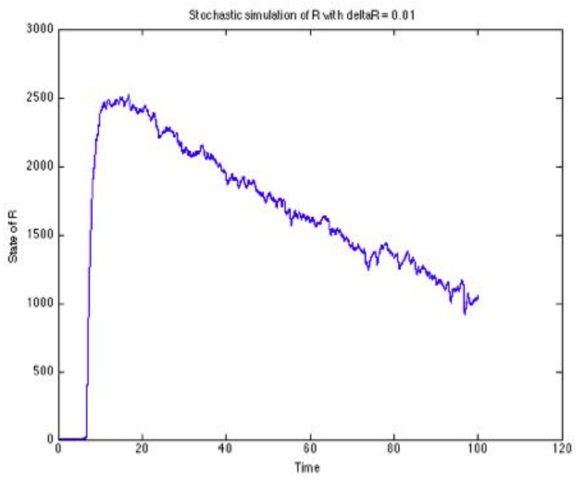


Figure 12. Stochastic Simulation of R with $\delta_R = 0.01$

4 Discussion and Conclusion

As can be seen from figure above, the clock oscillates roughly with a periof of 24 hours . However,in figure 8, the periodic characteristic disappears. With the parameter of δ_R getting smaller, say , 0.01, the periodic characteristic as in figure 9 is still missing. This means that with small values of the parameter δ_R, the deterministic method fails.

On the contrast, as can be seen in figure10, the periodic characteristic is shown with no problem for the stochastic model when $\delta_R = 0.2$, it also works fine when $\delta_R = 0.08$ as can be seen in figure 11.

However, it also need to be noticed that the stochastic method is much more time consuming than deterministic one.

In this paper we have compared different simulate method in accuracy and efficiency on the circadian clock model . The deterministic method can be useful to grasp the main character of the system with good efficiency but under certain conditions which is not known to us until we do the stochastic analysis. We need to be careful with the parameter values. Stochastic simulation gives reliable result but no one would like to use that if the model is not that complicated due to the efficiency. So when we facing choices between deterministic and stochastic method, we would use deterministic one if it applicable. The complexity of the model is also need to be taken into account. If its also many computation work for the deterministic method, we should resort the problem using stochastic algorithm.

References

[1] Vilar, Jos MG, et al. "Mechanisms of noise-resistance in genetic oscillators."Proceedings of the National Academy of Sciences 99.9 (2002): 5988-5992.

[2] Tuttle, Lisa M., et al. "Model-driven designs of an oscillating gene network."Biophysical journal 89.6 (2005): 3873-3883.

[3] Barkai, Naama, and Stanislas Leibler. "Biological rhythms: Circadian clocks limited by noise." Nature 403.6767 (2000): 267-268.

[4] Ingemar Kaj, Stochastic Markov Models. Lecture notes for the course: Markov Processes, 1MS012

[5] Van Kampen, Nicolaas Godfried. Stochastic processes in physics and chemistry. Vol. 1. Elsevier, 1992.

[6] Bretzl H. Botanische Forschungen des Alexanderzuges[M]. BG Teubner, 1903.

[7] de Mairan J. Observation botanique[J]. Hist. Acad. Roy. Sci, 1729, 35: 36.

[8] Konopka R J, Benzer S. Clock mutants of Drosophila melanogaster[J]. Proceedings of the National Academy of Sciences, 1971, 68(9): 2112-2116.

[9] Kojima, F. "Simulation Algorithms for Continuous Time Markov Chain Models."Simulation and Modeling Related to Computational Science and Robotics Technology: Proceedings of SiMCRT 2011 37 (2012): 3.

5 Appendix

1.

```
%ODE function
function y_out = circadianOde(t, y)
global theR theA;
global alphA alphR alphAPr alphRPr ;
global betA betR;
global gaA gaC gaR;
global deltMA deltA deltMR deltR;

y_out = [theA * y(3) - gaA * y(1) * y(6);
theR * y(4) - gaR * y(2) * y(6);
gaA * y(1) * y(6) - theA * y(3);
gaR * y(2) * y(6) - theR * y(4);
alphAPr * y(3) + alphA * y(1) - deltMA * y(5);
betA * y(5) + theA * y(3) + theR * y(4) - y(6) * (gaA * y(1) + gaR * y(2) +
gaC * y(8) + deltA);
alphRPr * y(4) + alphR * y(2) - deltMR * y(7);
betR * y(7) - gaC * y(6) * y(8) + deltA * y(9) - deltR * y(8);
gaC * y(6) * y(8) - deltA * y(9);
];
```

2.

```
%simulation of the circadian clock clear all;
%%% Initialize the parameters %%%
starttime = 0;
endtime = 48;
interval = [starttime endtime];
DA=1;
```

```
DR=1;
DAp=0;
DRp=0;
MA=0;
A=0;
MR=0;
R=0;
C=0;
y0 = [DA,DR,DAp,DRp,MA,A,MR,R,C];
global theR theA;
global alphA alphR alphAPr alphRPr ;
global betA betR;
global gaA gaC gaR;
global deltMA deltA deltMR deltR;
theR = 100;
theA = 50;
alphA = 50;
alphR = 0.01;
alphAPr = 500;
alphRPr = 50;
betA = 50;
betR = 5;
gaA=1;
gaR=1;
gaC=2;
deltMA = 10;
deltMR = 0.5;
deltA = 1;
deltR = 0.2;

    %%% Calculate with ode45 and ode15s %%%
values = odeset(’RelTol’, 1e-5);
cput1 = cputime;
[t1, y1] = ode45(@circadianOde, interval, y0, values);
e1 = cputime-cput1;
cput2 = cputime;
[t2, y2] = ode15s(@circadianOde, interval, y0, values);
e2 = cputime-cput2;
```

```
%%% plot A and R with ode45 %%%
figure(1);
subplot(2,1,1);
plot(t1, y1(:,6));
xlabel('Time')
ylabel('Concentration of A');
title('Simulation of A with ODE45');
subplot(2,1,2);
plot(t1, y1(:,8))
xlabel('Time')
ylabel('Concentration of R');
title('Simulation of R with ODE45');
hold on;

    %%% plot A and R with ode15s %%%

    figure(2);
subplot(2,1,1);
plot(t2, y2(:,6));
xlabel('Time')
ylabel('Concentration of A');
title('Simulation of A with ODE15s');
subplot(2,1,2);
plot(t2, y2(:,8))
xlabel('Time')
ylabel('Concentration of R');
title('Simulation of R with ODE15s');
hold on

    %%% plot stepsize against discrete point %%%
% calculate step size
stepsize1=zeros(1,numel(t1));
stepsize1(1) = t1(2);
for i = 2:length(t1)
stepsize1(i) = t1(i)-t1(i-1);
end
stepsize2 = zeros(1,numel(t2));
stepsize2(1) = t2(2);
```

```
for i = 2:length(t2)
stepsize2(i) = t2(i) - t2(i-1);
end
% plot
figure(3);
subplot(2,1,1);
plot(t1, stepsize1', 'b-');
set(gca, 'XTick', 0:6:48);
xlabel('Discrete point in time');
ylabel('Time step');
title('Step size selection mechanism of ode45');
subplot(2,1,2);
plot(t2,stepsize2', 'b-');
set(gca, 'XTick', 0:6:48);
xlabel('Discrete point in time');
ylabel('Time step');
title('Step size selection mechanism of ode15s');
hold off;
```

3. %stochastic simulation

```
clear all; t=0;
tn =100;DA=1;DR=1;DAp=0;DRp=0;MA=0;A=0;MR=0;R=0;C=0;u=[A C
DA DAp DR DRp MA MR R]; u = u(:)';
nr = nr_vilar();x = u;n = 1;whilet(n) < tn
w =r =s =a0 = s(length(w)); u1 = rand(1,1);
prop_vilar(u);length(w);cumsum(w);
u2 = rand(1,1);
tau k =u =t =x =n =end plot(t, x(:,9)', 'b-'); xlabel('Time') ylabel('State
of R');
= -log(u1)/a0; find(a0*u2¡s,1); u+nr(k,:);[t t(n)+tau];
[x;u];n + 1;
title('Stochastic simulation of R with deltR = 0.01');
```