International Journal of Computational and Experimental Science and Engineering (IJCESEN)
Vol. 4-No.2 (2018) pp. 41-47
http://dergipark.gov.tr/ijcesen

ISSN: 2149-9144

Research Article

Exact Stochastic Simulation Algorithms and Impulses in Biological Systems

Derya ALTINTAN[1], Vilda PURUTÇUOĞLU[2,*]

[1]Selçuk University, Faculty of Sciences, Department of Mathematics, 42130, Konya-Turkey
[2]Middle East Technical University, Faculty of Arts and Science, Department of Statistics, 06800, Ankara-Turkey
**Both authors have equal contributions in this paper. The list of authors is written via the alphabetic order of the authors' surnames.*

* Corresponding Author : vpurutcu@metu.edu.tr
ORCID: 0000-0002-3913-9005

(First received 14 March 2018 and in final form 04 June 2018)

Keywords
Stochastic simulations
Impulses
Chemical master equations
Biological systems
Bioinformatics

Abstract: The stochastic model is the only sort of expressions which can capture the randomness of biological systems under different reactions. There are mainly three methods; direct (Gillespie), first reaction and next reaction algorithms; for implementing exact stochastic simulations in these systems. Although these algorithms are successful in explaining the natural behaviors of the systems' activation, they cannot describe the absurd changes, i.e., impulses. Moreover, the source codes in R are not available and open for all users. In this study, we produce these R codes and insert two major scenarios inside. In the application, we use biological systems with different sizes and compare their computational demands.

1. Introduction

In biochemical systems, the probabilistic manner of different biological reactions has a crucial role as it reflects the small system variability that occurs with a low frequency like the transcription of proteins. Therefore, the stochastic processes are the natural choices for the dynamic modelling of biochemical networks. The algorithms which generate these systems under stochasticity are based on the master equation [6,12,20]. These equations produce a network in two steps: (*i*) All possible reactions are listed by their probabilities of occurrences (*ii*) The system is expressed as a set of linear differential equations with a constant reaction hazard per molecule and per unit time. Although the master equations are solvable for the small number of states and for monomolecular reaction systems [10], they become intractable for large systems as the number of variables, i.e., the probabilities of states, increases quickly. To unravel this challenge, various Monte Carlo algorithms have been suggested. The general idea of these techniques is to find an approximation of the actual probability distribution by sampling repeatedly from the corresponding distribution.

There are three main methods which can stochastically generate biochemical systems based on the master equations: (*i*) direct method, which is also known as the Gillespie algorithm, (*ii*) first reaction method and (*iii*) the next reaction method, also named as the Gibson and Bruck algorithm. The Gillespie algorithm is the most common and, usually, most efficient approach, for simulating small systems. But it becomes inefficient for simulating large networks since the time step for the next reaction is taken very small such that only a single reaction can occur [1]. The first reaction method is an alternative algorithm to overcome this computational cost. Different from the Gillespie technique, it detects a presumed time for the *j*-th reaction which could occur if no other reaction occurred first. The Gibson and Bruck algorithm is faster and more efficient than these two methods, in particular, for complex biochemical networks since it can update the system locally by means of a dependency graph [5,21]. Indeed, in order to unravel the problem of computational demand, two more algorithms are also suggested. These are the optimized direct method [2] and the sorting direct method [16]. These two algorithms mainly partition the system based on slow reactions and relatively fast reactions. If the following reaction in the

simulation is slow, then the algorithms produce it via deterministic approaches, otherwise, apply stochastic algorithms. Due to this computational scheme, they are also similar to the hybrid methods [20] and need a pre-processing run to classify the speed of the reactions [16]. Therefore, their applications are not as common as the main three methods. Based on the idea of partitioning the reactions/species into subgrops that can be modelled similarly, different hybrid methods have been proposed [4,8,11].

Hence, in this study, we deal with only direct, first reaction and next reaction methods and propose two major contributions. As the first novelty, we present the R source codes of these algorithms. Indeed, the Gillespie codes have been already presented to all users within an R programme package, called GillespieSSA [17]. Whereas, the codes of the next reaction algorithm are originally written in C and the codes of the first reaction method have not been shared yet publicly. Furthermore, we also prepare certain sub-functions to detect linearly dependent species in the systems, which we call as the structural dependency [18], and to compute the net effect matrix [20], which is the difference between the product and reactant matrices. Furthermore, in the codes, we prefer the R programme since it is one of the most well-known open-source and free-downloadable softwares for the statistical calculations. On the other hand, although these algorithms are run under the constant volume and the constant temperature, they have not implemented yet if the system has impulses.

Hereby, as the second novelty in this study, we extent these three algorithms by including impulses under fixed times and fixed states. In general, the impulse, which implies the absurd variations in the dynamics of the species, is seen many systems from population to epidemic models [3, 13] and the Gillespie multiparticle method (GMP) [19] consider similar unexpected changes as the diffusion events in spatial models by using the diffusion chemical master equation [9]. Whereas the performance of their methods is highly dependent on the size of the lattice that is used to decide on the neighborhood in the diffusion. Moreover, in complex biochemical systems, the optimal size of lattices is not unique and this affects the computational speed of the algorithm and the variance of the simulated values [19]. Hereby, in this study, we insert this effect into the algorithms suitable for the general chemical master equations. Then, we apply all algorithms with/without impulses in different dimensional systems and compare them with respect to their computational costs. We use the average search depth, the average weighted degree and the central processing unit (CPU) for this purpose.

Accordingly, in the following section, we shortly present the ideas of exact stochastic simulation algorithms and describe how we include impulses in chemical master equations. In Section 3, we perform our algorithms in small, moderate and large biochemical systems and compare their findings regarding their computational demands. Finally, we conclude our results and discuss the future works in Section 4. We also prepare the R codes of all algorithms with/without impulses as the Supplementary Materials. This document is available upon request.

2. Exact Stochastic Simulation Algorithms and Impulses

There are mainly three exact stochastic simulation algorithms (SSA) to generate realisations of biochemical systems. These algorithms are exact since they are based on the chemical master equation (CME) which is represented as below.

$$\frac{\partial P(Y,t)}{\partial t} = \sum_{j=}^{r} \begin{Bmatrix} h_j(Y-\nu_j)P(Y-\nu_j,t)dt \\ -h_j(Y)P(Y,t)dt \end{Bmatrix} \quad (1)$$

In this equation, r is the number of reactions. The n-dimensional vector $Y=(Y_1,Y_2,\Lambda\ ,Y_n)$ represents the state of the system at time t, ν_j denotes the net effect of the j-th reaction. Accordingly, $h_j(Y)$ describes the hazard, also called the rate law of the reaction, which is the product of the number of distinct molecular reactant combinations available in the state Y for reaction j with stochastic rate constant c_j so that the terms $h_j(Y-\nu_j)P(Y-\nu_j,t)dt$ indicates the probability that the reaction R_j occurs over time interval $[t,t+dt)$ moving the state from $Y-\nu_j$ to Y. Hereby similar to all exact algorithms, the direct method separates the master equation by two questions: (*i*) When will the next reaction occur and (*ii*) What kind of reaction will it be? The algorithm answers them by defining a reaction probability density function on the space of continuous time random variable τ $(0\le\tau<\infty)$ and the discrete reaction indicator variable j $(j=1,\Lambda\ ,r)$ as below.

$$P(\tau,j)=\begin{cases} h_j(Y)\exp(-h_0(Y)\tau) & if \quad 0\le\tau<\infty \\ 0 & otherwise \end{cases} \quad (2)$$

where $h_j(Y) = a_j(Y)c_j$ and $h_0(Y) = \sum_{j=1}^{r} h_j(Y) = \sum_{j=1}^{r} a_j(Y)c_j$. Furthermore, $a_j(Y)$ and c_j are the number of the distinct molecular combinations of the given state Y and the reaction rate constant of the j-th reaction, respectively. Even though this algorithm is the most common and usually the most efficient simulator based on the CME, its application is limited by spatially homogeneous, thermodynamically equilibrated systems where the nonreactive molecules collide more frequently than reactive once [6]. Therefore, it is difficult to perform these algorithms for simulating the localization or the spatial heterogeneity [1, 20]. Additionally, though the algorithm works well for simulating small systems, it is inefficient for developing realistic complex models since the time step for the next reaction, which is generated from the exponential distribution with a rate $h_0(Y)$, is taken very small such that only a single reaction can occur in a given time interval [1, 20].

The first reaction method [7] is different from the direct method in the sense that the latter produces the reaction indicator *j* and the continuous time τ directly, however, the former generates a presumed time τ_j for the reaction *j* which would occur if no other reaction occurred first. Moreover, although both methods use the same probability distribution in order to choose *j* and τ, the first reaction method executes *r* reactions per iteration. Thereby, the method generates the *r* number of time variables τ_j $(j = 1, \Lambda, r)$ both for updating the hazard function $h_j(Y) = a_j(Y)c_j$ and for finding the smallest τ_j in each iteration.

The Gibson-Bruck algorithm [5], also called the next reaction method, is an exact algorithm which is faster and more efficient than the direct algorithm, especially, for complex systems. The algorithm develops an efficient computation by storing both the time step τ_j $(j = 1, \Lambda, r)$ and the hazard function $h_j(Y)$, rather than $h_j(Y)$ alone, for each reaction and by recalculating $h_j(Y)$ according to a dependency graph ζ for the system. The graph shows the set of species that is affected by the value of $h_j(Y)$ if the *j*-th reaction is fired. This can be done from the algorithm by connecting the node *j* to the set of all children of the node *j* whose hazards should be updated according to *j*. This representation speeds up the simulation since it enables us to update the system locally. On the other hand, the connections between these local subsets are maintained by another graph, called the indexed priority queue. That graph contains two sorts of information. The first one is a tree structure of an ordered pairs of the form (j, τ_j) where *j* stands for the reaction and τ_j is the presumed time when the reaction *j* occurs and the second one is an index structure whose element *j* is a pointer of the location in the tree which contains (j, τ_j). In the tree structure of the queue, each parent has a lower τ_j than either of its children [20].

In this study, initially, we extend CME given in Equation (1) by inserting impulses at the fixed time and the fixed state. The following expression shows the impulsive CME under this condition

$$\Delta Y|_{t=\lambda_m(Y)} = A(Y,t) \ , \quad (3)$$

where $\lambda_m(Y)$ denotes the *m*-dimensional $(m = 1,2,\Lambda)$ vector of predefined time points at which the impulse fires by a function $A(Y,t)$. Hence, if the trajectory of the impulse function intersects the entry of the time point $t = \lambda_m(Y)$ for the next time step τ_j, then the impulse will occur and the trajectory of the system will jump to a new state via $Y := Y + A(Y(\lambda_m))$. Later, the system continues its movement by the selected exact algorithm until the trajectory of the system intersects another entry in the *m*-dimensional vector $\lambda_m(Y)$. On the other hand, if the impulse is seen at the fixed state, rather than fixed time, then Equation (3) will be revised by describing a new *m*-dimensional vector of predefined state values for the selected species in the system at which the impulse fires by a function $A(Y,t)$. In the following section, we perform these three exact algorithms with their impulsive versions in different dimensional networks and compare their computational efficiencies.

On the other side, the original R codes of all algorithms with/without impulses are presented as Supplementary Materials.

In this document, we also prepare an example set to describe how an ordinary simulation and a simulation with impulses can be implemented by these functions.

3. Applications

In this section, we compare the computational costs of the direct, first reaction and the next reaction

methods in terms of the average search depth, average weighted degree and CPU as defined below [2]. Here, we apply them to the Lotka-Volterra, PKC and the JAK/STAT systems as the toy, moderately large and large dimensional systems, respectively. The idea is to consider each system with or without impulses and to compare the performance of the algorithms in both models to see the effect for computing impulses. Below, we describe our comparison criteria in details.

1. *Average search depth (S):* This measure denotes the average number of operations to find the index of the firing reaction by using $S=\sum_{j=1}^{r} jk_j / \sum_{j=1}^{r} k_j$ where r denotes the number of reactions in the system as stated beforehand and k_j is the number firing times of the *j*-th reaction.
2. *Average weighted degree (D):* This measure is the average degree for the dependency graph of the system which tells the reactions whose hazards are affected when a given reaction is fired. It is computed by $D=\sum_{j=1}^{r} d_j k_j / \sum_{j=1}^{r} k_j$ presenting the average number of operations to find the index of the firing reaction. Here, d_j stands for the total number of reactions affected by the firing of the *j*-th reaction.
3. *Central processing unit time (CPU):* This measure indicates the programme run-time and represents a quantitative value of the busyness of the system.

3.1 Lotka-Volterra Model

The Lotka-Volterra model [20], also called the prey-predator system, is a well-known system that describes the population dynamics of two competing species, namely, Y_1 and Y_2 representing preys and predators, respectively. In the analyses, the initial population of species is taken as $Y(0)=(Y_1,Y_2)=(4,10)$ and the system is analyzed in $t\in[0,5]$. After calculating CPU, *S* and *D*, we add impulses to the system and compute the same quantities. Here, we fix the number of preys to 4 when the number of preys is greater than 4 which will give CME in the following form.

$$\frac{\partial P((Y_1,Y_2),t)}{\partial t} = k_1(Y_1-1)P((Y_1-1,Y_2),t)$$
$$-k_1Y_1P((Y_1,Y_2),t)$$
$$+k_2(Y_1+1)(Y_2-1)P((Y_1+1,Y_2-1),t)$$
$$-k_2Y_1Y_2P((Y_1,Y_2),t)+k_3(Y_2+1)P((Y_1Y_2+1),t)$$
$$-k_3Y_2P((Y_1Y_2)t),$$

$$\Delta|_{t+\tau}=\begin{cases}5 & if \quad Y_l(\tau)>5\\ 0 & otherwise\end{cases},$$

$$Y(0)=(4,10).$$

The reactions, reaction constants, net effect and hazard of each reaction for this system can be seen in Table 1 as the example of the contruction of the systems' simulation.

Table 1. *Reactions, reaction constants, net effect (ν_j) and hazard $h_j(Y)$ of each reaction $j=1,2,3$ for the Lotka-Volterra system.*

Reactions	Reaction Constants	Net Effects	Hazards
$R_1: Y_1 \to 2Y_1$	$k_1=1.0$	$\nu_1=(1,0)$	$h_1(Y)=k_1Y_1$
$R_2: Y_1+Y_2 \to 2Y_2$	$k_1=0.1$	$\nu_2=(-1,1)$	$h_2(Y)=k_1Y_1Y_2$
$R_3: Y_2 \to 0$	$k_1=0.1$	$\nu_3=(0,-1)$	$h_3(Y)=k_3Y_2$

3.2 Protein Kinase C Signalling Pathway

As a second model, we consider the protein kinase C (PKC) signaling pathway which plays an important role for regulating several neural functions such as the long-term potentiation and the depression [15]. The system is described by 14 species and 10 reversible reactions and presents the production of the active PKC. The biological details of reactions and initial values of the species can be found in [15].

After obtaining the computational cost of the original model without impulses, we consider the PKC system with impulses by adding 1e-16nM molecules to PKCbasal* in the system when the number of molecules of this species becomes 0. Finally, the simulations with/without impulses run for time interval $t\in[0,5]$ and the results of both models are presented in Table 2.

3.3 JAK/STAT Signalling Pathway

The janus kinase signal transduction pathway (JAK/STAT), which transmits the extracellular stimuli to the gene in the target cells without second messengers, is the largest system used in our application. In the biological sense, it is the major signalling transaction system which is activated by the type I interferon (IFN) and regulates cytokine-dependent gene expressions and growth factors of mammals. The activation flow of this pathway begins with an external ligand binding to the receptor which in turn activates JAK. Then, activated JAK phosphorylates activators of STAT and finally, active STAT enters the nucleus and binds to a specific region in DNA which in turn translates this particular part to result in a certain type of proteins. Hereby, in the biological description of this system, 41 reactions with 38 species are used.

In the analyses, initial concentrations of each species are set to 100 molecules and the reactions as well as the associated rates are taken from the study of Mailwald et. al (2010) [14].

To construct the JAK/STAT system with impulses, we add 10 molecules to JAK in every 100-step and simulate the reactions for $t \in [0,5]$. It must be noted that in previous models, we add abrupt amounts to the species when a specific concentration is reached by a given species. But in this model, instead of concentrations, we consider specific times to change the concentration of JAK. The associated findings are listed in Table 2.

Thereby, from the comparison of all results in Table 2, we observe that although the next reaction method is supposed to be the most efficient algorithm in terms of computational demand for the selected pathways, there is no sharp change among all these methods and different from the expectation, the direct method is generally more efficient in computational time. This findings show that, indeed, the performance of the algorithms is highly dependent on the system of interest. Accordingly, the next reaction method cannot guarantee gain in computational demand under all conditions, even though it has been developed for this purpose. Moreover, in this study, all the algorithms are originally written in R and from the tabulated outputs, we see that the direct method is still fast, in particular, when we include impulses and it is still computationally the most friendly and simplest algorithm. In order to understand the reason behind this conclusion, we compare *D*, *S* versus CPU simultaneously. We detect that the species in each system are highly dependent on each other in such a way that from the entries of *D*, LV shows both 2 species, PKC indicates 5 species over totally 11 species and JAK-STAT reports 7 species over totally 38 species that are affected by firing each reaction on average. Under such conditions, running the dependency graph in the next reaction method can cause an additional computational demand in the algorithm since majority of the species already change their states by firing of each reaction and hereby, there is no necessity to control the interaction of species at the end of every reaction. Additionally, when we compare *S* terms, it is seen that all the three algorithms almost use the same number of steps to decide on the next reaction. Hence, under such equality, CPU of the direct method is typically less than other two methods. We detect the same findings from the outputs under impulses. As a result, we conclude that the users can choose the most appropriate algorithm for their applications according to the reaction lists of their quasi true networks. If the reactions are composed on a few common species, then, the direct method can be preferable against others to gain in CPU. On the other hand, if the quasi reaction list has many species and each reaction consists of only common number of species, the next reaction method can be more suitable among alternatives due to the advantage of its dependency graph.

From the comparison of all results, we observe that all methods have the same accuracy and computational demand when they are coded in the same programme language R. Hence, we conclude that the major advantage of the next reaction algorithm may not be due to the dependency graph in the calculation, rather, the programme language of the original algorithm, which is the C compiled language. On the contrary, R is an interpreted language whose source codes are not directly run by the target machine although the development of codes are easier. Furthermore, when we evaluate their performances under impulses, it is found that the firing of impulses in all exact methods does not cause any significant computational demand. Moreover, as detected for the non-impulsive simulations, there is no significant difference in the final *S*, *D* and CPU measures meaning that the average number of operations to detect the next reaction and its associated siblings are almost the same for all algorithms.

4 Conclusion

In this study, we have written the R codes of the direct, first and the next reaction stochastic simulation algorithms, designed for the biological networks.

We have prepared them as a user-friendly functional form so that these open-source codes can be utilized for all researchers in the field of systems biology and bioinformatics. Additionally, we have extended these algorithms by adding impulsive functions which can change the trajectory of the system based on the chemical master equations at the fixed time and states. The key and very simple idea behind these algorithms is that we run very well-known direct, first reaction and next reaction methods until the impulse condition is reached which will be checked by using an if clause. When the impulse condition is reached, we update the system state depending on the impulse under consideration. We apply this process recurtsively until the end of the time interval of interest. By this way, we can also implement these algorithms to successfully capture the absurd changes in the systems which are observed in many epidemic and population models. Then, we have performed these exact methods in different dimensional pathways and compared their computational demands with various measures in order to show the performance of all R codes and the algorithm. The outcomes have indicated similarity in computational costs of all algorithms with/without impulses. From the results, we have detected that if the system is small and its biological description is not structurally highly dependent, then, the direct method can be favourable due to its simplicity in calculations. But if the reactions in the system do not have common species very much, i. e., *D*, is small on average, the next reaction method can be computationally more efficient.

Table 2. *CPU (central processing unit) time, S (average search depth), D (average weighthed degree) values for the Lotka-Volterra, PKC and JAK/STAT systems without impulses (WI) and with impulses (I) at fixed time (states) for the given stochastic simulation algorithms.*

System	Method	Condition	CPU	*S*	*D*
Lotka Volterra	Direct	WI	40.816	2.083	2.416
	Direct	I-Fixed State	40.805	1.791	2.375
	First React.	WI	42.337	2.031	2.406
	First React.	I-Fixed State	42.325	1.933	2.400
	Next React.	WI	42.831	2.031	2.406
	Next React.	I-Fixed State	42.843	2.052	2.421
PKC	Direct	WI	47.935	2.000	5.000
	Direct	I-Fixed State	47.964	1.500	5.000
	First React.	WI	48.609	2.000	5.000
	First React.	I-Fixed State	48.625	1.556	5.000
	Next React.	WI	49.771	2.000	5.000
	Next React.	I-Fixed State	49.792	2.000	5.000
JAK/ STAT	Direct	WI	15.046	21.537	7.420
	Direct	I-Fixed Time	9.330	21.263	7.311
	First React.	WI	22.453	21.951	7.055
	First React.	I-Fixed Time	29.333	21.014	7.067
	Next React.	WI	35.807	21.600	7.109
	Next React.	I-Fixed Time	39.542	21.451	7.061

But in general, we consider that our open-source codes and extended version of the algorithms can be helpful for the researchers who need to generate complex biological systems in all dimensions easily. As the extension of this study, we suggest to insert the idea of the impulses in approximate simulation algorithms and adopt our codes for these methods. Furthermore, we consider to insert the bifurcation graph in exact simulation algorithm in order to generate all possible stochastic scenarios of the systems in a single run.

Acknowledgement

The authors would like to thank the AGEP grant (project no: BAP-08-11-2014-007) of the Middle East Technical University for its support and V. Purutçuoğlu thanks to Prof. Dr. Ernst Wit for his helpful discussion in coding of the Gillespie function.

References

[1] J. M. Bower, H. Bolouri, "Computational Modelling of Genetic and Biochemical Networks", MIT Press, 2000.

[2] Y. Cao, H. Li, L. Petzold. "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems", Journal of Chemical Physics, 121 (9) (2004) 4059-4067.

[3] A. d'Onofrio, "Stability properties of pulse vaccination strategy in SEIR epidemic model", Mathematical Biosciences, 179 (2002) 52-72.

[4] A. Ganguly, D. Altıntan, and H. Koeppl, "Jump-diffusion approximation of stochastic reaction dynamics: Error bounds and algorithms", Multiscale Modeling & Simulation,, 13(4) (2015) 1390–1419.

[5] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels", Journal of Physical Chemistry A, 104 (9) (2000) 1876-1889.

[6] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions", Journal of Physical Chemistry, 81 (1977) 2340-2361.

[7] D. T. Gillespie, "A rigorous derivation of the chemical master equation", Physica A, 188 (1992) 404-425.

[8] J. Hasenauer, V.Wolf, A. Kazeroonian, and F. J. Theis, "Method of conditional moments (MCM) for the chemical master equation : A unified framework for the method of moments and hybrid stochastic-deterministic models.", Journal of Mathematical Biology 69 (3) (2014) 687–735.

[9] S. A. Isaacson, "The reaction diffusion master equation as an asymptotic approximation of diffusion to a small target", SIAM Journal of Applied Mathematics, 70 (1) (2009) 77-111.

[10] T. Jahnke, W. Huisinga, W. "Solving the chemical master equation for monomolecular reaction

systems analytically", Journal of Mathematical Biology 54 (1) (2006) 1–26.

[11] T. Jahnke and M. Kreim, "Error bound for piecewise deterministic processes modeling stochastic reaction systems", Multiscale Modeling & Simulation, 10 (2012) 1119–1147.

[12] N. G. van Kampen, "Stochastic processes in physics and chemistry", Amsterdam; New York, Elsevier North-Holland, 1981.

[13] Z. Lu, X. Chi, L. Chen, "The effect of constant and pulse vaccination on SIR epidemic model with horizontal and vertical transmission", Mathematical and Computer Modelling, 36 (2002) 1039-1057.

[14] T. Mailwald, A. Schneider, H. Busch, S. Sahle, N. Gretz, T. S. Weiss, U. Kummer, U. Klinqmüller, "Combining theoretical analysis and experimental data generation reveals irf9 as a crucial factor for accelerating interferon-induced early antiviral signaling", FEBS Journal 277 (22) (2010) 4741-4754.

[15] T. Manninen, E. Makiraatikka, A. Ylipaa, A. Pettinen, K. Leinonen, M. L. Linne, "Discrete stochastic simulation of cell signalling: comparison of computational tools", 28th IEEE EMBS Annual International Conference, New York City-USA pp. 2013-2016 (2006). DOI: 10.1109/IEMBS.2006. 260023

[16] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, N. F. Samatova, "The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior", Computational Biology and Chemistry, 30 (1) (2006) 39-49.

[17] M. Pineda-Krch, "GillespieSSA: implementing the stochastic simulation algorithm in R", Journal of Statistical Software, 25 (12) (2008) 1-18.

[18] V. Purutçuoğlu, E. Wit, "Bayesian inference for the MAPK/ERK pathway by considering the dependency of the kinetic parameters", Bayesian Analysis, 3 (2008) 851-886.

[19] J. V. Rodriguez, J. A. Kaandorp, M. Dobrzynski, and J. G. Blom, "Spatial stochastic modelling of the phosphoenolpyruvate-dependent phosphotrans ferase (PTS) pathway in Escherichia coli", Bioinformatics, 22 (15) (2006) 1895-1901.

[20] D. J. Wilkinson, "Stochastic modelling for systems biology", Chapman & Hall/CRC Mathematical and computational biology series. Boca Raton, FL: Taylor & Francis, 2006