Miniprojekt 2

Beräkningsvetenskap II

Per Engström & Nathalie Proos Vedin

21 februari 2015

Innehåll

1 Inledning

För att anpassa sig till dygnets cykliska beteende använder sig många organismer av en så kallad cirkadisk rytm, vilket är en biologisk klocka med en period på 24 timmar. I en förenklad modell över hur denna klocka fungerar kan rytmen sägas bero på två specifika, reglerande proteiner, ett som undertrycker (repressor, i rapporten nämnd protein R) och ett som aktiverar (activator, protein A) relevanta processer. Undersökningar visar att oscillatorns beteende till största del beror på två faktorer: koncentrationen av protein R och molekylärdynamiken i processen då protein A bildar ett inaktivt komplex med R [1].

Syftet med detta miniprojekt var att utnyttja *Gillespies algoritm* på det studerade systemet för att avgöra hur den stokastiska metodens resultat skiljer sig från den tidigare använda [2] deterministiska metoden. Resultaten för två olika värden på nedbrytningshastigheten för protein R skulle jämföras och allmänna skillnader mellan de både metoderna skulle studeras.

2 Matematisk modell

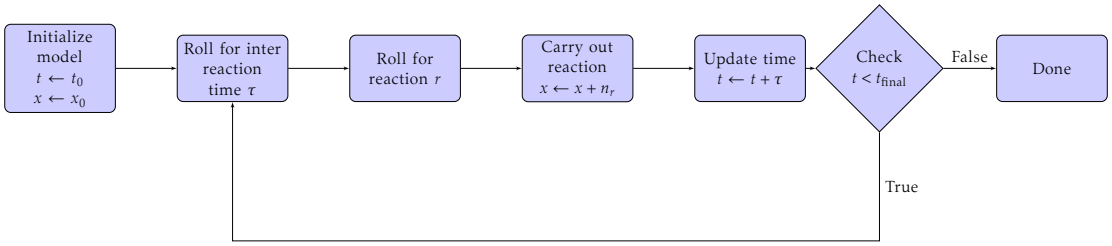
Den cirkadiska rytmen utgörs av 18 reaktioner som beskrivs i ekvation 20 (sidan 21) i [3]. Varje reaktion r har en tillhörande propensitet w_r som beskriver hur frekvent reaktionen är. Varje reaktion kan beskrivas algebraiskt som en differens n_r som beskriver hur varje partikel förbrukas eller produceras.

Propensiteterna w_r är funktioner av tillståndet x. Givet w_r för varje reaktion kan vi tala om den *totala propensiteten* β som är summan av varje enskild reaktions propensitet. Då kommer väntetiden mellan två reaktioner τ vara exponentialfördelad med parameter β.

Om vi jämför med MINIPROJEKT 1 så har vi övergått från koncentration av ämnen och deras till en diskret modell där varje reaktion har en viss sannolikhet att inträffa. Sannolikheten att reaktionen r sker i intervallet $[t, t+\tau]$ är $w_r\beta^{-1}$.

3 Numeriska metoder

Till skillnad från MINIPROJEKT 1 där vi med ordinära differentialekvationer arbetade deterministiskt, använde vi nu en stokastisk metod. Detta krävde dock en omskrivning av den matematiska modellen (se avsnitt 2).



Figur 1: Schematisk beskrivning av *Gillespies algoritm*. I varje steg slumpas ett tidssteg $\tau \sim \exp(\beta)$ och en reaktion r viktat efter deras propensiteter. Slutligen uppdateras tiden t och tillståndet x.

Den ursprungliga kontinuerliga och deterministiska metoden baserades på lösning av differentialekvationer som beskrev koncentrationen av de intressanta partiklarna. Detta antar att antalet partiklar är stort så att koncentrationen är approximativt kontinuerlig. I en cell är dock det antagandet svårt att motivera, ty antalet av varje partikel kan finnas i endast ett fåtal exemplar.

Vi använde istället den diskreta och stokastiska *Gillespies algoritm*. En beskrivning av algoritmen ses i figur 1. Nu arbetar vi direkt med partiklars antal istället för koncentrationer, och istället för reaktionshastigheter har vi sannolikheter (propensiteter) för varje reaktion.

3.1 Hitta tiden till nästa reaktion

Först initierar vi vårt tillstånd med initialvärden för tiden t (`t`) och tillståndet x (`x`) enligt avsnitt 4. Sedan beräknar vi propensiteterna w_r enligt `w = prop_vilar(x, b)` där `b` är en parametervektor enligt avsnitt 4 och `w` är en 18×1-kolumnvektor. Vi kan då beräkna summan β som `beta = sum(w)`.

Sedan slumpar vi fram τ med hjälp av att invertera den kumulativa sannolikhetsfördelningen. Detta görs med ett slumptal $u_1 \sim U[0,1]$ enligt

$$\tau = \frac{-\ln u_1}{\beta}. \tag{1}$$

Detta implementerar vi i matlab som

```
u1 = rand;
tau = -log(u1)/beta;
```

3.2 Hitta nästa reaktion

Nästa steg är att uppdatera tillståndet, vilket betyder att vi måste ta reda på vilken reaktion som sker. Vi kan ta reda på sannolikheten för varje reaktion genom att dela propensitetsvektorn med den totala propensiteten `w_norm = w./beta`.

Vi beräknar den kumulativa sannolikhestfördelningen (CDF) som

```
w_cumsum = cumsum(w);
w_cdf = [0; w_cumsum(w)];
```

Prefix-nollan är nödvändig för att varje reaktion skall företrädas av ett intervall. Eftersom summan är kumulativ kommer intervallet mellan två reaktioner vara sannolikheten för den senare reaktionen. Vi kan då ta ett likformigt slumptal $u_2 \sim U[0,1]$ och hitta intervallet det faller i. Detta implementerar vi som

```
jfr = u2 < w_cdf;
j = find(jfr);
next = j(1)-1;
```

där olikheten utförs elementvis och ger en vektor av resultatet. Vi vet att det är falskt tills intervallet som u_2 faller i, och sant efteråt. Vi vill då hitta det första sanna elementet, och använder `find` som returnerar de nollskiljda (sanna) elementen. Första elementet i `j` är då den senare randpunkten av intervallet, vilket är reaktionen vi väljer. Dock måste vi kompensera för prefix-nollan och subtraherar då 1 från resultatet.

Nu är det bara att uppdatera tillståndet genom att inkrementera x med stökiometrivektorn n_r för den valda reaktionen `x = x + nr(next,:)`. Här är `nr` vår stökiometrimatris där varje rad svarar mot en reaktion.

3.3 Avrunda lösningen i ett temporalt "grid"

För att kontrollera mängden datapunkter använder vi `linspace` för att definiera en tidsvektor `T` med tider vi är intresserade av. Vi initierar även lika stora vektorer för A och R.

Under simulationen kontrollerar vi om vi har passerat en intressant tidpunkt och sparar resultaten i vektorerna för A och R. Sedan inkrementerar vi index och väntar tills den passerar nästa tidpunkt av intresse. Med andra ord fortgår simuleringen som vanligt, men vi gör specifika urplock av data.

4 Indata

Initialvärden och parametrar för systemet ges i tabell 1 och 2.

D_A	D_R	D'_A	D'_R	M_A	A	M_R	R	C
1	1	0	0	0	0	0	0	0

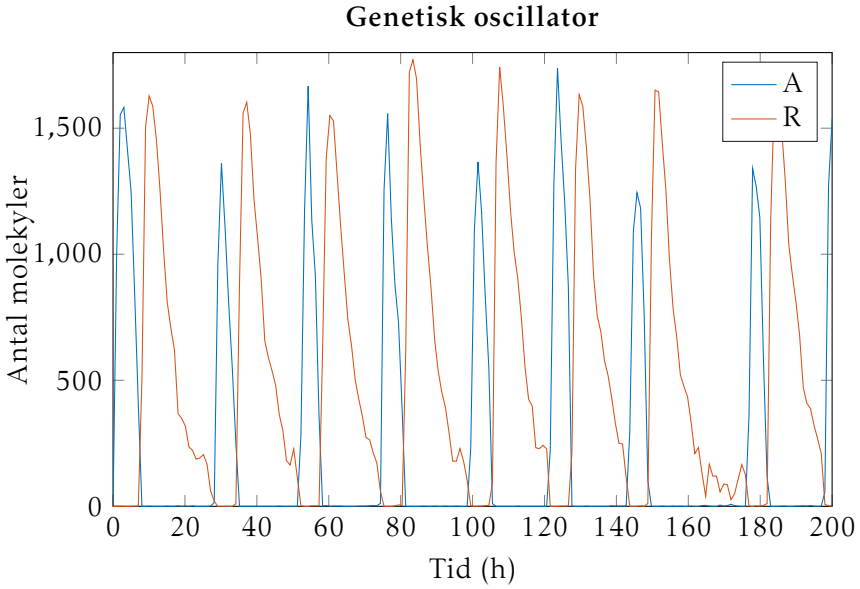
Tabell 1: Initialvärden för systemet. Enheten är mol. Värdenas biologiska betydelse finns beskriven i figurtexten till figur 1 i artikeln skriven av Vilar *et al*. [1].

α_A	α'_A	α_R	α'_R	β_A	β_R	δ_{M_A}	δ_{M_R}	δ_A	δ_R	γ_A	γ_R	γ_C	θ_A	θ_R
50	500	0.01	50	50	5	10	0.5	1	0.2 resp. 0.08	1	1	2	50	100

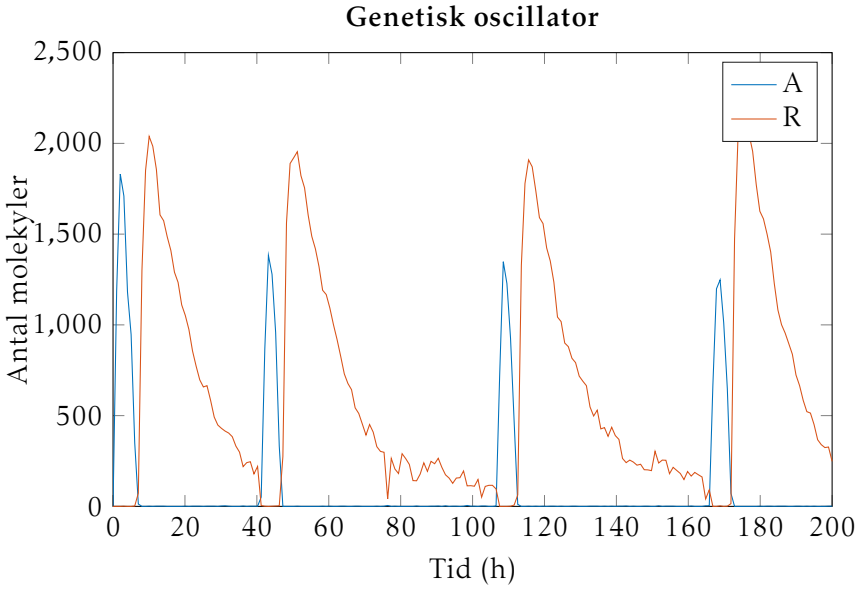
Tabell 2: Parametrar för systemet. Alla parametrar har enheterna h^{-1} förutom γ som har enheten $mol^{-1}h^{-1}$. Parametrarnas biologiska betydelse finns beskriven i figurtexten till figur 1 i artikeln skriven av Vilar *et al*. [1].

5 Resultat

Kod till samtliga script finns att skåda i sin helhet i Appendixet, eller på `https://github.com/legopelle/bervet2-miniproj2`.



Figur 2: Graf över hur de studerade proteinernas mängder varierade över tiden, med parameter- och initialvärden givna i avsnitt 4 och för $\delta_R = 0,2\ \mathrm{h}^{-1}$.



Figur 3: Graf över hur de studerade proteinernas mängder varierade över tiden, med parameter- och initialvärden givna i avsnitt 4 och för $\delta_R = 0,08\ \mathrm{h}^{-1}$.

Figur 2 visar hur mängden av de båda proteinerna varierar under 200 timmar då has-

tigheten för den spontana nedbrytningen av protein R (δ_R) är 0.2 h^{-1}. Figur 3 visar detsamma fast med $\delta_R = 0.08$ h^{-1}.

6 Diskussion

Om vi undersöker topparna i intervallet $t \in [0, 200h]$ så kan vi uppskatta periodtiden. För $\delta_R = 0.2$ ger detta ungefär 25h och för $\delta_R = 0.08$ ungefär 50h. Detta är rimligt då δ_R är proportionell mot R-proteinets nedbrytning. Om den bryts ner mer sällan avtar den långsammare och perioderna blir längre.

I MINIPROJEKT 1 mättes D_A och D_R med koncentrationer (egentligen antal i enhetsvolymen), medan vi nu mätte direkt i antal. Koncentrationen antogs vara kontinuerlig så att ett differentialekvationssystem var rimligt. I detta projekt arbetade vi istället med diskreta antal och sannolikheter för enskilda reaktioner.

Referenser

[1] Vilar, José M.G., Kueh, Hao Yuan, Barkai, Naama och Leibler, Stanislas. 2002. Mechanisms of noise-resistance in genetic oscillator. *Proceedings of the National Academy of Sciences of the United States of America* 99 (9): 5988-5992. doi: 10.1073/pnas.082111899

[2] Proos Vedin, Nathalie och Engström, Per. 2015. *Genetisk oscillator*. Uppsala universitet.

[3] Hellander, Andreas, *Stochastic simulaton and monte carlo methods*, February 23, 2009

```
function [tau, next] = next_reaction (x,prop)

w = prop(x);

    %Frekvens
    beta = sum(w);

    %Inter event time
    u1 = rand;
    tau = -log(u1)/beta;

%Nästa rxn
    w_norm = w./beta; %Normalisering av w
    u2 = rand; %Likformigt slumptal

    % find corresponding reaction
    jfr = [0;cumsum(w_norm)];
    g = u2 < jfr;
    j = find(g);

    next = j(1) - 1;
```

```
function b = parameters_vilar ( )

b = zeros(15,1);

%Parametrar (alla har enhet h^(-1), utom gammavariablerna som
%istället har mol^(-1) h^(-1) )
alphaA = 50;
alphapA = 500;
alphaR = 0.01;
alphapR = 50;
betaA = 50;
betaR = 5;
deltaMA = 10;
deltaMR = 0.5;
deltaA = 1;
deltaR = 0.2;
%deltaR = 0.08;
gammaA = 1;
gammaR = 1;
gammaC = 2;
thetaA = 50;
thetaR = 100;

%En allokerad vektor som innehåller problemets samtliga
%parametrar, samt elementens betydelser
b(1) = alphaA;
b(2) = alphapA;
b(3) = alphaR;
b(4) = alphapR;
b(5) = betaA;
b(6) = betaR;
```

```
b(7) = thetaA;
b(8) = thetaR;
b(9) = gammaA;
b(10) = gammaR;
b(11) = gammaC;
b(12) = deltaMR;
b(13) = deltaMA;
b(14) = deltaA;
b(15) = deltaR;
```

```
%Kommandofil för den stokastiska simuleringen av den cirkadiska rytmen.

clear all;
close all;

%Initialdata
t0 = 0;
tf = 200;
element = 200;

b = parameters_vilar(); %Parametrar
nr = nr_vilar(); %Stökiometrimatris
prop = @(x) prop_vilar(x,b); %Propensitetsfunktion

%Begynnelseantal
DA = 1;
DR = 1;
DpA = 0;
DpR = 0;
MA = 0;
A = 0;
MR = 0;
R = 0;
C = 0;

x0 = [A;C;DA;DpA;DR;DpR;MA;MR;R]';

%Start
t = t0; %Time
x = x0; %State

T = linspace(t0,tf,element); %time grid
A = zeros(element,1); %Allocate A grid
R = zeros(element,1); %and R grid
A(1) = x0(1); % First A grid point
R(1) = x0(9); % First R grid point
i = 2; % Next index of next grid point

%Algoritm

while t < tf; %Simulate until final time

    %Propensiteter
```

```
    [tau, next] = next_reaction(x, prop); %Next reaction

    if t > T(i);
        A(i) = x(1);
        R(i) = x(9);
        i = i + 1;
    end

    %Uppdatera systemet
    x = x + nr(next,:); %Uppdatera state
    t = t + tau;%uppdaetra tid

end

%Efter loopen måste sista värdena sättas maanuellt
A(end) = x(1);
R(end) = x(9);

%Plot
plot(T, A, T, R);
title('Genetisk oscillator')
xlabel('Tid (h)')
ylabel('Antal molekyler')
legend('A','R')
```

```
% Stochiometry matrix for the vilar oscillator.
%
% Andreas Hellander, 2009.
%

function nr = nr_vilar()

nr=zeros(18,9);

nr(1,:) =[ 0  0  1 -1  0  0  0  0  0];
nr(2,:) =[-1  0 -1  1  0  0  0  0  0];
nr(3,:) =[ 0  0  0  0  1 -1  0  0  0];
nr(4,:) =[-1  0  0  0 -1  1  0  0  0];
nr(5,:) =[ 0  0  0  0  0  0  0  1  0];
nr(6,:) =[ 0  0  0  0  0  0  0  1  0];
nr(7,:) =[ 0  0  0  0  0  0  0 -1  0];
nr(8,:) =[ 0  0  0  0  0  0  1  0  0];
nr(9,:) =[ 0  0  0  0  0  0  1  0  0];
nr(10,:)=[ 0  0  0  0  0  0 -1  0  0];
nr(11,:)=[ 0  0  0  0  0  0  0  0  1];
nr(12,:)=[ 0  0  0  0  0  0  0  0 -1];
nr(13,:)=[ 0 -1  0  0  0  0  0  0  1];
nr(14,:)=[ 1  0  0  0  0  0  0  0  0];
nr(15,:)=[ 1  0  0  0  0  0  0  0  0];
nr(16,:)=[ 1  0  0  0  0  0  0  0  0];
nr(17,:)=[-1  0  0  0  0  0  0  0  0];
nr(18,:)=[-1  1  0  0  0  0  0  0 -1];
```

```
function w = prop_vilar(u, p)
%
% w = prop_vilar(u, p)
% Propensities, w, for the Vilar oscillator.
%
% Input: u - the current state.
%        p - list of parameters
%
% The current state variables (u) are ordered as:
%    A  C  D_A  D_A'  D_R  D_R'  M_A  M_R  R
% The parameters (in p) are ordered as:
%   alfa_A  alfa'_A  alfa_R  alfa'_R  beta_A  beta_R  teta_A  teta_R ...
%   gamma_A  gamma_R  gamma_C  delta_M_R  delta_M_A  delta_A  delta_R
%

alfaA = p(1); alfapA = p(2);
alfaR = p(3);  alfapR = p(4);
betaA = p(5);  betaR = p(6);
tetaA = p(7);  tetaR = p(8);
gammaA = p(9); gammaR = p(10); gammaC = p(11);
deltaMR = p(12);  deltaMA = p(13); deltaA = p(14);
deltaR = p(15);

w = zeros(18,1);
w(1 ) =   tetaA*u(4);
w(2 ) =  gammaA*u(1)*u(3);
w(3 ) =   tetaR*u(6);
w(4 ) =  gammaR*u(5)*u(1);
w(5 ) =  alfapR*u(6);
w(6 ) =   alfaR*u(5);
w(7 ) = deltaMR*u(8);
w(8 ) =  alfapA*u(4);
w(9 ) =   alfaA*u(3);
w(10) = deltaMA*u(7);
w(11) =   betaR*u(8);
w(12) =  deltaR*u(9);
w(13) =  deltaA*u(2);
w(14) =   betaA*u(7);
w(15) =   tetaA*u(4);
w(16) =   tetaR*u(6);
w(17) =  deltaA*u(1);
w(18) =  gammaC*u(1)*u(9);
```

```
function [tau, next] = next_reaction (x,prop)

w = prop(x);

    %Frekvens
    beta = sum(w);

    %Inter event time
    u1 = rand;
    tau = -log(u1)/beta;
```

```
%Nästa rxn
    w_norm = w./beta; %Normalisering av w
    u2 = rand; %Likformigt slumptal

    % find corresponding reaction
    jfr = [0;cumsum(w_norm)];
    g = u2 < jfr;
    j = find(g);

    next = j(1) - 1;
```