MODELLING CELLULAR PROCESSES IN SPACE AND TIME
10-17 OCTOBER 2015
A PRIMER ON STOCHASTIC MODELS OF BIOCHEMICAL REACTIONS

KARIN SASAKI (EMBL CENTRE FOR BIOLOGICAL MODELLING)

Contents

1. Preliminaries

This course introduces stochastic models as a mathematical method to study biochemical reactions. It also makes an explicit comparison with deterministic models. We look at one example of regulation of gene expression.

Margin note: 'Math' indicates a mathematics tool. 'Comp', which indicates a computational tool.

Exercises can be found in section 3.

2. Stochasticity

2.1. **Motivation for considering stochasticity when studying biochemical reactions.**

Biological systems are successful despite existing in a **stochastic environment** and despite the **probabilistic nature of the biochemical reactions**. In fact, the importance of **noise** in some biological systems (e.g. neural, genetic, and metabolic networks) has long been recognised and unambiguously measured (e.g. using fluorescent reporters).

Noise can have various effects on the dynamics of the system. For example, 'nocuous' noise-induced variability in cells is responsible for population heterogeneity, phenotypic variations and imprecision in biological clocks. On the other hand, cells can take advantage of the noise and respond with 'beneficial' noise-induced behaviours, such as tuning of a response (sensitivity of the signal), stochastic resonance (amplification of the response), noise-induced oscillations, noise-induced synchronisation, noise-induced excitability, or noise-induced bistability.

To unravel the intricate interplay between noise with determinism in these systems, scientists are guided by a number of theoretical, computational, and experimental tools. In this course, we will concentrate on some mathematical and computational approaches that can be used to demonstrate that stochastic effects alter cellular phenotypes and how.

2.2. **Noise.**

Most biological systems are complex processes consisting of many biochemical reactions that are potentially significantly stochastic: reacting molecules come together by diffusion, their motion being driven by random collisions with other molecules. Such collisions randomly alter the internal energies of the reactants and thereby their propensity to react (that is, the **probability** that an encounter is a successful reaction). This type of effects are only important when mean numbers of molecules are low; then, individual reactions, which at most change the numbers of molecules by one or two, matter. Such stochasticity is referred to as **intrinsic noise** and it is inherent to the dynamics of any biochemical system. When the number of molecules is high, however, the fluctuations are averaged out and the molecular noise can be neglected (see [3] for an introduction to modelling this types of systems).

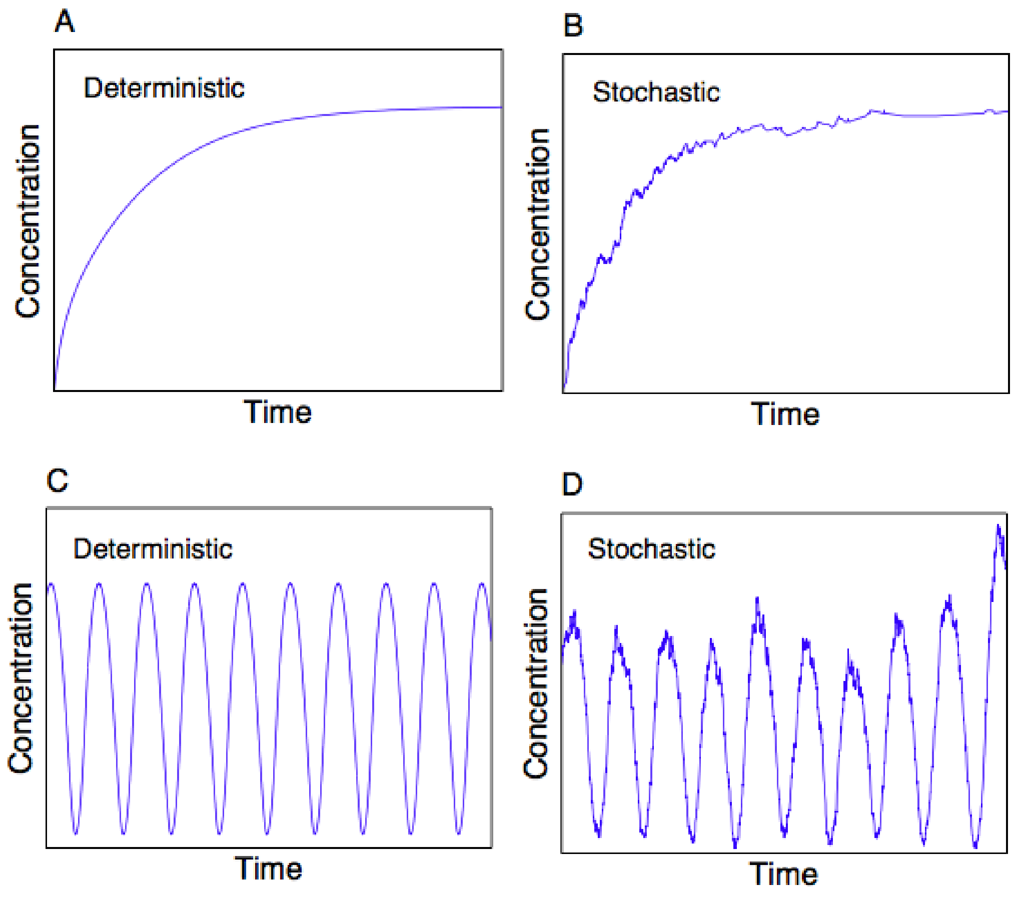
Extrinsic noise is due to the random fluctuations in environmental parameters, e.g. as the cell grows, the number of ribosomes and their variance can change, altering the noise, e.g. in gene expression; fluctuations in the numbers of nutrients in the extracellular environment; temperature; the number of amino acids available intracellularly; pH; kinetic parameters...

Here we will concentrate on intrinsic noise only.

2.3. **Deterministic vs Stochastic approaches.** **Deterministic models** includes several classes of models, whose, the most usual, is represented by systems of *ordinary differential equations (ODE)*. For an introduction to this type of modelling, see the course in [3]. In these approaches the behaviour of the model is perfectly **predictable**, from the initial conditions.

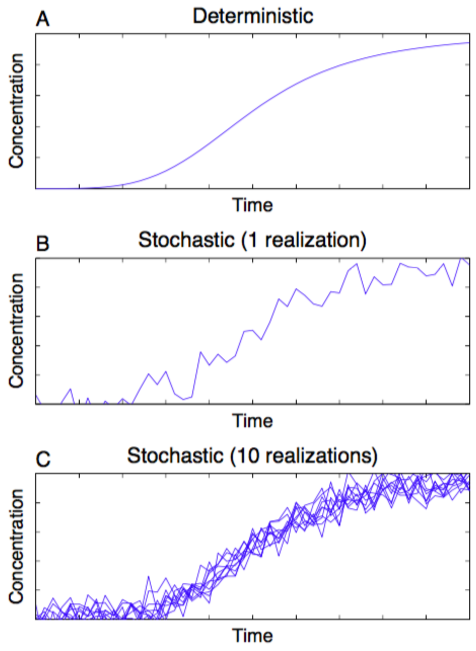
In **stochastic models**, the **probabilistic aspects**, alluded to above, are taken into account. Figure 1 shows the evolution of two systems that have been simulated deterministically and stochastically. There images show that when there are only a few molecules that take part in a reaction, stochastic effects become prominent and are manifested by occurrence of fluctuations in the time course of the reactants. When large numbers of molecules are present reactions usually proceed in a predictable manner because the fluctuations are averaged out. **Deterministic behaviour can be seen as a limit of the stochastic behaviour when the number of molecules is high** (figure 2).

FIGURE 1. Deterministic vs stochastic evolution: A, B - Evolution to a steady state. C, D - Self-sustained oscillations.



2.4. **Stochastic methods.**

FIGURE 2. Deterministic behaviour can be seen as a limit of the stochastic behaviour when the number of molecules is high.



In probability theory and statistics, a **Markov process** is a stochastic/random process for which, loosely speaking, one can make predictions for the future of the process based solely on its present state; its future and past are independent. Biochemical reactions, genetic and molecular networks and other biosystems behave in this way.

2.4.1. *The Chemical Master Equation.*

Math The **chemical master equation (CME)** governs the stochastic dynamics of Markov processes. In other words, this equation describes how the probability for any state of a Markov system (where each state is defined by the number of molecules, of each chemical species, present) changes with time. The CME consists of the deterministic, differential equation approximations that are obeyed by the mean of each species, as the numbers of molecules of all species increase.

Math The CME itself is analytically solvable, but only for 'simple' systems[1]. Nevertheless, several approximations exist, all of which exploit the tendency of fluctuations to decrease as the numbers of molecules increase.

2.4.2. *The Gillespie Algorithm.*

Comp As mentioned above, the CME becomes intractable once the number of chemical species in the system reaches relatively large numbers and/or have complex (non-linear) relations. However, one can readily generate realisations of the stochastic process described by the chemical master equation using the **Gillespie algorithm**.

This algorithm simulates **one sample time course** from the master equation. By doing many simulations and averaging, the mean and variance for each chemical species can be calculated as a

[1]Since the order of the CME is equal the number of possible molecular combinations, the dimension rapidly explodes with increasing number of molecules and reactions. e.g. For 200 molecules, there are one million different molecular combinations.

function of time (figure 2). The equivalence between the discrete description and the continuous master equation has been demonstrated by Gillespie [5].

The Gillespie algorithm is based on the assumptions that the system is homogeneous and well mixed. It simulates the time evolution of the system as follows: At each time step, the chemical system is exactly in one state (as a specific type and number of molecules). The algorithm determines the nature of the next reaction, as well as the time interval Δt till this reaction takes place, given that the system is in a given state at time t. Asptn

In practice, after setting the initial species populations S_i, reaction constants k_j and the final time t_{max} the algorithm runs in loop the following steps (until the final time is reached):

Step 1. Given the current state of the system, calculate the probability of each reaction occurring next.

Step 2. Pick two random numbers to specify the next reaction that will occur and when it will occur. The choice of these numbers is weighted, based on the probabilities of each reaction happening, calculated in Step 1.

Step 3. Adjust all the concentrations of the individual components to account for whatever reaction took place.

Step 4. Record the new state of the system either run the reaction again or terminate the simulation.

In this course you will not have to program this algorithm. We have provided you with one that you can use, written by Francois Nedelec. Let's look at an example of how to use this algorithm:

Example 1: Based on experimental observations, Michaelis and Menten (1913) proposed the following mechanism for enzyme-catalysed biochemical reactions:

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \xrightarrow{k_2} E + P$$

The deterministic evolution equations for each of the species, following the **mass action law** (see [3]) are:

$$\frac{dS}{dt} = -k_1 ES + k_1 C$$
$$\frac{dE}{dt} = -k_1 ES + k_1 C + k2_C$$
$$\frac{dC}{dt} = k_1 ES - k_1 C - k_2 C$$
$$\frac{dP}{dt} = k_2 C$$

In contrast, the stochastic version, we have to consider each reaction step and to associate to each of them a certain **probability** (reaction rate), which is denoted by $P(...)$. The equation looks like:

$$\begin{aligned}\frac{\partial P(S,C,E;t)}{\partial t} &= -(k_1 SE + (k_{-1} + k_2)C)P(S,C;t)\\ &+k_1(S+1)(E+1)P(S+1,C-1;t)\\ &+k_{-1}(C+1)P(S-1,C+1;t)\\ &+k_2(C+1)P(S,C+1;t)\end{aligned}$$

The following MATLAB code shows how to model the system, both stochastically (A) and deterministically (B) (see [3] for an introduction), and how to get a numerical solution to both models; the stochastic model is solved using Gillespie and the deterministic one, using MATLAB's *ode*45 function. The results can be observed in figure 3.

A. Stochastic simulation - recall that instead of trying to solve the CME, we try to approximate it using the Gillespie method.

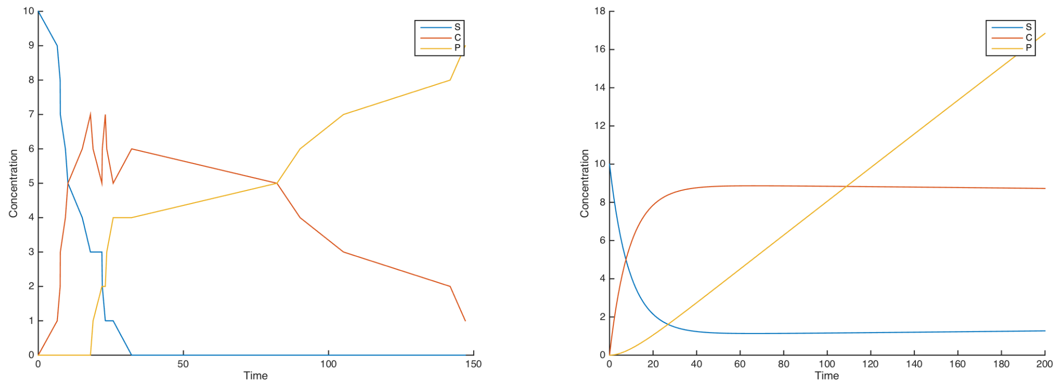
```
*In MATLAB - in a function file named stochmm.m*

% initial abundance [S E C P]
y = [10 100 0 0];

% rate constatns
rates = [0.001; 0.001; 0.01];

% stochiometry matrix: cols = reactions, rows = species
stoch = [-1, -1, 1, 0;1, 1, -1, 0; 0, 1, -1, 1];

% simulation stop time
tmax = 200;

% call function that simulates system with a stochastic solver
% this one was written by Francois Nedelec
% (use the documentation of this function to learn how to use it)
[Tstoch,Ystoch] = gillespie(stoch, rates, [], y, tmax);

%plot final concentrations as a function of time
figure
hold on
plot(Tstoch,Ystoch(:,1));
plot(Tstoch,Ystoch(:,3:4));
legend('S','C','P')
xlabel('Time');
ylabel('Concentration');
```

B. Deterministic simulation

```
*In MATLAB - in a function file named enzyme_reaction.m*

function dydt = enzyme_reaction(t,y)

% rates
k = [0.001; 0.001; 0.01];

% ODEs
dydt = zeros(4,1);

dydt(1) = -k(1)*y(1)*y(2) + k(2)*y(3);
dydt(2) = -k(1)*y(1)*y(2) + k(2)*y(3) + k(3)*y(3);
dydt(3) = k(1)*y(1)*y(2) - k(2)*y(3) - k(3)*y(3);
dydt(4) = k(3)*y(3);
```

```
*In a new script file*

% Initial conditions
y0 = [10,100,0,0];

% Integrate the system of differential equations
[t,y] = ode45(@enzyme_reaction,[0:200],y0);

% plot
figure
%plot(t,y);
hold on
plot(t,y(:,1));
plot(t,y(:,3:4));
hold off
legend('S','E*S','P')
xlabel('Time');
ylabel('Concentration');
```

FIGURE 3. Michaelis-Menten kinetics: stochastic vs deterministic, respectively.



Now it is your turn; try exercise 1.

2.4.3. *The Langevin equation.*

An alternative way to simulate stochastic systems is to introduce a stochastic term $\xi(t)$ in the deterministic evolution equation. This method is called the **Langevin approach**. In this method, the description is not in terms of molecules, as in the Gillespie algorithm, but in terms of the concentration, as in the deterministic case. The generic equation is:

$$\frac{dX}{dt} = f(X) + \xi(t)$$

This is called a **stochastic differential equation (SDE)**.

In practise, the easiest way to proceed in order to obtain a numerical solution of an SDE, is via **Euler's method**, which is one of the simplest possible numerical method for approximating solutions of equations. For an introduction, see [4].

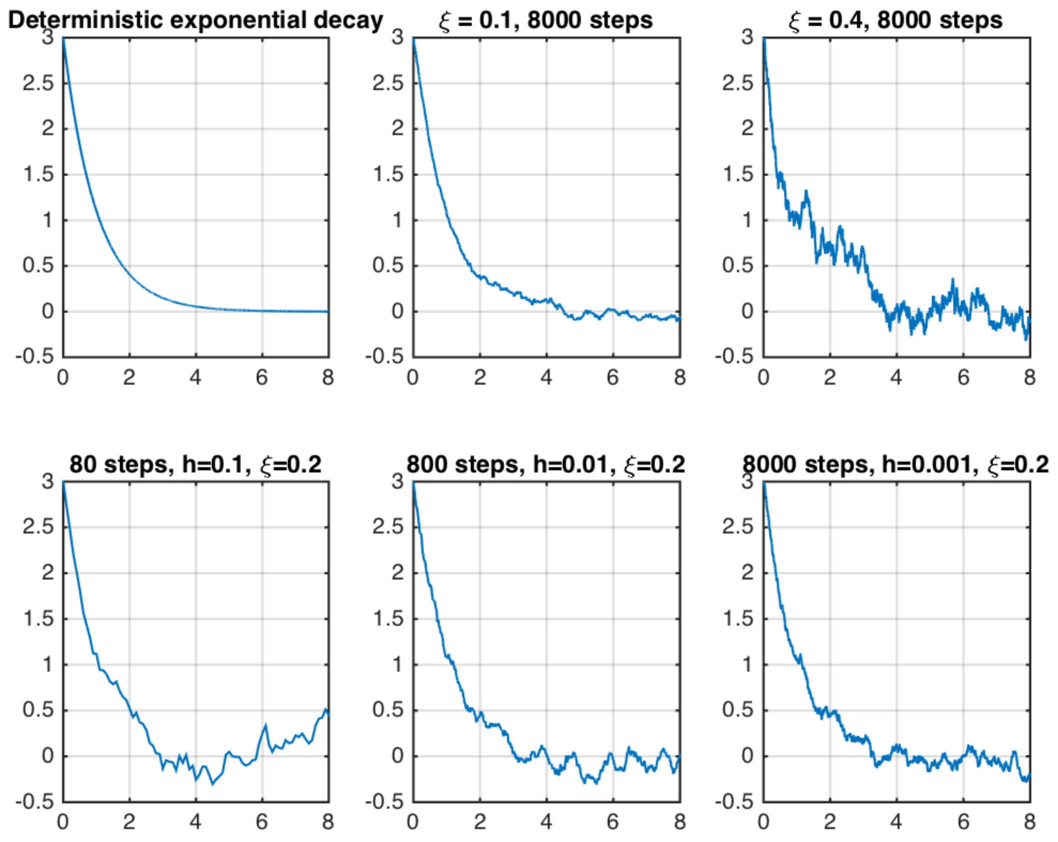
The following code illustrates how to use Euler's method to approximate the Langevin equation for decay with noise. Figure 4 shows the evolution of the system with different magnitudes of noise and step size and compares these with a deterministic model:

```
*In a new MATLAB script*

% Outcomes of exponential decay with noise (Langevin equation).
% This script investigaes the differences that arise with various values
% of h and xi.

figure

N = 8000;                    % number of steps to take
tmax = 8;                    % maximum time
h = tmax/N;                  % time step
t = (0:h:tmax);              % t is the vector [0 1h 2h 3h ... Nh]
y = zeros(size(t));          % prepare place to store locations

%deterministic
subplot(2,3,1);
y(1) = 3;                    % initial height
for i = 1:N                  % start taking steps
y(i+1) = y(i) - y(i)*h;
end;
plot(t,y), hold on         % plot more permanently
axis([0 tmax -2 3]);       % set axis limits
grid on;
title('Deterministic exponential decay');


% Langevin, noise = 0.1, step = 0.001
subplot(2,3,2);
xi = 0.1;                     % xi (noise)

y(1) = 3;                    % initial height
for i = 1:N                  % start taking steps
y(i+1) = y(i) - y(i)*h + xi*sqrt(h)*randn;
end;
plot(t,y), hold on         % plot more permanently
axis([0 tmax -1 3]);       % set axis limits
grid on;
title('\xi = 0.1, 8000 steps');


% Langevin, noise = 0.4, step = 0.001
```

```
subplot(2,3,3);
xi = 0.4;                     % xi

y(1) = 3;                    % initial height
for i = 1:N                  % start taking steps
y(i+1) = y(i) - y(i)*h + xi*sqrt(h)*randn;
end;
plot(t,y), hold on         % plot more permanently
axis([0 tmax -1 3]);       % set axis limits
grid on;
title('\xi = 0.4, 8000 steps');


% Langevin, noise = 0.2, step = 0.1
subplot(2,3,4);
N = 80;                      % number of steps to take
tmax = 8;                    % maximum time
h = tmax/N;                  % time step
t = (0:h:tmax);              % t is the vector [0 1h 2h 3h ... Nh]
y = zeros(size(t));          % prepare place to store locations
xi = 0.2;                     % xi

y(1) = 3;                    % initial height
for i = 1:N                  % start taking steps
y(i+1) = y(i) - y(i)*h + xi*sqrt(h)*randn;
end;
plot(t,y), hold on         % plot more permanently
axis([0 tmax -1 3]);       % set axis limits
grid on;
title('80 steps, h=0.1, \xi=0.2');


% Langevin, noise = 0.2, step = 0.01
subplot(2,3,5);
N = 800;                     % number of steps to take
tmax = 8;                    % maximum time
h = tmax/N;                  % time step
t = (0:h:tmax);              % t is the vector [0 1h 2h 3h ... Nh]
y = zeros(size(t));          % prepare place to store locations

y(1) = 3;                    % initial height
for i = 1:N                  % start taking steps
y(i+1) = y(i) - y(i)*h + xi*sqrt(h)*randn;
end;
plot(t,y), hold on         % plot more permanently
axis([0 tmax -1 3]);       % set axis limits
grid on;
title('800 steps, h=0.01, \xi=0.2');
```

```
 % Langevin, noise = 0.2, step = 0.001
subplot(2,3,6);
N = 8000;                    % number of steps to take
tmax = 8;                    % maximum time
h = tmax/N;                  % time step
t = (0:h:tmax);              % t is the vector [0 1h 2h 3h ... Nh]
y = zeros(size(t));          % prepare place to store locations

y(1) = 3;                    % initial height
for i = 1:N                  % start taking steps
y(i+1) = y(i) - y(i)*h + xi*sqrt(h)*randn;
end;
plot(t,y), hold on         % plot more permanently
axis([0 tmax -1 3]);    % set axis limits
grid on;
title('8000 steps, h=0.001, \xi=0.2');
```

FIGURE 4. Numerical approximation to the Langevin equation of decay with noise, using Euler's method.

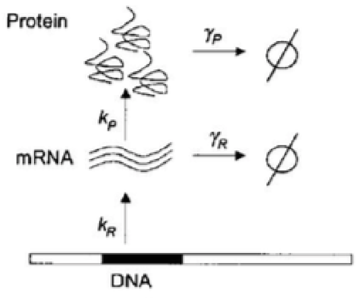


Now is your turn. Try exercise 2.

3. Exercises

Exercise 1. Use MATLAB to simulate the model in figure 5 of gene expression, using both a deterministic (MATLAB's *ode*45 function) and a stochastic (Gillespie algorithm) approach. Explore appropriate values for the rates of the reactions. Note that there is a compound that is not consumed in the reactions. Compare the two models and make observations. Make use of histograms of the number of molecules to help you. If you are interested, expand this system to include feedback loops and model.

Figure 5. Gene expression model, by Thattai and van Oudenaarden.



Answer 1. See figure 6.
ODE model

```
*In MATLAB, in a function file called genetic_expression_simple*

function dydt = genetic_expression_simple(t,y)

% rates
k = [0.01;log(2)/120;log(2)/60;log(2)/3600];
% 120 and 3600 are the half-life of mRNA and P, respectively

% ODEs
dydt = zeros(3,1);

dydt(1) = 0;
dydt(2) = k(1)*y(1) - k(2)*y(2) - k(3)*y(2);
dydt(3) = k(3)*y(2) - k(4)*y(3);
```

```
*In MATLAB, in a script file*

% Initial conditions
y0 = [1,0,0];

% Integrate the system of differential equations
[t,y] = ode45(@genetic_expression_simple,[0:200],y0);
```

```
% plot
figure
plot(t,y);
%hold on
%plot(t,y(:,1));
%plot(t,y(:,3:4));
%hold off
legend('DNA', 'mRNA','P')
xlabel('Time');
ylabel('Concentration');
```

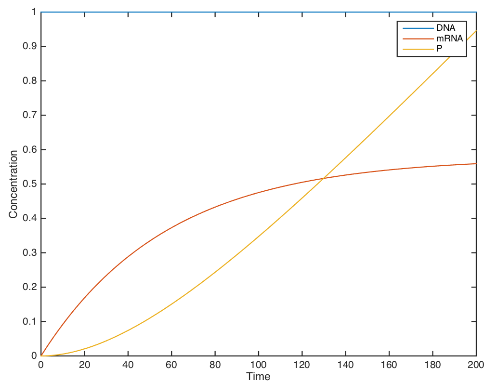
Stochastic model

```
% initial abundance [S E C P]
y = [1 0 1];

% rate constatns
k = [0.01;log(2)/120;log(2)/60;log(2)/3600];
% 120 and 3600 are the half-life of mRNA and P, respectively

% stochiometry matrix: cols = reactions, rows = species
stoch = [0,1,0;0,-1,0;0,-1,1;0,0,-1];

% simulation stop time
tmax = 2000;

% call function that simulates system with a stochastic solver
% this one was written by Francois Nedelec
[Tstoch,Ystoch] = gillespie(stoch, rates, [], y, tmax);

%plot final concentrations as a function of time
figure
%hold on
plot(Tstoch,Ystoch(:,3));
%plot(Tstoch,Ystoch(:,3:4));
legend('P')
xlabel('Time');
ylabel('Concentration');


P = Ystoch(:,3);
mRNA = Ystoch(:,2);
histogram(P)
histogram(mRNA)
```

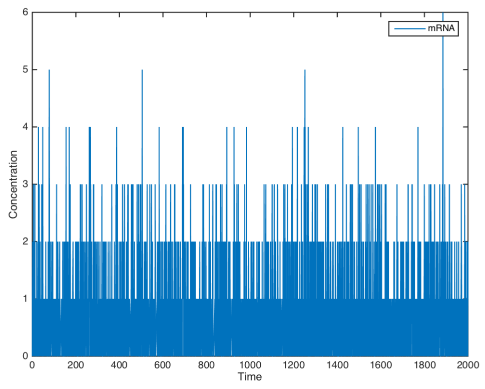
Observations:

In the deterministic case, mRNA reaches a steady state equal to $RSS = k1/k2$. At the steady state, the protein level depends on mRNA and rapidly reaches steady state after mRNA does.

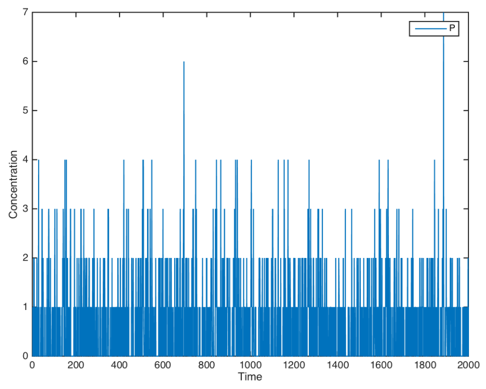
Figure 6. Gene expression: deterministic (A) and stochastic simulation of mRNA (B) and protein (C), with respective histograms (D, E).



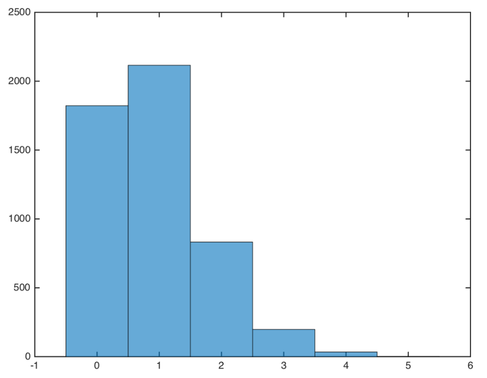
(A)



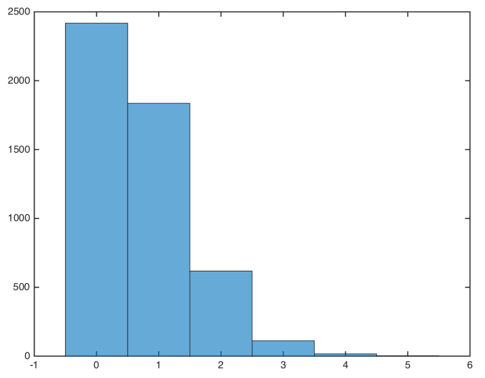
(B)



(C)



(D)



(E)

The stochastic simulation of this model reveals fluctuation at the level of both mRNA and protein, but with different characteristics.

Exercise 2. Regulation of gene expression by signals from both outside and inside the cell play an important role in many biological processes and it has become clear that it is a complex process, involving nonlinear interactions, positive and negative feedback within signalling pathways, time delays, crosstalk, fluctuations, etc.

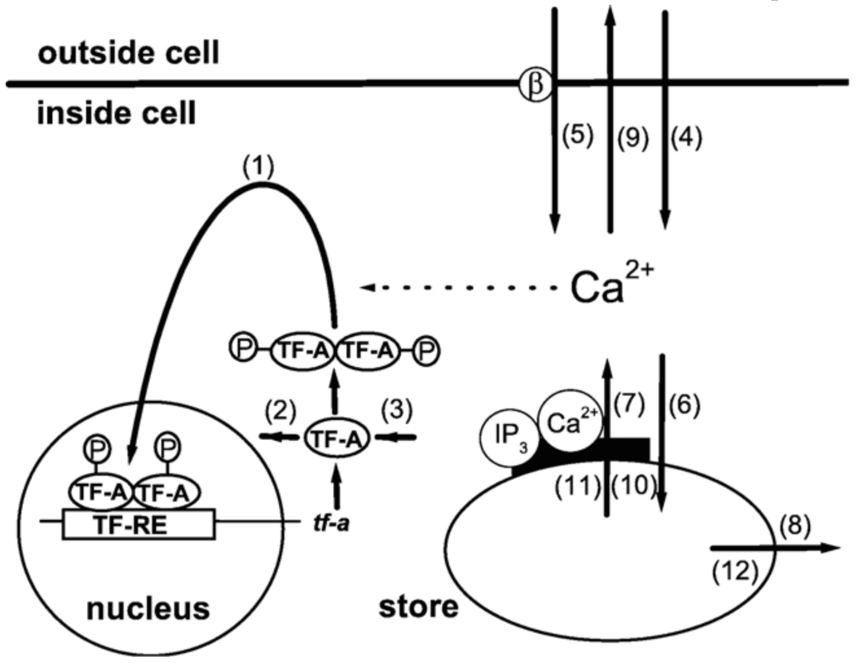
The oscillation of intracellular calcium (Ca2+) plays an important role in the control of many cellular processes. In order to study its involvement in the activation of a transcription factor, we will use the model in figure 7. The relevant equations are: equations (2) and (3) in [2] describe the 'minimal model of cytosolic Ca^{2+} oscillations'. Equation (1) in the paper models the activation of a transcription factor. Table 1 has the parameter values and table 2 the rates of the reactions.

(A). The models used in [2] can be downloaded from the Biomodels Database Go to the website and inspect how the information is provided.
(B). You do not need to download the model, it is below, in blue. It currently contains enough information encoding a deterministic model of the system. You need to edit it in MATLAB and run what you get (using ode45), in order to get a numerical solution.
(C). Now, make a Langevin model of the same system and use the Euler method for numerical approximation to a solution; use figure 4 to help you choose appropriate step size and noise (h and ξ, respectively)).

Answer the following questions:

(i). Compare the results of B and C.
(ii). Change the values of the step and the noise in C and make observations.
(iii) Can you reproduce any of the conclusions of this paper?

FIGURE 7. Model of intracellular Ca^{2+} and activation of a transcription factor.



```
%ODEs:
d(TF_A)/dt = (RF1x1 - RF1x2 + RF1x3); %X
```

```
d(Cc)/dt = (RFlx4 + RFlx5 - RFlx6 + RFlx7 + RFlx8 - RFlx9); %Z
d(Cs)/dt = (RFlx6 - RFlx7 - RFlx8); %Y

%Fluxes:
RFlx1 = kf*(TF_A)^2/((TF_A)^2+Kd); % r1
RFlx2 = kd*TF_A; % r2
RFlx3 = Rbas; % r3
RFlx4 = v0; % r4
RFlx5 = v1*beta; % r5
RFlx6 = Vm2*(Cc)^n/((K2)^n+(Cc)^n); % r6
RFlx7 = (Vm3*(Cs)^m/((Kr)^m+(Cs)^m))*((Cc)^p/((K_A)^p+(Cc)^p)); % r6, r10
RFlx8 = k1*Cs; % r7, r11
RFlx9 = k*Cc; % r8, r12

%Repeated Assignments:
Kd = Kd0/(1+(Cc)^4/(Kb)^4);
kf = kf0*(1+gamma*(Cc)^4/((Ka)^4+(Cc)^4));

%Parameter Values:
Ka = 0.5;
gamma = 9;
kd = 1;
k1 = 0.7;
v0 = 1;
Vm2 = 30;
Vm3 = 325;
K_A = 0.46;
m = 2;
Kb = 0.5;
Kd0 = 10;
Rbas = 0.1;
k = 10;
v1 = 5.7;
K2 = 0.5;
Kr = 1.7;
n = 2;
p = 4;

beta = 0.3; % external stimulation
kf = 9.1765;
Kd = 9.4118;
kf0 = 6;

%Initial Conditions:
TF_A = 15;
Cc = 0.25;
Cs = 0;
```

Answer 2.(B). See figure 8.

```
*In a MALAB function file named calcium_oscillations_model.m*

function dydt = calcium_oscillations_model(t,y)

TF_A = y(1);
Cc = y(2);
Cs = y(3);

%Parameter Values:
Ka = 0.5;
gamma = 9;
kd = 1;
k1 = 0.7;
v0 = 1;
Vm2 = 30;
Vm3 = 325;
K_A = 0.46;
m = 2;
Kb = 0.5;
Kd0 = 10;
Rbas = 0.1;
k = 10;
v1 = 5.7;
K2 = 0.5;
Kr = 1.7;
n = 2;
p = 4;

beta = 0.15; % external stimulation
%cytoplasm = 1;
%store = 1;
%kf = 9.1765;
%Kd = 9.4118;
kf0 = 6;


%Repeated Assignments:
Kd = Kd0/(1+((Cc^4)/(Kb^4))); %Kd0/(1+(((Cc)^4)/((Kb)^4));
kf = kf0*(1+((gamma*(Cc^4))/(Ka^4+Cc^4))); %kf0*((1+gamma*(Cc)^4)/((Ka)^4+(Cc)^4));

%Fluxes:
RFlx1 = kf*(TF_A)^2/((TF_A)^2+Kd); % r1
RFlx2 = kd*TF_A; % r2
RFlx3 = Rbas; % r3
RFlx4 = v0; % r4
RFlx5 = v1*beta; % r5
RFlx6 = Vm2*(Cc)^n/((K2)^n+(Cc)^n); % r6
RFlx7 = (Vm3*(Cs)^m/((Kr)^m+(Cs)^m))*((Cc)^p/((K_A)^p+(Cc)^p)); % r6, r10
RFlx8 = k1*Cs; % r7, r11
```

```
RFlx9 = k*Cc; % r8, r12


%ODEs:
%d(TF_A)/dt = 1/cytoplasm*(RFlx1 - RFlx2 + RFlx3); %X
%d(Cc)/dt = 1/cytoplasm*(RFlx4 + RFlx5 - RFlx6 + RFlx7 + RFlx8 - RFlx9); %Z
%d(Cs)/dt = 1/store*(RFlx6 - RFlx7 - RFlx8); %Y

dydt = zeros(3,1);

dTF_Adt = (RFlx1 - RFlx2 + RFlx3); %X
dCcdt = (RFlx4 + RFlx5 - RFlx6 + RFlx7 + RFlx8 - RFlx9); %Z
dCsdt = (RFlx6 - RFlx7 - RFlx8); %Y

dydt(1) = dTF_Adt;
dydt(2) = dCcdt;
dydt(3) = dCsdt;
```

```
*In a new MATLAB script*

%Initial Conditions:
TF_A = 15; %X
Cc = 0.25; %Z
Cs = 0; %Y

y = [TF_A, Cc, Cs];
t = [0 80];

% Integrate the system of differential equations
[t,y] = ode45(@calcium_oscillations_model,t,y);

figure
plot(t,y)
%hold on
%plot(t,y(:,1))
%plot(t,y(:,3))
%hold off
xlabel('Time, s', 'FontSize',12)
ylabel('Concentration, mM', 'FontSize',12)
legend('TF_A', 'Cc', 'Cs')
```

Answer 2.(C). See figure 9.

```
*In MATLAB in a new script*


%% Preliminaries

clc
```

```
clear
%close all

N=80000;                 % number of steps to take
T=80;                    % maximum time
h=T/N;                   % time step
t=(0:h:T);               % t is the vector [0 1h 2h 3h ... Nh]

TF_A=zeros(size(t));     % prepare place to store locations
Cc=zeros(size(t));       % prepare place to store locations
Cs=zeros(size(t));       % prepare place to store locations

% Init. conds.
TF_A(1) = 15;            %X
Cc(1) = 0.25;            %Z
Cs(1) = 0;               %Y

%Parameter Values:
Ka = 0.5;
gamma = 9;
kd = 1;
k1 = 0.7;
v0 = 1;
Vm2 = 30;
Vm3 = 325;
K_A = 0.46;
m = 2;
Kb = 0.5;
Kd0 = 10;
Rbas = 0.1;
k = 10;
v1 = 5.7;
K2 = 0.5;
Kr = 1.7;
n = 2;
p = 4;

beta = 0.15;             % external stimulation
%cytoplasm = 1;
%store = 1;
%kf = 9.1765;
%Kd = 9.4118;
kf0 = 6;


%% Simulation (Euler)

xi=0.2;                  % noise
```

```
for i=1:N                % start taking steps
    %Repeated Assignments:
    Kd = Kd0/(1+((Cc(i)^4)/(Kb^4)));
    kf = kf0*(1+((gamma*(Cc(i)^4))/(Ka^4+Cc(i)^4)));

    %Fluxes:
    RFlx1 = kf*(TF_A(i))^2/((TF_A(i))^2+Kd);        % r1
    RFlx2 = kd*TF_A(i);                             % r2
    RFlx3 = Rbas;                                   % r3
    RFlx4 = v0;                                     % r4
    RFlx5 = v1*beta;                                % r5
    RFlx6 = Vm2*(Cc(i))^n/((K2)^n+(Cc(i))^n);       % r6
    RFlx7 = (Vm3*(Cs(i))^m/((Kr)^m+(Cs(i))^m))*((Cc(i))^p/((K_A)^p+(Cc(i))^p)); % r6, r10
    RFlx8 = k1*Cs(i);                               % r7, r11
    RFlx9 = k*Cc(i);                                % r8, r12

    % ODEs
    TF_A(i+1)=TF_A(i)+ h*(RFlx1 - RFlx2 + RFlx3) + xi*randn*sqrt(h); %TF_A
    Cc(i+1)=Cc(i)+ h*(RFlx4 + RFlx5 - RFlx6 + RFlx7 + RFlx8 - RFlx9) +
    xi*randn*sqrt(h); %Cc
    Cs(i+1)=Cs(i)+ h*(RFlx6 - RFlx7 - RFlx8) + xi*randn*sqrt(h);     %Cs
end;


figure
hold on
plot(t,TF_A)
plot(t,Cc)
plot(t,Cs)
hold off
grid on;
title('80000 steps, h=0.001, \xi=0.2');
```

FIGURE 8. Results of deterministic model of the activation of transcription factor modulated by intracellular Ca^{2+} oscillations. Numerical integration using MATLAB's ode45.

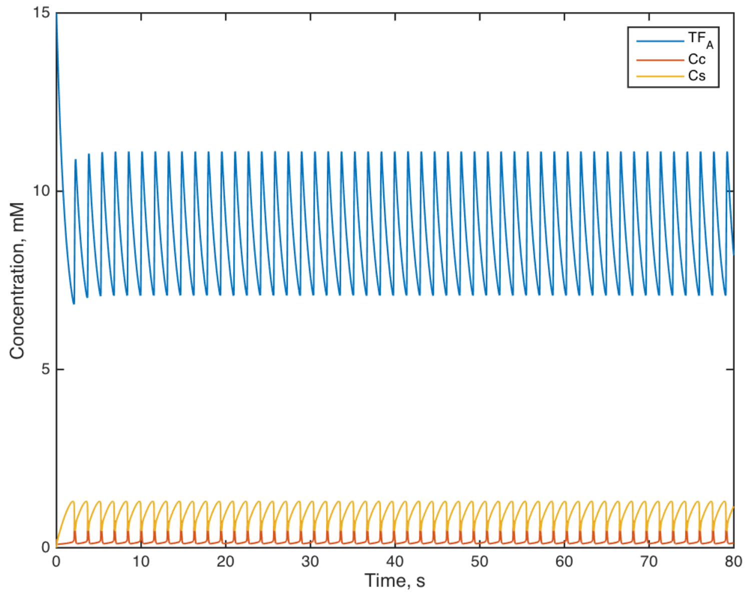
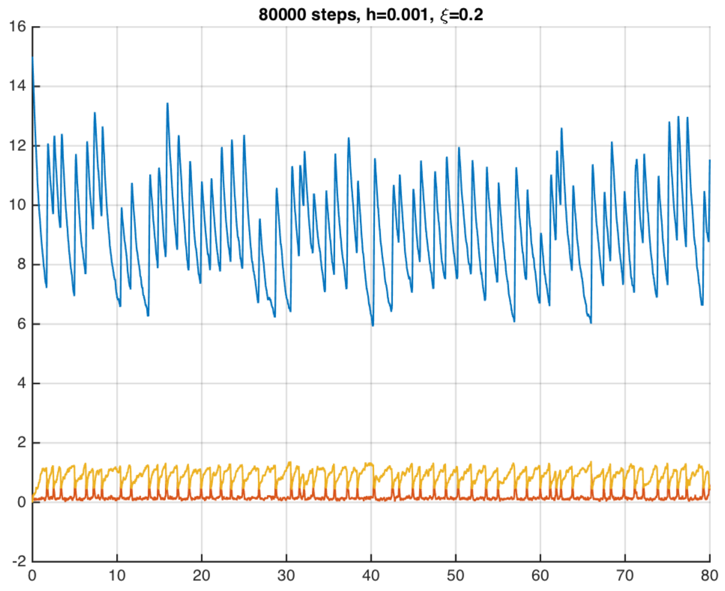


FIGURE 9. Results of stochastic (Langevin) model of the activation of transcription factor modulated by intracellular Ca^{2+} oscillations. Numerical integration using the Euler method.



4. References

[1] Stochastic simulations - Application to molecular networks, by Dieder Gonze, Les Houches Spring School, 2007.

[2] Chun-lian Zhu, Yuan Zheng, Ya Jia, A theoretical study on activation of transcription factor modulated by intracellular Ca^{2+} oscillations, *Biophysical Chemistry 129, 49-55*, 2007.

[3] Karin Sasaki, EMBL-CBM, A primer on deterministic models of biochemical reactions

[4] Karin Sasaki, EMBL-CBM, A primer on numerical methods - Euler's method

[5] Daniel T. Gillespie, Exact Stochastic Simulation of Coupled Chemical Reactions, *The Journal of Physical Chemistry*, 81 (25): 2340-2361.