Mathematical Model of Infection Mechanism of Bordetella
Fall 2016 ChBE 6500

Course:	**ChBE 6500**
Name:	**Nikhil Raj**
	Yamin Zhang
	Mengqi Fan
Professor:	**Dr. Martha Grover**
Date:	**12/01/2016**

Abstract

In this work, the infection strategy of Bordetella has been modelled. Four classes of genes, called virulence factors, need to perform specific dynamic response in Bordetella to infect its host. These genes are the products of a two-component signal transduction system known as BvgAS. The proteins that are modelled in this papers are namely, BvgA, BvgS, class 1, class 2, class 3 and class 4. BvgA can exist in three possible states which depends on the phosphorylation of specific domain of protein, these states are namely BvgA-P1, BvgA-P2, BvgA-P3. Also BvgA protein can also state exist in two active and inactive states. Active BvgA is responsible for auto-transcription and transcription of class1 to class 4 genes. We used stochastic simulation to solve the 20 model equations as given in the paper. Gillespie algorithm will be used to solve the system of equations. In Gillespie algorithm we define the probability of each reaction happening and then this probability will decide which reaction will happen more frequently. We assume that at any instance only one reaction will be occurring (proportional to its probability). The number of molecules of certain species will then be changed according to these defined probabilities. We reproduced all the different results given in the paper by reproducing the dynamic plots of class 1-class 4 gene expressions.

Introduction

Central Dogma process explains how genetic information is transferred from DNA to protein through several steps such as DNA replication, transcription, translation, and protein production. This production of protein can be triggered by some environmental changes outside the cell. When the environmental signals outside fluctuates, the protein which are embeded on the cell membrane senses these signals and transfer this information further by activating another protein inside the cell called transcription factor. The transcription protein after getting activated by membrane protein then interacts with particular gene and trigger the protein that gene's production. This protein then responses to the changes in the environment.

In this work we mathematically modelled the signal transduction and central dogma process of a bacteria called bordetella. Bordetella is genus of small (0.2-0.7 μm), Gram-negative coccobacilli of the phylum Proteobacteria. There are three closely related subspecies: B. bronchiseptica, B. parapertussis, B. pertussis and they are pathogenic in many hosts.[1, 2] Among these, B. pertussis is a strict human pathogen that is the causative agent of pertussis (whooping cough). Its natural habitat is in the human respiratory mucosa. Whooping Cough, or pertussis, is a respiratory infection in which a "whooping" sound is produced when the sufferer breathes.[3]

Virulence factors are the proteins which are responsible for the infection in the host. In Bordetella there are four virulence proteins which are regulated by a protein system inside a cell called BvgAS. In its life time bordetella faces various types of environment and in response to that it changes the dynamics of virulence factors production so that it can effectively infect its host. Three types of environmental changes were considered in our model namely Bvg^-, Bvg^i and Bvg^+,[4, 5] changes in these environemnt were modelled by changing the signal strength (S). Bvg^-

represents absence of signal, Bvg^+ represents the signal strength is very high and Bvg^i represents that the concentration of signal is intermediate.

In the presence of signal, the trans-membrane protein (BvgS) activates itself by attaching a phosphate group (PO_4^-). This phosphate group is then travels through three domains of BvgS to ultimately activate another protein called BvgA (Figure 1).[6] Once BvgA is activated to BvgA-P, it then acts as a transcription factor which regulates the protein production of all the virulence genes (Class 1 to Class 4) and also responsible for production of BvgAS system proteins (Positive feedback).

Each biochemical and genetic interaction can be considered to be a chemical reaction whose rate laws are written as described in method section in this report. In this project, we used stochastic simulation to solve these 20 "chemical reactions" equations as given in the paper. Gillespie algorithm was used to solve the system of equations which is decribed later in the methid section. We reproduced different results given in the paper by reproducing the dynamic plots of class 1-class 4 gene expressions.

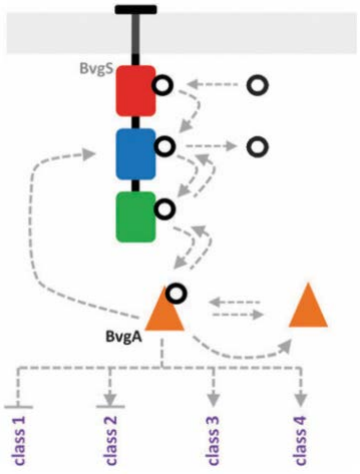

Fig.1 Cartoon representation of the BvgAS regulatory network. In presence of signal, BvgS autophosphorylates itself, and passes the phosphate group (black circles) to BvgA (triangle) in three steps. Phosphorylated BvgA then controls expression of class 1, 2, 3, and 4 genes. In addition, BvgA-P also controls expression of BvgA and BvgS.

.

Model construction

In the paper, there are 20 "chemical reaction" equations. The rate constants in above reactions are listed in Table 1. Since the model is stochastic in nature, Gillespie Algorithm were used to solve the system of equations.[7, 8].

Gillespie Algorithm

In probability theory, the Gillespie algorithm generates a statistically correct trajectory of a stochastic equation.

The physical basis of the algorithm is the collision of molecules within a reaction vessel. It is assumed that collisions are frequent, but collisions with the proper orientation and energy are infrequent. Therefore, all reactions within the Gillespie framework must involve at most two molecules. Reactions involving three molecules are assumed to be extremely rare and are modeled as a sequence of binary reactions. It is also assumed that the reaction environment is well mixed.

Below is a summary of the steps to run the algorithm (math omitted):

a) Initialization: Initialize the number of molecules in the system, reaction constants, and random number generators.
b) 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.
c) Update: Increase the time step by the randomly generated time in Step 2. Update the molecule count based on the reaction that occurred.
d) Iterate: Go back to Step 2 unless the number of reactants is zero or the simulation time has been exceeded.

Simple example: Reversible binding of A and B to form AB dimers

In the system A and B reversibly bind together to form AB dimers. So there are two reactions.

(1) A reacts with B to form AB. The rate constant is k_D.

$$\mathrm{A} + \mathrm{B} \rightarrow \mathrm{AB}$$

(2) AB dissociates into A and B. The rate constant is k_B.

$$\mathrm{AB} \rightarrow \mathrm{A} + \mathrm{B}$$

The rate expression

$$r_1 = k_D[A][B]$$
$$r_2 = k_B[AB]$$

The total reaction rate, R_{TOT}, at time *t* is then given by

$$R_{TOT} = k_D[A][B] + k_B[AB]$$

Reactions are assumed to be completely random. The probability that reaction is

$$P(\mathrm{A} + \mathrm{B} \rightarrow \mathrm{AB}) = {k_D[A][B]}\Big/{R_{TOT}}$$

$$P(\mathrm{AB} \rightarrow \mathrm{A} + \mathrm{B}) = 1 - P(\mathrm{A} + \mathrm{B} \rightarrow \mathrm{AB})$$

Let

$$\mathrm{r} = R_{TOT} \times random$$

If $\mathrm{r} \leq r_1$, assume that only the first reaction occurs. If $r_1 < \mathrm{r} \leq R_{TOT}$, assume that only the second reaction occurs.

Model description and assumptions

This model captures the biochemical and genetic interactions in the BvgAS regulatory network.

The following interactions were captured in the model:

a) Auto-phosphorylation of BvgS in response to environmental signals.
b) Transfer of phosphate group leading to phosphorylation of BvgA.
c) Reverse transfer of phosphate from BvgA to BvgS.
d) Binding of phosphorylated BvgA at virulence gene promoters to control the production of virulence proteins.
e) Degradation terms for all species of BvgS, BvgA, and the various classes of genes.
f) An environmental signal (S) was assumed which triggers BvgS autophosphorylation. The strength of the environmental signal is accounted for in the model by varying the numerical values assigned to the variable S. The S values represent the Bvg^- (at low value, S =0), Bvg^i (at intermediate value, S =1.5), and Bvg^+ (at high value, S = 9) phases.
g) For modeling mutants, the appropriate parameters values were changed to another appropriate value.

Each simulation was carried out for 50 cells, and then concentration of different protein of the whole system were calculated by taking the average concentration of all the 50 cells. The simulation was run for a particular signal value till the system reaches steady state, after the steady state has been reached, the signal strength value was changed, and he dynamics of the transitions were studied. Same procedure was followed to capture the dynamics of all 4 class of proteins in six different environmental transitions. These transitions are (a) Bvg^- to Bvg^i (b) Bvg^i to Bvg^+; (c) Bvg^- to Bvg^+;(d) Bvg^i to Bvg^-; (e) Bvg^+ to Bvg^i; and (f) Bvg^+ to Bvg^-.

Model equations

We used parameter values listed in the paper (Table 1).

Table 1 List of parameters used in the model

Parameter	Value
k_1	2
k_{1a}	10
k_{1b}	5
k_2	15
k_3	1.6
k_4	15
k_5	1.7
k_6	2
$k6_p$	0.5
k_{6q}	0.5
k_7	1
k_{7r}	0.5
k_{7s}	0.5
k_8	0.01
k_9	0.01
k_{10}	0.025
k_{11}	4
k_{11a}	1.5
k_{12}	1
k_{12r}	0.55
k_{13}	12
k_{13r}	4.5
k_{13a}	1.8
k_{14}	5
k_{14a}	0.1
k_{15}	6.5
k_{15a}	10
k_{16}	0.05
k_{17}	0.06
k_{18}	0.09
k_{19}	0.09
k_{20}	0.1

The rate laws for different interactions are listed below.

(1) Auto-phosphorylation of BvgS into BvgS-P1 (activation of BvgS)

$$r_1 = \frac{k_1 \times BvgS \times S/k_{1a}}{\left(1 + \frac{S}{k_{1a}}\right) \times (k_{1b} + BvgS)}$$

(2) Transfer from BvgS-P1 to BvgS-P2

$$r_2 = k_2 \times BvgS - P1$$

(3) Transfer from BvgS-P2 to null (Dephosphorylation)

$$r_3 = k_3 \times BvgS - P2$$

(4) Transfer from BvgS-P2 to BvgS-P3

$$r_4 = k_4 \times BvgS - P2$$

(5) Transfer from BvgS-P3 to BvgS-P2 (reverse transfer of phosphate group)

$$r_5 = k_5 \times BvgS - P3$$

(6) Transfer from BvgS-P3 to BvgA (activation of BvgA protein)

$$r_6 = \frac{k_6 \times BvgS - P3 \times BvgA}{(k_{6P} + BvgS - P3) \times (k_{6q} + BvgA)}$$

(7) Transfer from BvgA to BvgS-P3

$$r_6 = \frac{k_7 \times BvgA - P \times BvgS}{(k_{7r} + BvgS) \times (k_{7s} + BvgA - P)}$$

(8) Degradation of BvgS

$$r_8 = k_8 \times BvgS$$

(9) Degradation of BvgA

$$r_9 = k_9 \times BvgA$$

(10) Degradation of BvgA-P

$$r_{10} = k_{10} \times BvgA - P$$

(11) Production of BvgA and BvgS proteins

$$r_{11} = \frac{k_{11} \times (BvgA - P)^2}{(k_{11a})^2 + (BvgA - P)^2}$$

(12) Production of Class 1 proteins

$$r_{12} = \frac{k_{12}}{1 + (BvgA - P/k_{12r})^2}$$

(13) Production of Class 2 proteins

$$r_{13} = \frac{k_{13} \times (BvgA - P)^2}{[(k_{13a})^2 + (BvgA - P)^2] \times [1 + (BvgA - P/k_{13r})^2]}$$

(14) Production of Class 3 proteins

$$r_{14} = \frac{k_{14} \times (BvgA - P)^2}{(k_{14a})^2 + (BvgA - P)^2}$$

(15) Production of Class 4 proteins

$$r_{15} = \frac{k_{15} \times (BvgA - P)^4}{(k_{15a})^4 + (BvgA - P)^4}$$

(16) Degradation of class 1

$$r_{16} = k_{16} \times class\ 1$$

(17) Degradation of class 2

$$r_{17} = k_{17} \times class\ 2$$

(18) Degradation of class 3

$$r_{18} = k_{18} \times class\ 3$$

(19) Degradation of class 4

$$r_{19} = k_{19} \times class\ 4$$

(20) BvgA-independent (basal) production of BvgAS

$$r_{20} = k_{20}$$

The dynamics of virulence protein production in wild type Bordetella

As detailed in the methods section, the simulations were carried out using the Gillespie algorithm for stochastic reactions. We did six iterations. The values of S are listed below (Table 2).

Table 2 List of S in six iterations

Iteration	S	Phenotypic phase
1	0	Bvg^-
2	9	Bvg^+
3	0	Bvg^-
4	1.5	Bvg^i
5	9	Bvg^+
6	1.5	Bvg^i

The picture below is our simulation results for the virulence protein dynamics for wild type bordetella. (Fig. 2).

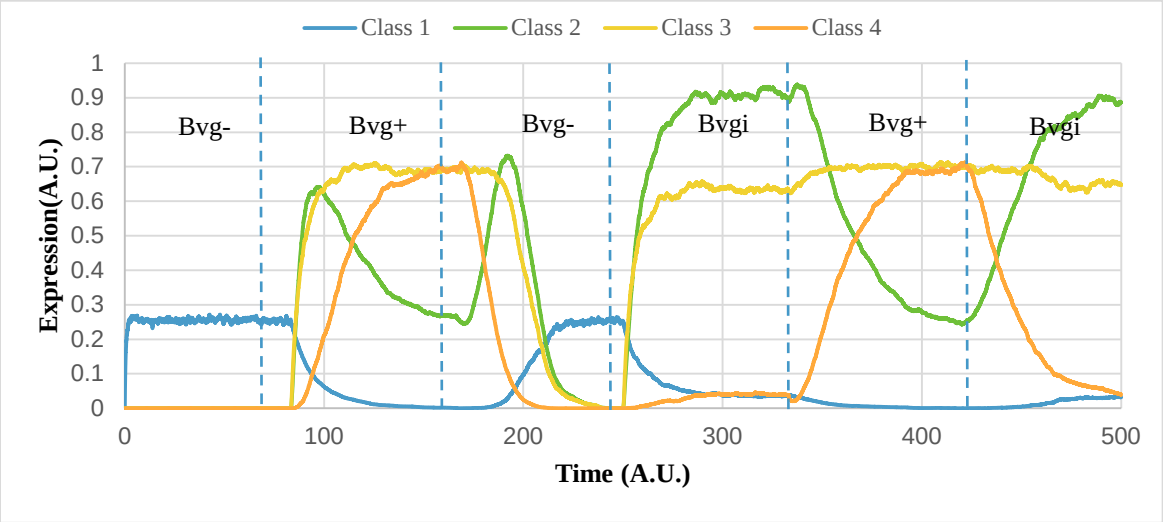

Fig. 2 Normalized gene expression dynamics of class 1, class 2, class 3, and class 4 under various environmental conditions. The graph captures dynamics in Bordetella in the six possible transitions between Bvg^-, Bvg^i, and Bvg^+ conditions. The transitions were studied by varying the environmental signal strength, S.

Results and discussion

Here we show the results how dynamics of virulence protein production changes in the mutant bordetella. We modelled the mutant bordetella in various ways by appropriately changing the rate constants of the rate aw. The dynamics of all 4 class of genes were then compared with the wild type bordetella.

Comparing the dynamics of mutant protein with wild type when only one domain of BvgS is present instead of three

To investigate the significance of the presence of three domains, we performed simulations for systems where BvgS possessed only one phosphotransfer domain, and compared against the wild-type BvgS. Simulations were performed with a different BvgS mutant. We changed k_2, k_3

and k_4 to 1500, 0 and 1500, respectively. All other biochemical parameters of the system were kept identical as the wild type. The simulation results are indicated in the below figure (Fig. 3).
As clearly seen in the figure 2, the mutant has significant impact in the dynamics of virulence proteins. The class 4 genes when transitioning from Bvg^- to Bvg^i condition were unable to repress its production as it is supposed to do according to wild type. Class 2 gene dynamics is also very different as compared to wild type. The dynamics indicates that the presence of three domains in in BvgS confers flexibility to the system in response to environmental signal. Results developed from our simulation matches with the one in paper.

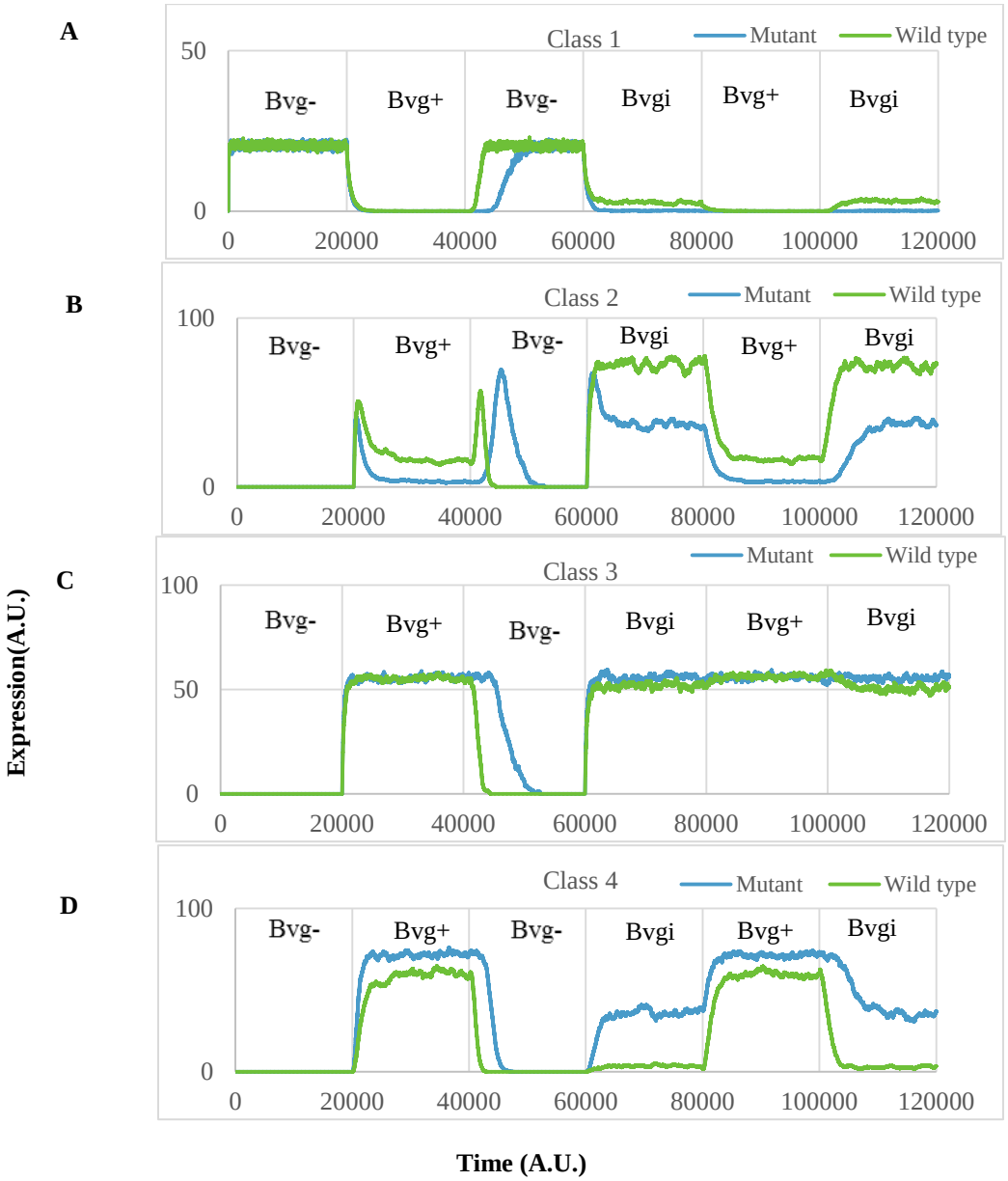

Fig. 3 Dynamics of expression of class 1 (A), class 2 (B), class 3 (C), and class 4 (D) genes in wild-type Bordetella (green line), BvgS with a single component (BvgS1C) (blue line) in the six transitions between Bvg^-, Bvg^i, and Bvg^+ environments.

Comparing the dynamics of mutant protein with wild type when Dephosphorylation is not allowed

We investigated the impact of the dephosphorylation step in the BvgS. We compared the dynamics of the four classes of genes in wild-type Bordetella and a BvgS-mutant (BvgS-DP), where the dephosphorylation step is not permitted. In the simulations, we set k_3=0, which means the rate constant for dephosphorylation equal to zero. Results are showed in Fig. 4, which is the same as the figure given in the paper. The results show that removal of the dephosphorylation step makes the system more sensitive to the input signal at low values of the input signal and it makes dynamics of class 1 genes taking place in the Bvg^i to Bvg^- phase change. When cells enter the the Bvg^i conditions, class 1 gene expression is completely off as the same behavior with class 2 genes. The removal of dephosphorylation was found to have a great effect on deactivation dynamics of class 3 genes. Because of the increased sensitivity to the input signal, the mutant showed a delayed inactivation as compared to the wild type.

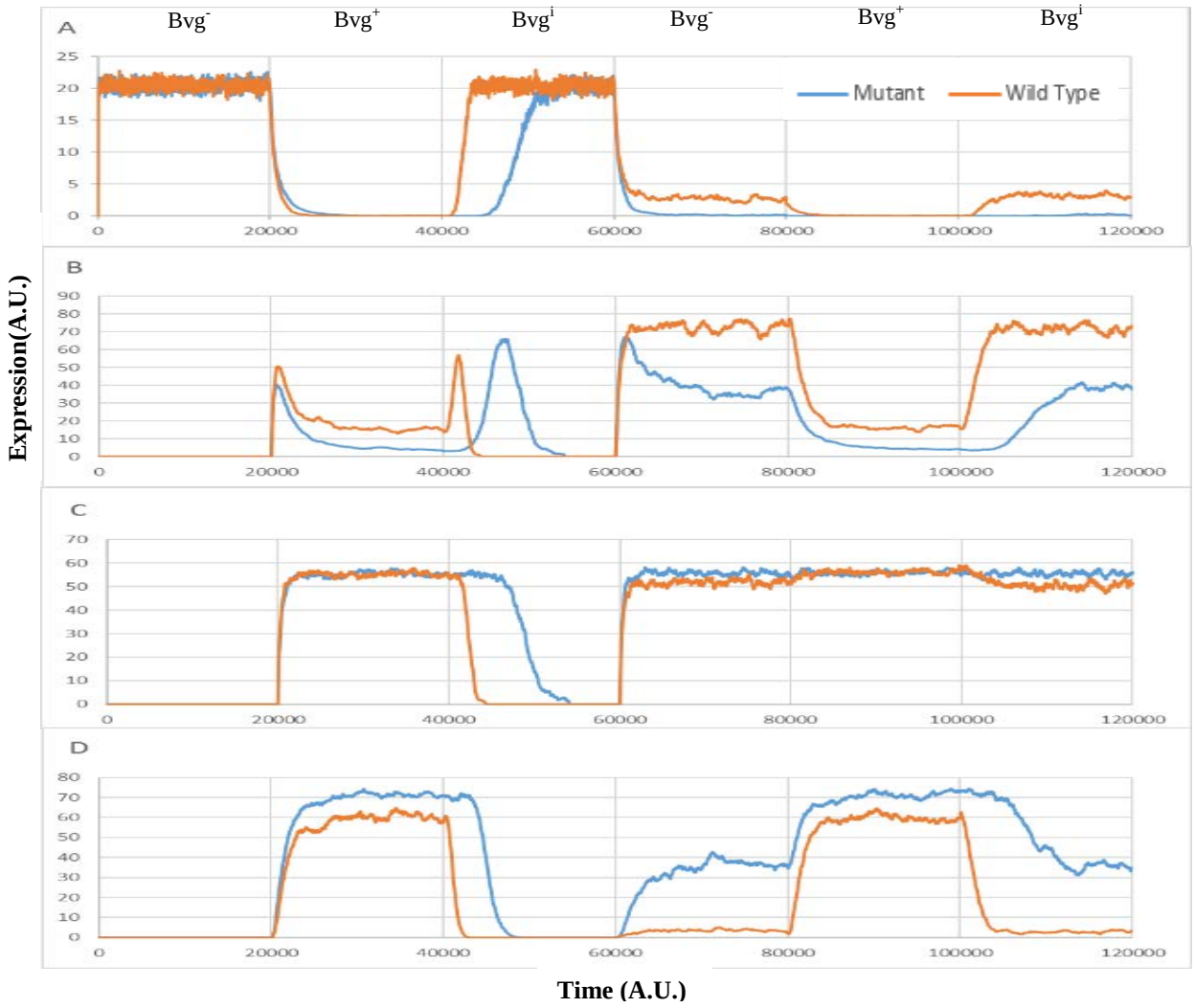

Fig.4 Comparison of dynamics of class 1 (A), class 2 (B), class 3 (C), and class 4 (D) genes in wild type (orange) and BvgS-DP (BvgS mutant where dephosphorylation is not allowed) (blue) in transitions between Bvg^-, Bvg^i, and Bvg^+ conditions.

Comparing the dynamics of mutant protein when BvgA is not allowed to produce itself

To test the importance of positive feedback in the system, we let $k_{11}=0$ and make other terms in the model were kept the same. The results areas shown in Fig.5 (same as the figure showed the paper). By comparing the dynamics of different classes of genes between wild type and the mutant without the feedback, we find that positive feedback is essential to control the long term expression profiles. In the figure, we can see the without the positive regulation class 4 protein is produced insignificantly which suggests that in order to produce all classes of virulence genes, hence to cause infection positive regulation is necessary to be presents in this BvgAS system.

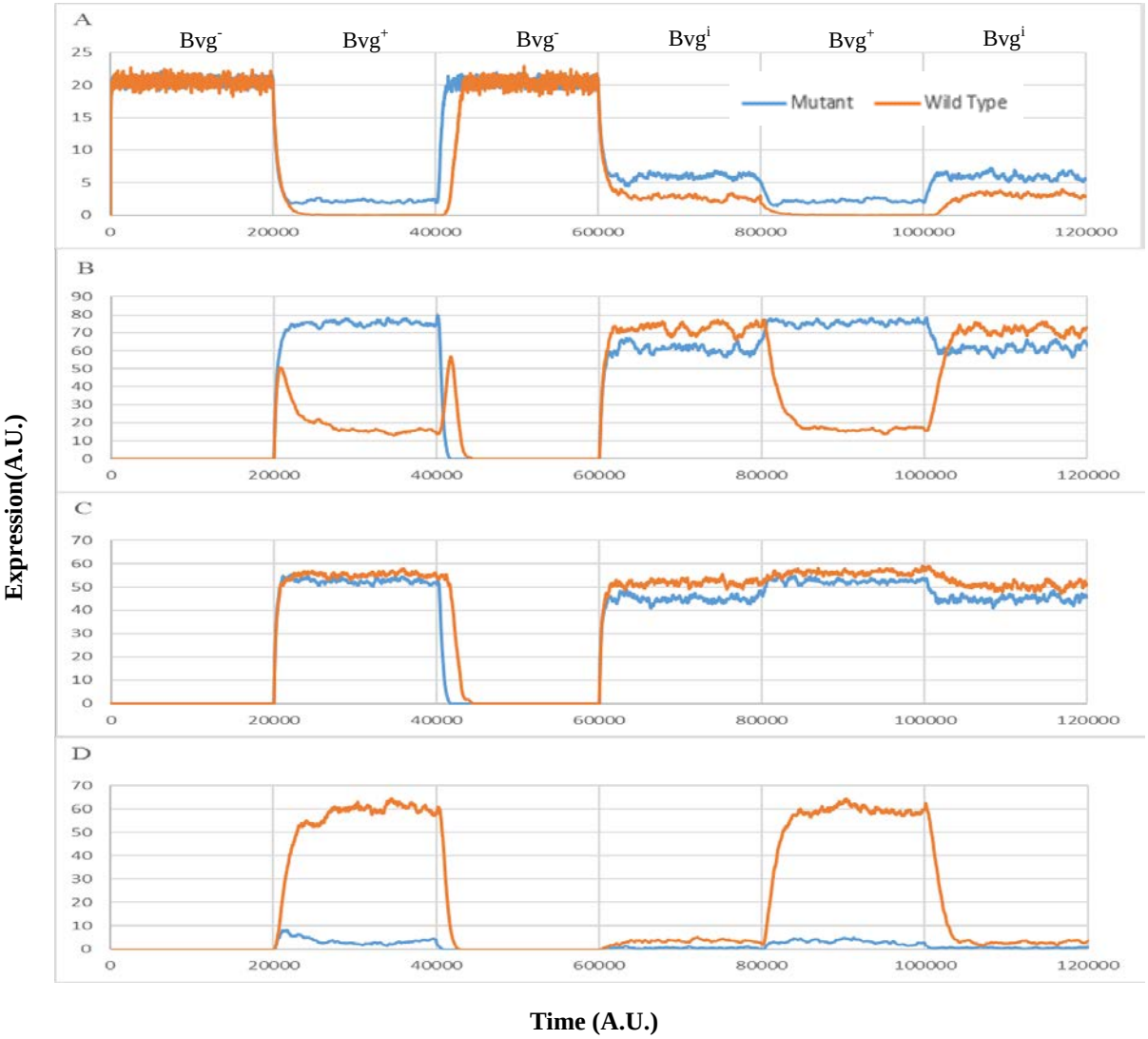

Fig.5 Comparison of dynamics of class 1 (A), class 2 (B), class 3 (C), and class 4 (D) genes in wild type (orange) and WT-FB (BvgS mutant where positive feedback is not allowed) (blue) in transitions between Bvg^-, Bvg^i, and Bvg^+ conditions.

Comparing the dynamics of mutant protein when reverse flow of phosphate group is not allowed

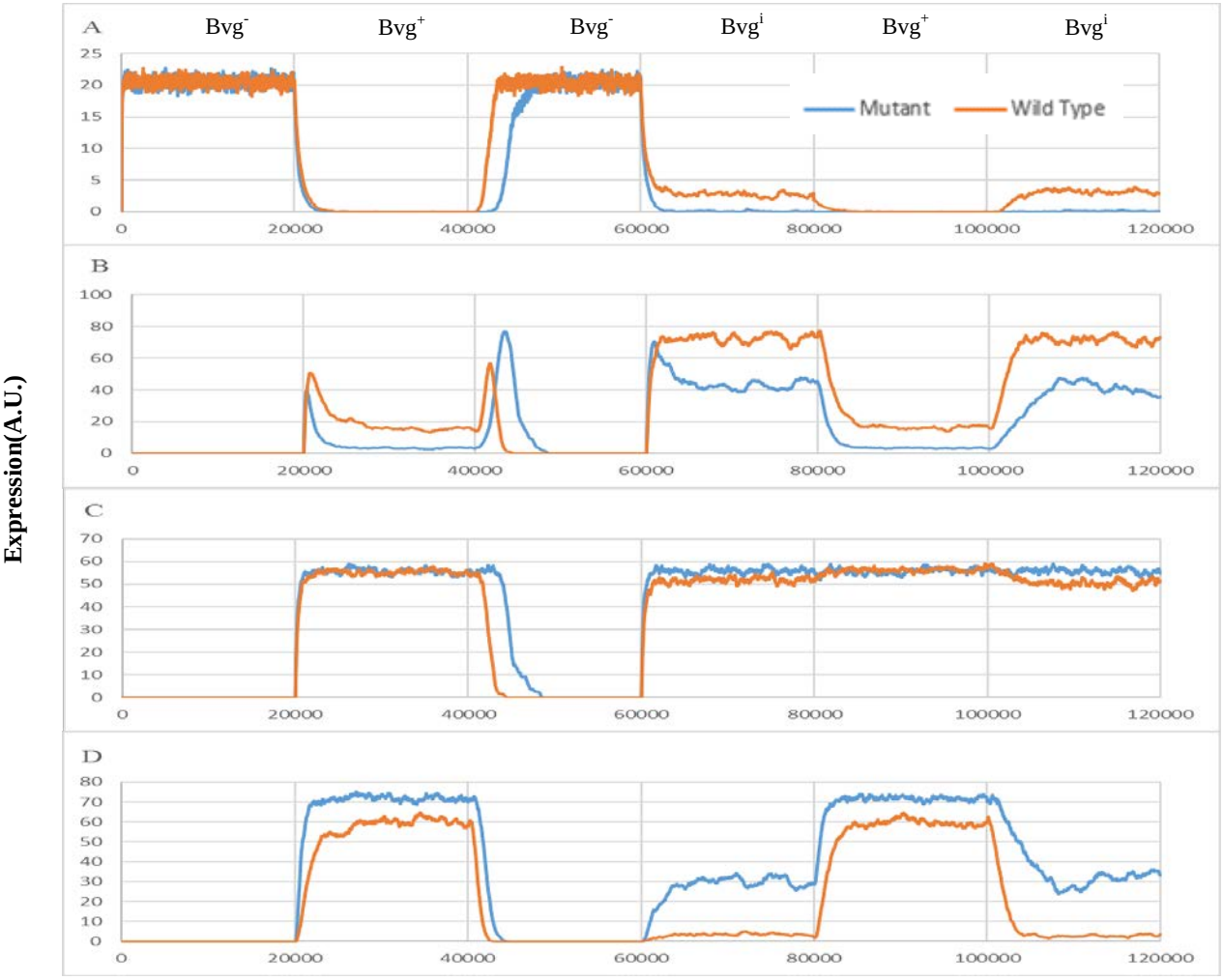

Fig.6 Comparison of dynamics of class 1 (A), class 2 (B), class 3 (C), and class 4 (D) genes in wild type (orange) and a mutant where reversible phosphorylation is not allowed (BvgS-R) (blue) in transitions between Bvg^-, Bvg^i and Bvg^+ conditions.

To test the role of reversible steps in phosphate transfer, we let k_5, k_7=0 and kept the other parameter value same as wild type. We tested the dynamics of class 1, 2, 3, and 4 genes in wild type and in mutants where the reversibility of the phosphate flow was not allowed (Fig. 6). The results show that the reversibility in the phosphorelay makes bacterium to adapt according to the environment quickly. As indicated from class 1 and class 4 protein dynamics in figure 6, when signal goes from high to low (Bvg^+ to Bvg^-) wild type are very quick to turn off the production of protein relative to wild type, indicating its superior adaptability than mutant.

Conclusion

The signal transduction system of bordetella were mathematically modelled. The modelled was then solved using a stochastic simulation method called Gillespie algorithm. We plotted the dynamics of virulence protein of wild type in order to compare the results with experimental data in literature. Later different types of mutant were developed by changing the parameter values to appropriate values and the dynamics of virulence protein were compared with mutant. We discovered that there is quite a difference in the dynamics of virulence protein between wild type and mutants. The results found by our simulation matches with the results produced in the paper.

References

1 A. van der Zee, F. Mooi, J. Van Embden and J. Musser, J. Bacteriol., 1997, 179, 6609–6617.
2 J. M. Musser, E. L. Hewlett, M. S. Peppler and R. K. Selander, J. Bacteriol., 1986, 166, 230-237.
3 Crowcroft NS and Pebody RG. “Recent developments in pertussis.” Lancet. 2006. 367(9526): 1926-36.
4 P. A. Cotter and J. F. Miller, in Principles of Bacterial Pathogenesis, ed. E. Groisman, Elsevier, 2001, pp. 619–674.
5 P. A. Cotter and A. M. Jones, Trends Microbiol., 2003, 11, 367–373.
6 A. M. Jones, P. E. Boucher, C. L. Williams, S. Stibitz and P. A. Cotter, Mol. Microbiol., 2005, 58, 700–713.
7 D. T. Gillespie, J. Phys. Chem., 1977, 81, 2340–2361.
8 D. T. Gillespie, J. Comput. Phys., 1976, 22, 403–434.

Code

```
clear all;
clc;
%PARAMETER VALUES
k1 = 2;
k1a = 10;
k1b = 5;
k2 = 15;
k3 = 1.6;
k4 = 15;
k5 = 1.7;
k6 = 2;
k6p = 0.5;
k6q = 0.5;
k7 = 1.0;
k7s = 0.5;
k7r = 0.5;
k8 = 0.01;
k9 = 0.01;
k10 = 0.025;
k11 = 4.0;
k11a = 1.5;
k12 = 1;
k12r = 0.55;
k13 = 12;
k13r = 4.5;
k13a = 1.8;
k14 = 5;
k14a = 0.1;
k15 = 6.5;
k15a = 10;
k16 = 0.05;
k17= 0.06;
k18 = 0.09;
k19 = 0.09;
k20 = 0.1;
M = 1;
N = 5000;%TIME OF ONE PHASE
k = 50;%NUMBER OF CELLS
P = zeros(N+1,10,k); %INITIAL PROTEIN LEVEL
for ijk = 1:6; %THIS LOOPIS FOR TRANSITIONING BETWEEN DIFFERENT SIGNALS
    ijk
   if ijk == 1
    S = 0;
   elseif (ijk ==2)
       S = 10;
   elseif (ijk == 3)
       S = 0;
   elseif (ijk == 4)
```

```
    S = 1.5;
  elseif(ijk == 5)
    S = 10;
  else
    S = 1.5;
  end

   for j = 1:k; %THIS LOOP IS FOR CALCULATING PROTEIN DYNAMICS FOR 50 DIFFERENT
CELLS
      j

      for n=M:N %choosing next generation
         R(1) = (k1*P(n,1,j)*(S/k1a))/((1 + (S/k1a))*(k1b + P(n,1,j))); %Autotphosphorylation form BvgS
to BvgS-P1
         R(2) = k2*P(n,2,j); %p1 to p2
         R(3) = k3*P(n,3,j); %p2 to null
         R(4) = k4*P(n,3,j);
         R(5) = k5*P(n,4,j);
         R(6) = (k6*P(n,4,j)*P(n,5,j))/((k6p + P(n,4,j)) * (k6q + P(n,5,j)));
         R(7) = (k7*P(n,1,j)*P(n,6,j))/((k7r + P(n,1,j)) * (k7s + P(n,6,j)));
         R(8) = k8*P(n,1,j);
         R(9) = k9*P(n,5,j);
         R(10) = k10*P(n,6,j);
         R(11) = (k11*((P(n,6,j)^2)))/((k11a^2) + (P(n,6,j)^2));
         R(12) = k12/(1 + ((P(n,6,j)/k12r)^2));
         R(13) = (k13*(P(n,6,j)^2))/(((k13a^2) + (P(n,6,j)^2))*(1 + ((P(n,6,j)/k13r)^2)));
         R(14) = (k14*(P(n,6,j)^2))/((k14a^2) + (P(n,6,j)^2));
         R(15) = (k15*(P(n,6,j)^4))/((k15a^4) + (P(n,6,j)^4));
         R(16) = k16*P(n,7,j);
         R(17) = k17*P(n,8,j);
         R(18) = k18*P(n,9,j);
         R(19) = k19*P(n,10,j);
         R(20) = k20;

         A = sum(R);
         flag = A*rand; %GENERATING RANDOM NUMBER BETWEEN 0 AND SUM(R)
         %CHOOSING WHICH REACTION WILL HAPPEN ACCORDING TO RANDOM NUMBER
         %GENERATED BEFORE
         if(flag < R(1))
            P(n+1,1,j) = P(n,1,j) - 1;
            P(n+1,2,j) = P(n,2,j) + 1;
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);

         elseif((flag >= R(1)) && (flag < R(1)+ R(2)))
```

```
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j) - 1;
            P(n+1,3,j) = P(n,3,j) + 1;
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);

        elseif((flag >= R(1) + R(2)) && (flag < R(1)+ R(2) + R(3)))
            P(n+1,1,j) = P(n,1,j) + 1;
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j) - 1;
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3)) && (flag < R(1)+ R(2) + R(3) + R(4)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j) - 1;
            P(n+1,4,j) = P(n,4,j) + 1;
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3) + R(4)) && (flag < R(1)+ R(2) + R(3) + R(4) + R(5)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j) + 1;
            P(n+1,4,j) = P(n,4,j) - 1;
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3) + R(4) + R(5)) && (flag < R(1)+ R(2) + R(3) + R(4) + R(5) +
R(6)))
            P(n+1,1,j) = P(n,1,j) + 1;
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j) - 1;
            P(n+1,5,j) = P(n,5,j) - 1;
```

```
            P(n+1,6,j) = P(n,6,j) + 1;
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3) + R(4) + R(5) + R(6)) && (flag < R(1)+ R(2) + R(3) + R(4) +
R(5) + R(6) + R(7)))
            P(n+1,1,j) = P(n,1,j) - 1;
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j) + 1;
            P(n+1,5,j) = P(n,5,j) + 1;
            P(n+1,6,j) = P(n,6,j) - 1;
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3) + R(5) + R(4) + R(6) + R(7)) && (flag < R(1)+ R(2) + R(3) +
R(4) + R(5) + R(6) + R(7) + R(8)))
            P(n+1,1,j) = P(n,1,j) - 1;
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j) ;
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3) + R(5) + R(4) + R(6) + R(7) + R(8)) && flag < (R(1)+ R(2) +
R(3) + R(4) + R(5) + R(6) + R(7) + R(8) + R(9)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j) - 1;
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1) + R(2) + R(3) + R(5) + R(4) + R(6) + R(7) + R(8) + R(9)) && (flag < R(1)+
R(2) + R(3) + R(4) + R(5) + R(6) + R(7) + R(8) + R(9) + R(10)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j) - 1;
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
```

```
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)) && (flag <
R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)))
            P(n+1,1,j) = P(n,1,j) + 1;
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j) + 1;
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)) && (flag
<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j) + 1;
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)) &&
(flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j) + 1;
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >= R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13))
&& (flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j) + 1;
            P(n+1,10,j) = P(n,10,j);
```

```
        elseif((flag >=
R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)) &&
(flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j) + 1;
        elseif((flag >=
R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)) &&
(flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(
16)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j)-1;
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >=
R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(16))
&&
(flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(
16)+R(17)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j)-1;
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >=
R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(16)+R
(17)) &&
(flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(
16)+R(17)+R(18)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
```

```
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j)-1;
            P(n+1,10,j) = P(n,10,j);
        elseif((flag >=
R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(16)+R
(17)+R(18)) &&
(flag<R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(
16)+R(17)+R(18)+R(19)))
            P(n+1,1,j) = P(n,1,j);
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j);
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j)-1;
        else(flag >=
R(1)+R(2)+R(3)+R(5)+R(4)+R(6)+R(7)+R(8)+R(9)+R(10)+R(11)+R(12)+R(13)+R(14)+R(15)+R(16)+R
(17)+R(18)+R(19));
            P(n+1,1,j) = P(n,1,j) + 1;
            P(n+1,2,j) = P(n,2,j);
            P(n+1,3,j) = P(n,3,j);
            P(n+1,4,j) = P(n,4,j);
            P(n+1,5,j) = P(n,5,j) + 1;
            P(n+1,6,j) = P(n,6,j);
            P(n+1,7,j) = P(n,7,j);
            P(n+1,8,j) = P(n,8,j);
            P(n+1,9,j) = P(n,9,j);
            P(n+1,10,j) = P(n,10,j);
        end

      end
      for index_3 = M:N;
      class_1(index_3,j) = P(index_3,7);
      end
   end
   P_avg = mean(P,3); %MEAN PROTEIN PRODUCTION
   for index_2 = 1:k;
      for index_1 = 1:10;
     P(N,index_1,index_2) = floor(P_avg(N+1,index_1));
      end
   end
   M = N;
   N = N + 5000;
end
%PLOTS
```

```
h(ijk) = figure();
axes('FontSize',26,'LineWidth',4.5);
plot(P_avg(:,7),'color','r','LineWidth',2.5);
hold on;
plot(P_avg(:,8),'color','g','LineWidth',2.5);
plot(P_avg(:,9),'color','b','LineWidth',2.5);
plot(P_avg(:,10),'color','y','LineWidth',2.5);
ylabel('Protein level(A.U.)','FontSize',26);
xlabel('time (A.U.)','FontSize',26);
leg = legend('class1','class2','class3','class4','Location','northwest','Fontsize',16);
legend boxoff;
print(h(ijk),'-djpeg','-r600',[num2str(ijk)]);
```