`mcode.sty` Circadian Rhythm - A Genetic Oscillator

Alari Varmann

Scientific Bridging Course Project 3

Supervisor: Siyang Wang

UPPSALA UNIVERSITY
Department of Information Technology
2015

Circadian Rhythm Model Setup Explanation, General Conceptual Ideas

The idea of this Scientific Computing Bridging course homework simulation project is to model the changes in the counts of activator and repressor transcription factor proteins that modulate the gene expression directly by inducing a chromatin shape change by affecting the promoter area of a complementary DNA strand. The protein count dynamics, what is the topic in this miniproject, is explained in [1].
The basic idea is the feedback inhibition [2] concept in which the enzymes switch off the synthesis pathway once there is plenty of end product molecules available.

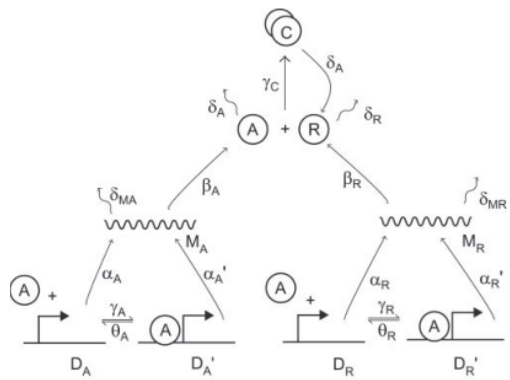
Quote from source [3]: *"The research article that is the basis for this mini project was one of their articles in systems biology to point out that the deterministic reaction rate equations did not give a sufficiently accurate description of biochemical systems."*

Let us explain the last quote.
In comparison to the circadian rhythm deterministic model developed in the last miniproject, this time we implement the stochastic model of the circadian clock biochemical reaction network. Since the cell is very small, with a volume about the magnitude in liters as the diameter of the proton in meters, there may be very few proteins, only 1 to 10. Since the small number of consituent molecules, the deterministic model based on concentrations could be very inaccurate.

This is the motivation for implementing the stochastic algorithm – called `Gillespie's algorithm`.

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

(a) Circadian Rhythm - Genetic Oscillator ODE System. Source ([1])



(b) Biochemical Network of Circadian Rhythm Oscillator Model. Source ([1])

Figure 1: The Circadian ODE System and Biochemical Reactions

Stochastic algorithm implementation

Algorithm 5 Gillespie's direct method (DM)

Initialize: Set the initial state $\mathbf{x} = \mathbf{x}_0$ and a final time T_f. Set $t = t_0$.
while $t < T_f$ **do**
  Compute $a_0(\mathbf{x}) = \sum_{r=1}^{R} \omega_r(\mathbf{x})$.
  Generate two uniform random number u_1, u_2 from $U(0,1)$.
  $\tau = -ln(u_1)/a_0$.
  Find r such that $\sum_{k=1}^{r-1} \omega_k(\mathbf{x}) < a_0(\mathbf{x})u_2 \leq \sum_{k=1}^{r} \omega_k(\mathbf{x})$.
  Update the state according to $\mathbf{x} = \mathbf{x} + \mathbf{n}_r$.
  $t = t + \tau$.
end while

Figure 2: Gillespie's Algorithm. Source: ([4]).

The main tool for the implementation is the mentioned `Gillespie's algorithm`. In this algorithm, we make repeated use of simulation a random uniformly distributed variable to find a inter reaction time τ which is distributed exponentially. We use this time to find the reaction r that occurs by requiring:

$$\sum_{k=1}^{r-1} \omega_k(\mathbf{x}) < a(\mathbf{x})u_2 \leq \sum_{k=1}^{r} \omega_k(\mathbf{x}) \Leftrightarrow \quad F(r-1) < u_2 \leq F(r), \tag{1}$$

since the cumulative distribution function is the sum of all propensity functions up to that reaction (the cumulative distribution is constant in between the discrete numbers

– reaction numbers). u_2 is the uniformly distributed random number, $a = \sum_{r=1}^{R} \omega_r(\mathbf{x})$ and ω_k are the propensity functions, which determine the probability of reaction happening by

$$p(Y = r|X = \mathbf{x}) = \omega_r(\mathbf{x})/a(\mathbf{x}) \tag{2}$$

When we've found out the next occurring reaction, we'll update the state vector of 9 different simulated molecules through

$$\mathbf{x} \xrightarrow{\omega_\mathbf{r}} \mathbf{x} + \mathbf{n}_r \tag{3}$$

(where $\mathbf{n}$ are stored in a stoichiometry matrix given by the MATLAB file `nr_vilar()`). Every row of the stoichiometry matrix is a 9-element vector of molecule number changes – basically a creation-annihilation vector and by calling $\mathbf{n}_r$ at each iteration, we get our updated state depending on the reaction type r by equation (3). Then we'll update the time by $t = t + \tau$ to get the next time. Only 1 reaction can occur at a time – that is the state space model. The propensities are proportional to the molecule counts – the more molecules, the more probable it is for the reaction to occur – this is consistent with logic as well.

The mathematical background for the stochastic algorithm is given by **continuous time discrete space Markov chain** model. In the latter, one has a finite state space – in this case the 9-dimensional vector of molecule counts, and τ is distributed exponentially. I presume that Markov chain is originally discrete, but this model is a continuous-time version of it – the output of our model is a sequence of molecule count states in the state space.
We can reduce the dimensionality of the computations by storing the molecule counts only every hour, this way we would need to compute 399×9 new molecule counts, which is less than 3600.

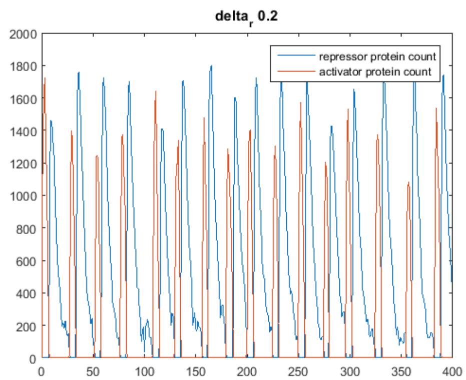
Analysis

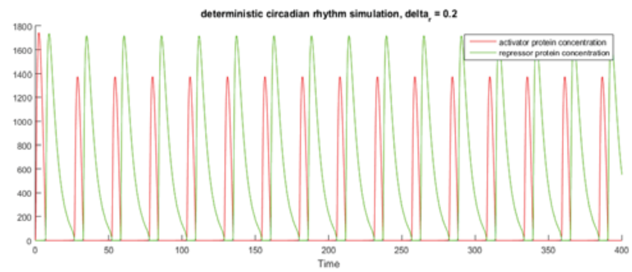
If we have propensity functions and stoichiometry vectors for all reactions, the chemical system is completely determined. So even if the model is stochastic, we can study it with deterministic methods, in this case one could use the `chemical master equation`, solve it using for example finite difference methods to obtain the probability densities and through them, the expected values of the molecule counts. We could suppose that if we have a vector – (molecule type, molecule count) as a variable – each of those combinations would be a different state in the state space. The problem is the number of molecules – if each of the molecules can have a values and there are b molecules, then the number of unknown of the deterministic linear equation system would be a^b. The unknown state would If $a = 10 = b$, then it would be 10^{10}, but if we have say 20 molecules, then 10^{20} is a huge number. In these cases, we would use **Monte carlo simulations**, for example one could implement Markov process using `stochastic simulation algorithm`, that was solved by Gillespie. Then one can make use of `inverse transform` (sampling)– for generating sample numbers at random from any probability distribution given its cumulative distribution function and a random number for example from the uniform distribution like in our case.

$$
\left.\begin{array}{rcl}
D'_a & \xrightarrow{\theta_a D'_a} & D_a \\
D_a + A & \xrightarrow{\gamma_a D_a A} & D'_a \\
D'_r & \xrightarrow{\theta_r D'_r} & D_r \\
D_r + A & \xrightarrow{\gamma_r D_r A} & D'_r
\end{array}\right\}
\left.\begin{array}{rcl}
\emptyset & \xrightarrow{\alpha'_a D'_a} & M_a \\
\emptyset & \xrightarrow{\alpha_a D_a} & M_a \\
M_a & \xrightarrow{\delta_{ma} M_a} & \emptyset
\end{array}\right\}
\left.\begin{array}{rcl}
\emptyset & \xrightarrow{\beta_a M_a} & A \\
\emptyset & \xrightarrow{\theta_a D'_a} & A \\
\emptyset & \xrightarrow{\theta_r D'_r} & A \\
A & \xrightarrow{\delta_a A} & \emptyset \\
A + R & \xrightarrow{\gamma_c A R} & C
\end{array}\right\}
$$

$$
\left.\begin{array}{rcl}
\emptyset & \xrightarrow{\alpha'_r D'_r} & M_r \\
\emptyset & \xrightarrow{\alpha_r D_r} & M_r \\
M_r & \xrightarrow{\delta_{mr} M_r} & \emptyset
\end{array}\right\}
\left.\begin{array}{rcl}
\emptyset & \xrightarrow{\beta_r M_r} & R \\
R & \xrightarrow{\delta_r R} & \emptyset \\
C & \xrightarrow{\delta_a C} & R
\end{array}\right\}
\tag{20}
$$

Figure 3: The reactions in the stochastic circadian rhythm network. Source ([4]).

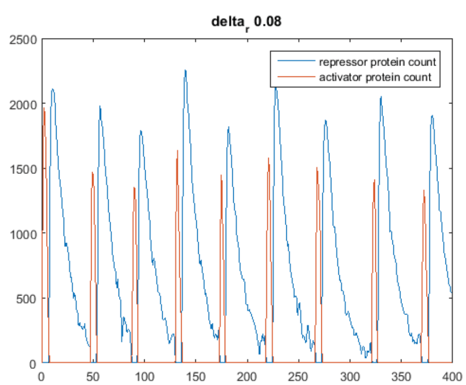


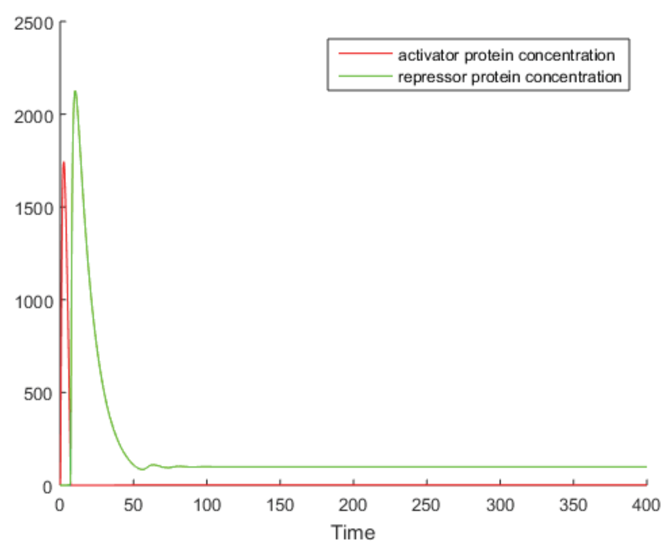


(b) Deterministic model from differential equations for $\delta_r = 0.2$

(a) Stochastic model of circadian rhythm for $\delta_r = 0.2$

Figure 4: Comparison of deterministic and stochastic models. We see that for $\delta_r = 0.2$, both the stochastic and deterministic clocks show cyclic period oscillations, with period $T = 24$ approximately hours. We see that if the propensity for the annihilation of repressor protein is high then then the decrease in R is very sharp

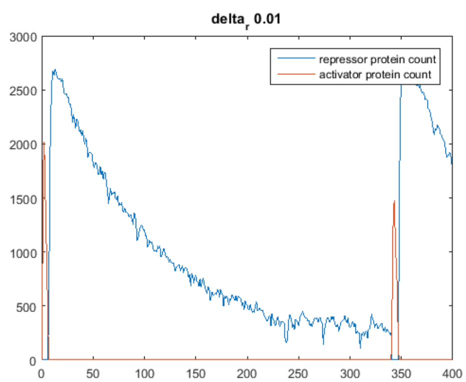




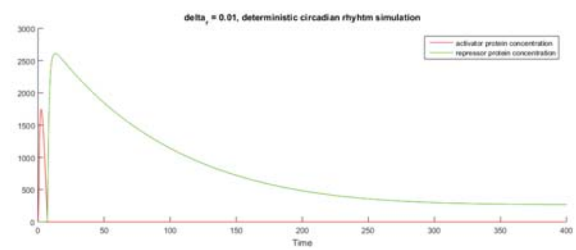
(a) Stochastic model of circadian rhythm for $\delta_r = 0.08$

(b) Deterministic model from differential equations for $\delta_r = 0.08$

Figure 5: We see a major difference in the behaviour of the stochastic and deterministic clocks. That demonstrates the stochastic nature of our simulations



(a) Stochastic model of circadian rhythm for $\delta_r = 0.01$



(b) Deterministic model from differential equations for $\delta_r = 0.01$

Figure 6: Again we see that the stochastic clock doesn't switch itself off in contrast to the deterministic clock, which is insensitive to noise since the fixed point at extremal δ_r values becomes stable, the oscillation period seems to have extended to about 350 hours.. I am sure the cells would be dead by that time already, but at some low value of δ_r, R count should not decay anymore. The question is — why doesn't my R rate decrease slower with really small δ_r ? I think that depends on the randomness of my simulations

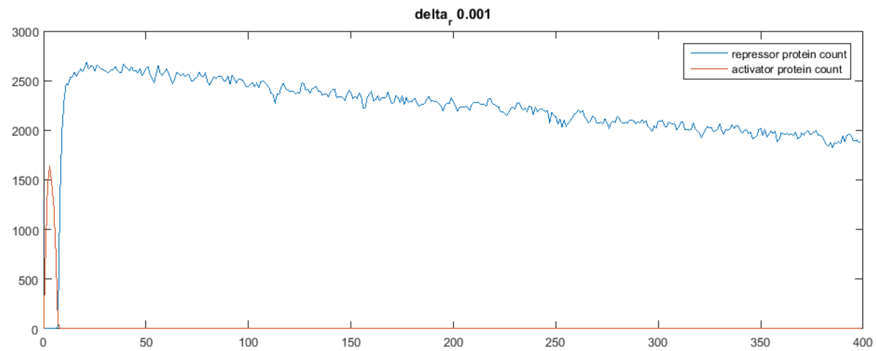


Figure 7: We see that if δ_r is very small, then the repressor protein is annihilated very slowly, thus and since there is a lot of repression transcription factor. Thus, not enough proteins are produced and the cycle is downregulated so that the circadian rhythm is not established – and thus, the cell could not produce day-night type of oscillations .

Since any organism is made out of cells and if the cells are not able to produce stable day-night oscillations, then the organism would not sustain any form of life, since I think it would not be able to regenerate itself.
Simple comparison: Yesterday I was really tired and it felt so hard to do useful work and my efficiency was certainly small. Today, after having properly rested, I feel much better and can do useful work much more efficiently. This is just the generalization from the cellular level.

Conclusion analysis

Analysis of Oscillations

It is interesting to note, that mRNA molecule count does not directly enter the dynamics (if the protein count remains constant).
`In the deterministic case, the oscillations are always regular`, but in the `stochastic case the additional parameters may alter the oscillation cycle` (δ_r in our case).
`In the stochastic case, the fixed point may not become stable` [1] for extermal values of δ_r . So this means that the stochastic network may still continue oscillating, although the propensity for the degradation of R protein could be really small, and it looks like its quantity shouldn't change a lot. That's exactly what happened in my simulations – the oscillations still continued down to a very small value of $\delta_r < 0.01$.

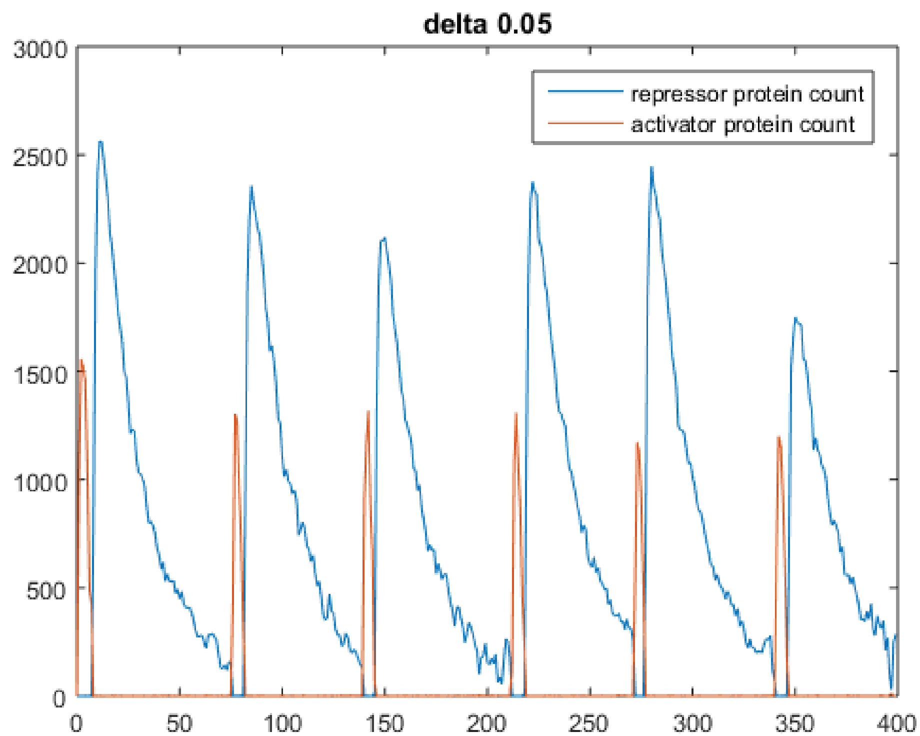


Figure 8: The simulation for $\delta_r = 0.05$ as done in the article ([1]). I get approximately the same chart, but still different on each run. Whereas in the case of ([4]) the oscillations ceased at $\delta_r = 0.08$, in ([1]) and in my case they are produced nevertheless. Account to the realistic situation: If the period of next activator/repressor transcription factor production is very high, the cells would not be able to implement that efficiently since the day on Earth lasts 23 hours and 56 minutes – and probably die in the long run nevertheless because of the increase of entropy of communication in the system .

Analysis of Stochastic vs Deterministic Algorithm Usage Feasibility

If a deterministic model for a problem is available and feasible (especially in low dimensions), one should use this instead of the stochastic model.
When the number of molecules is high, the stochastic algorithm is comparatively a lot faster than the deterministic algorithm. If I understand the situation correctly, then in our algorithm we "discretize" the time variable for faster execution time purposes, like in this particular case, then in the case of n timesteps, a molecules and b values for each, the dimension of the system would be $\mathcal{O}(a \times b \times n)$ – and the fact that we have Markov chain, would reduce the system dimension even more for our algorithm since the next step depends only on the last step, so this should even limit the dimension more. The only difference is that since we don't actually discretize the time variable, but only store discrete points of time, the actual dimension of the system is a lot higher – and the higher the dimension of the system, the more feasible it would be to use Monte Carlo methods to solve it. So we basically get an efficient algorithm for a very high dimensional system, by using Monte Carlo and then artificially even lower the dimension a little bit to get faster execution time [1]

Final Conclusive Remarks and Explanation of the Use of Stochasticity for Adaptivity and Intelligence

In general I would say that the results from the stochastic simulations are more unpredictable (another way – versatile for the organism). For example, in ([4], the circadian clock switches itself off for $\delta_r = 0.08$, but in ([1]) , the circadian clock for $\delta_r = 0.05$ produces oscillations nevertheless. In my case, I had oscillations even when $\delta_r = 0.01$. I think this exemplifies the stochastic nature of our algorithm well. I think what could be said is that *the stochastic simulation algorithm is an efficient way to tackle a problem in high dimensional state space, but the algorithm should be run many times to get a picture of what is going on – this way, we could explore the adaptivity phenomenon.*

Usage of stochasticity in cellular biochemical networks: The biochemical networks of organisms may actually take advantage of the cellular noise ([1]) to increase the amount of possible future actions [2](states) and maintain adaptivity for changing environmental conditions.

[1] the dimension actually remains the same, but since we don't store all the information at every step, we can say that the dimension is "lower"

[2] this is one way to define intelligence

References

[1] Vilar, J.M.G. , Kueh, H.Y. , N. Barkai, Leibler, S. Mechanisms of noise-resistance in genetic oscillators

[2] Gene Expression [WWW] http://highered.mheducation.com/olcweb/cgi/pluginpop.cgi?it=swf::535::535::/sites/dl/free/0072437316/120070/bio10.swf::Feedback%20Inhibition%20of%20Biochemical%20Pathways

[3] Mini Project 3 handout: "Circadian Rhythms and Stochastic Models"

[4] Hellander, A. Stochastic Simulation and Monte Carlo Methods

Appendix : MATLAB Code for the Solutions

```
clear all; clc; close all;
%% CREATION OF THE PARAMETER VECTOR
alpha_a = 50;
alpha_pr_a = 500;
alpha_r = 0.01;
alpha_pr_r = 50 ;
beta_a = 50;
beta_r = 5;
Δ_m_a = 10;
Δ_m_r = 0.5;
Δ_a = 1;

Δ_r= 0.05; % = 0.2 = 0.01 % change this parameter for stochastic



gamma_a = 1;
gamma_r = 1;
gamma_c = 2;
theta_a = 50;
theta_r = 100;

    p = [alpha_a alpha_pr_a alpha_r alpha_pr_r beta_a beta_r theta_a
        theta_r gamma_a gamma_r gamma_c Δ_m_r Δ_m_a Δ_a Δ_r];


    %% CREATION of input vector
    A = 0;
    C =0;
    D_A = 1; % only activator and repressor gene different from 0
    D_Apr = 0;
    D_R = 1;
    D_Rpr =0;
    M_A =0;
    M_R = 0;
```

```
    R =0;

    x=1:9;
    count(x) = [A C D_A D_Apr D_R D_Rpr M_A M_R R]; %initial state --- DISCRETE NUMBER OF MOLECULES!


    t = 0;
    T = 400;
    nr = nr_vilar();
    statemat = zeros(400,9); timecol = zeros(400,1); counter = 0;
    running_index = 1;
    %%
    while t < T
        w = prop_vilar(count(x), p); % creates the 18-dim propensity vector omega dependent of molecule count nr(x)
        u1 = rand; u2 = rand; % create 2 uniformly distributed random numbers

        a = sum(w); % now this is like the CDF / unit time (cumulative distribution function per unit time)
        tau = -log(u1)/a; % find exponentially distributed tau such that it gives a -- the uniform propensitysum

        % tau is the inter event time -- time until the next reaction occurs
        % find which reaction occurs

        r = next_reaction(w,u2);

        count(x) = count(x) + nr(r,:); % r'th reaction transformation, change molecule vector state

        t_integer = 0:1:400;
        check = ones(1,400);
        if (t > t_integer(running_index)) && (check(running_index))
            statemat(running_index,:) = count(x);
            timecol(running_index) = t;
            check(running_index) = 0;
            running_index = running_index + 1;
        end
         % update the state
        t = t+tau;
        counter = counter +1;
        count(x)
        %plot(timecol,statemat(:,1), timecol, statemat(:,9)); drawnow;

    end
    figure();
    plot(timecol,statemat(:,9), timecol, statemat(:,1));
    legend('repressor protein count', 'activator protein count');
    title(['Δ ', num2str(Δ_r)]);
```

```
function r = next_reaction(w,u2)
  k = cumsum(w(1:end));
  % cumk = k(1:end-1); cuml = k(2:end);
    a = sum(w); % cumulative distribution function coefficient

```

```
    rvec = find(a*u2 ≤ k);

r = rvec(1); % finds the first index for which it is true, then it is automatically bigger than the last sum
end
```