EPJ Web of Conferences **173**, 05006 (2018) https://doi.org/10.1051/epjconf/201817305006
Mathematical Modeling and Computational Physics 2017

The Automation of Stochastization Algorithm with Use of SymPy Computer Algebra Library

Anastasya Demidova[1,⋆], *Migran* Gevorkyan[1,⋆⋆], *Dmitry* Kulyabov[1,2,⋆⋆⋆], *Anna* Korolkova[1,⋆⋆⋆⋆], and *Leonid* Sevastianov[1,3,†]

[1] *Department of Applied Probability and Informatics, Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya str., Moscow, 117198, Russia*
[2] *Laboratory of Information Technologies, Joint Institute for Nuclear Research, 6, Joliot-Curie St., Dubna, Moscow region, 141980, Russia*
[3] *Bogoliubov Laboratory of Theoretical Physics, Joint Institute for Nuclear Research, 6, Joliot-Curie St., Dubna, Moscow region, 141980, Russia*

Abstract. SymPy computer algebra library is used for automatic generation of ordinary and stochastic systems of differential equations from the schemes of kinetic interaction. Schemes of this type are used not only in chemical kinetics but also in biological, ecological and technical models. This paper describes the automatic generation algorithm with an emphasis on application details.

1 Introduction

To describe models, where the intensity of the interaction between components depends on the concentration levels of the components, one can use the schemes of kinetic interactions [1–4]:

$$\sum_{i=1}^{n} N_i^s X_i \underset{k_s^-}{\overset{k_s^+}{\longleftrightarrow}} \sum_{i=1}^{n} M_i^s X_i, \qquad s = 1, \ldots, S.$$

Here X_i represents the quantity of i-th component, the matrices $\mathbf{M} = [M_i^s]$ and $\mathbf{N} = [N_i^s]$ are *system state matrices* which determine the number of interacting components of the system at each stage of the reaction. k_s^+ and k_s^- are *coefficients of interaction* in direct and reverse directions, n is system dimension, S is the number of interactions.

Kinetic schemes are often used to model chemical reactions, biological and ecological systems. From these diagrams one can derive a system of ordinary differential equations (ODEs) which gives the evolution of variables $X_i(t)$ over time line.

Our research team has generalized the method of stochastisation of deterministic models [5]. This method allows to obtain the stochastic model, based on implicit stochastic properties of the system.

⋆ e-mail: demidova_av@rudn.university
⋆⋆ e-mail: gevorkyan_mn@rudn.university
⋆⋆⋆ e-mail: kulyabov_ds@rudn.university
⋆⋆⋆⋆ e-mail: korolkova_av@rudn.university
† e-mail: sevastianov_la@rudn.university

All necessary information for the stochastisation process can be extracted from the kinetic interaction schemes. For multi-dimensional systems the stochastisation method requires a large amount of routine work. However, it is possible to program all the steps of the method and obtain the drift vector and the diffusion matrix. We carry out the implementation of the algorithm by using the `Python` language (version 3) and `SymPy` library for computer algebra.

2 An algorithm of obtaining ODE and SDE systems

The initial data are the system state matrices $\mathbf{M}$, $\mathbf{N}$ and the coefficients of the interaction between system components k_s^+ and k_s^-. At the first algorithm stage one has to find a matrix $\mathbf{R} = \mathbf{M}^T - \mathbf{N}^T$, the columns of which are denoted by vectors $\mathbf{r}^j$. Next, one has to calculate the auxiliary values s_j^+ and s_j^-:

$$s_j^+(\mathbf{x}) = k_j^+ \prod_{i=1}^{n} \frac{x^i!}{(x^i - N_i^j)!} = k_j^+ \times \frac{x^1!}{(x^1 - N_1^j)!} \times \frac{x^2!}{(x^2 - N_2^j)!} \times \cdots \times \frac{x^n!}{(x^n - N_n^j)!},$$
$$s_j^-(\mathbf{x}) = k_j^- \prod_{i=1}^{n} \frac{x^i!}{(x^i - M_i^j)!} = k_j^+ \times \frac{x^1!}{(x^1 - M_1^j)!} \times \frac{x^2!}{(x^2 - M_2^j)!} \times \cdots \times \frac{x^n!}{(x^n - M_n^j)!}.$$

Using these values one can obtain *the drift vector* $\mathbf{f}(\mathbf{x})$ and *the diffusion matrix* $\mathbf{G}(\mathbf{x})$. The drift vector is calculated by the formula:

$$\mathbf{f}(\mathbf{x}) = \mathbf{r}^1(s_1^+(\mathbf{x}) - s_1^-(\mathbf{x})) + \mathbf{r}^2(s_2^+(\mathbf{x}) - s_2^-(\mathbf{x})) + \cdots + \mathbf{r}^s(s_s^+(\mathbf{x}) - s_s^-(\mathbf{x})),$$

and it can be used to write out the system of ODEs

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}).$$

To get the system of stochastic differential equations (SDEs) in Itô form, it is necessary to calculate also the diffusion matrix $\mathbf{G}(\mathbf{x})$:

$$\mathbf{G}(\mathbf{x}) = \mathbf{r}^1(\mathbf{r}^1)^T(s_1^+(\mathbf{x}) - s_1^-(\mathbf{x})) + \mathbf{r}^2(\mathbf{r}^2)^T(s_2^+(\mathbf{x}) - s_2^-(\mathbf{x})) + \cdots + \mathbf{r}^s(\mathbf{r}^s)^T(s_s^+(\mathbf{x}) - s_s^-(\mathbf{x})).$$

After that it is possible to obtain the system of Itô SDE:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x})\,\mathrm{d}t + \sqrt{\mathbf{G}(\mathbf{x})}\mathbf{W}(t),$$

where $\mathbf{W}$ is the n-th dimensional Wiener process.

2.1 Brief description of the software implementation

For the automation of the above method we used `Python` [6] with `SymPy` [7] module. The program takes symbolic vectors X and K and numeric matrices N and M on input, and gives the drift vector and the diffusion matrix on output. For numerical matrices we use `NumPy` arrays [7]. As interface for our module we provide two functions:

```
f = drift_vector(X, K, N, M),
G = diffusion_matrix(X, K, N, M).
```

These functions return the symbolic drift vector and diffusion matrix. It is possible to use built-in `SymPy` methods to convert $\mathbf{f}$ and $\mathbf{G}$ to different formats, such as LaTeX formula notation or expression in any supported programming language. `SymPy` provides support for a large number of programming languages ranging from popular ones such as C/C++, Fortran, Java etc., to new growing languages such as Julia language.

3 Examples and program verification

Let us consider two examples. The first example is the Verhulst model describing the growth of population in conditions of limited resources (logistic curve), and the second example is the model of brusselator [8]. Ilya Prigogine has proposed this model as a simple example of Belousov–Zhabotinsky reaction type. These models are well known and we use them to test our program and to show algorithm in action.

3.1 Verhulst model

The model of logistic growth is described by the following scheme:

$$X \xrightarrow{k_1} 2X, \qquad X + X \xrightarrow{k_2} X.$$

In this case, the matrices $\mathbf{N}$ and $\mathbf{M}$ turn into vectors: $N_1^1 = 1$, $N_1^2 = 2$, $M_1^1 = 2$, $M_1^2 = 1$, and vectors $\mathbf{r}$ turn into scalars $r^1 = 1$, $r^2 = -1$. Because the reaction is not reversible, it is enough to compute only s^+:

$$s_1^+ = k_1 \cdot \frac{x!}{(x-1)!} = k_1 x,$$
$$s_2^+ = k_2 \cdot \frac{x!}{(x-2)!} = k_2 x(x-1) = k_2 x^2 - k_2 x.$$

$$f(x) = 1 \cdot k_1 x - 1(k_2 x^2 - k_2 x) = k_1 x + k_2 x - k_2 x^2,$$
$$g(x) = 1 \cdot k_1 x + (k_2 x^2 - k_2 x) = k_1 x - k_2 x + k_2 x^2.$$

Usually the term $k_2 x$ is discarded due to the fact that the x^2 value is large and the x value is small:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = k_1 x - k_2 x^2,$$

and

$$\mathrm{d}x(t) = (k_1 x - k_2 x^2)\,\mathrm{d}t + \sqrt{k_1 x + k_2 x^2}\,\mathrm{d}W\,.$$

3.2 The Brusselator model

The Brusselator model is completely determined by the following scheme:

$$A \xrightarrow{k_1} X, \qquad 2X + Y \xrightarrow{k_2} 2X,$$
$$B + X \xrightarrow{k_3} Y + D, \qquad X \xrightarrow{k_4} E.$$

Besides the usual reagents X and Y, the inexhaustible reagents A, B, D and E are also involved into reaction. Their number is considered to be time invariant. They cannot be taken into account while constructing the matrices M and N. However, even if we include them in M and N we will obtain correct results. Let use to demonstrate that fact.

$$\mathbf{N} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x \\ y \\ a \\ b \\ d \\ e \end{bmatrix}, \qquad \mathbf{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix}.$$

Based on the above matrices, our `Python` functions return the following drift vector and diffusion matrix:

$$\begin{pmatrix} ak_1 - bk_3x + k_2xy\,(x-1) - k_4x \\ bk_3x - k_2xy\,(x-1) \\ -ak_1 \\ -bk_3x \\ bk_3x \\ k_4x \end{pmatrix},$$

$$\begin{bmatrix} ak_1 + bk_3x + k_2xy\,(x-1) + k_4x & -bk_3x - k_2xy\,(x-1) & -ak_1 & bk_3x & -bk_3x & -k_4x \\ -bk_3x - k_2xy\,(x-1) & bk_3x + k_2xy\,(x-1) & 0 & -bk_3x & bk_3x & 0 \\ -ak_1 & 0 & ak_1 & 0 & 0 & 0 \\ bk_3x & -bk_3x & 0 & bk_3x & -bk_3x & 0 \\ -bk_3x & bk_3x & 0 & -bk_3x & bk_3x & 0 \\ -k_4x & 0 & 0 & 0 & 0 & k_4x \end{bmatrix}.$$

Now it is possible to write out the ODE system:

$$\begin{cases} \dot{x} = ak_1 - bk_3x + k_2xy(x-1) - k_4x, \\ \dot{y} = bk_3x - k_2xy(x-1). \end{cases}$$

4 Conclusion

In this paper, we described an algorithm of obtaining ODE and SDE systems from the interaction schemes. Taking into account the inhibitory interactions, will result in algorithm improvement.

Acknowledgement

This work is partially supported by RFBR grants No's 15-07-08795 and 16-07-00556. The publication was also financially supported by the "RUDN University Program 5-100".

References

[1] M. Hnatič, E.G. Eferina, A.V. Korolkova, D.S. Kulyabov, L.A. Sevastyanov, EPJ Web of Conferences **108**, 58 (2015), [arXiv]1603.02205

[2] E.G. Eferina, M. Hnatich, A.V. Korolkova, D.S. Kulyabov, L.A. Sevastianov, T.R. Velieva, in *Distributed Computer and Communication Networks. DCCN 2016. Communications in Computer and Information Science*, edited by V.M. Vishnevskiy, K.E. Samouylov, D.V. Kozyrev (Springer, Cham, 2016), Vol. 678, pp. 483–497

[3] A.V. Demidova, M.N. Gevorkyan, T-Comm **8**, 43 (2014)

[4] M. Hnatich, J. Honkonen, Physical Review E **61**, 3904 (2000)

[5] A. Demidova, M.N. Gevorkyan, A.D. Egorov, D.S. Kulyabov, A.V. Korolkova, L.A. Sevastyanov, Bulletin of Peoples' Friendship University of Russia. Series Mathematics. Information sciences. Physics. pp. 71–85 (2014)

[6] G. Rossum, Tech. rep., Amsterdam, The Netherlands, The Netherlands (1995)

[7] E. Jones, T. Oliphant, P. Peterson et al., *SciPy: Open source scientific tools for Python* (2001)

[8] C.W. Gardiner, *Handbook of Stochastic Methods: for Physics, Chemistry and the Natural Sciences* (Springer Series in Synergetics, 1985)