Automation of stochastization algorithm with use of SymPy computer algebra library

Demidova A. V. [1] Gevorkyan M. N. [1]
Sevastianov L. A.[1,2] Kulyabov D. S.[1,3]

[1] Department of Applied Probability and Informatics, RUDN University (Peoples' Friendship University of Russia)

[2] Bogoliubov Laboratory of Theoretical Physics Joint Institute for Nuclear Research 6 Joliot-Curie, Dubna

[3] Laboratory of Information Technologies Joint Institute for Nuclear Research 6 Joliot-Curie, Dubna.

Mathematical Modeling and Computational Physics, 3–7 Julie, 2017

Content

Main goal

To automate translation algorithm of kinetic reactions schemes to ordinary and stochastic equations systems.

Kinetic reactions schemes

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

- X_i — concentration i-th component;
- $\mathbf{M} = [M_i^s]$ и $\mathbf{N} = [N_i^s]$ — matrices of the system state;
- k_s^+ и k_s^- — the coefficients of interaction in direct and reverse reaction process.

Biological and ecological systems (population dynamics, epidemiological models), chemical reactions (e.g. chemical clocks).

The algorithm for obtaining of ODE and SDE systems

$$\mathbf{x} = \begin{pmatrix} x^1 \\ x^2 \\ x^3 \\ \vdots \\ x^n \end{pmatrix}, \ \mathbf{N} = \begin{bmatrix} N_1^1 & N_2^1 & N_3^1 & \dots & N_n^1 \\ N_1^2 & N_2^2 & N_3^2 & \dots & N_n^2 \\ N_1^3 & N_2^3 & N_3^3 & \dots & N_n^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ N_1^s & N_2^s & N_3^s & \dots & N_n^s \end{bmatrix}, \ \mathbf{M} = \begin{bmatrix} M_1^1 & M_2^1 & M_3^1 & \dots & M_n^1 \\ M_1^2 & M_2^2 & M_3^2 & \dots & M_n^2 \\ M_1^3 & M_2^3 & M_3^3 & \dots & M_n^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_1^s & M_2^s & M_3^s & \dots & M_n^s \end{bmatrix}$$

N_i^j and M_i^j the concentration of the component X^i in left and right sides.

$$\mathbf{R} = \mathbf{M}^T - \mathbf{N}^T = \begin{bmatrix} M_1^1 - N_1^1 & M_1^2 - N_1^2 & M_1^3 - N_1^3 & \dots & M_1^s - N_1^s \\ M_2^1 - N_2^1 & M_2^2 - N_2^2 & M_2^3 - N_2^3 & \dots & M_2^s - N_2^s \\ M_3^1 - N_3^1 & M_3^2 - N_3^2 & M_3^3 - N_3^3 & \dots & M_3^s - N_3^s \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_s^1 - N_s^1 & M_s^2 - N_s^2 & M_s^3 - N_s^3 & \dots & M_s^s - N_n^s \end{bmatrix}$$

The algorithm for obtaining of ODE and SDE systems

Let us denote the columns of the matrix **R** as separate vectors **r**

$$\mathbf{r}^j = \begin{bmatrix} M^j_1 - N^j_1 \\ M^j_2 - N^j_2 \\ M^j_3 - N^j_3 \\ \vdots \\ M^j_n - N^j_n \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \mathbf{r}^1 & \mathbf{r}^2 & \dots & \mathbf{r}^s \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

The algorithm for obtaining of ODE and SDE systems

First we have to compute the following coefficients

$$s_j^+(\mathbf{x}) = k_j^+ \prod_{i=1}^{n} \frac{x^i!}{(x^i - N_i^j)!} = k_j^+ \cdot \frac{x^1!}{(x^1 - N_1^j)!} \cdot \frac{x^2!}{(x^2 - N_2^j)!} \cdot \ldots \cdot \frac{x^n!}{(x^n - N_n^j)!}$$

$$s_j^-(\mathbf{x}) = k_j^- \prod_{i=1}^{n} \frac{x^i!}{(x^i - M_i^j)!} = k_j^+ \cdot \frac{x^1!}{(x^1 - M_1^j)!} \cdot \frac{x^2!}{(x^2 - M_2^j)!} \cdot \ldots \cdot \frac{x^n!}{(x^n - M_n^j)!}$$

The algorithm for obtaining of ODE and SDE systems

Drift vector is calculated by the following formula:

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

and it is possible to write out the ODE system:

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

The algorithm for obtaining of ODE and SDE systems

Diffusion matrix is obtained in same way:

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

where $\mathbf{r}^1(\mathbf{r}^1)^T$ is matrix. It is possible to write out Ito SDE system

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

where $\mathbf{W}$ — nth order Wiener process.

Simple one-dimensional example. Verhulst Model

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

M and **N** are vectors:

$$N_1^1 = 1,\ N_1^2 = 2,\ M_1^1 = 2,\ M_1^2 = 1,$$

and **r** are scalar values

$$r^1 = 1,\ r^2 = -1$$

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

Simple one-dimensional example. Verhulst Model

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

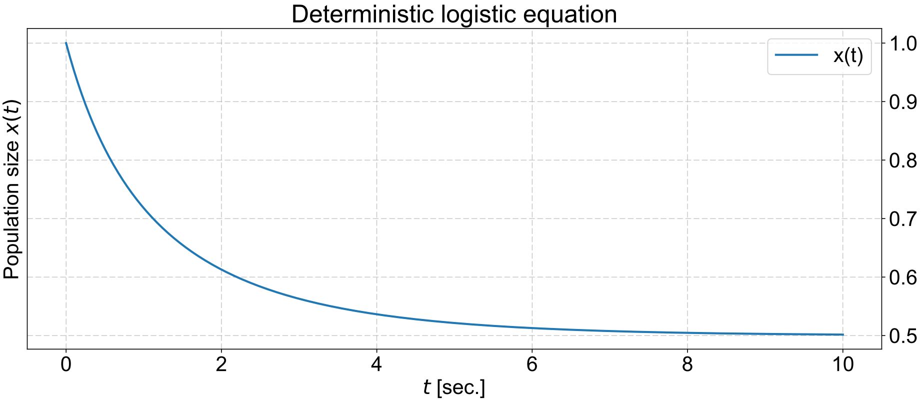
The common practice is to drop $k_2 x$ as x^2 has large value, and x is small.

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

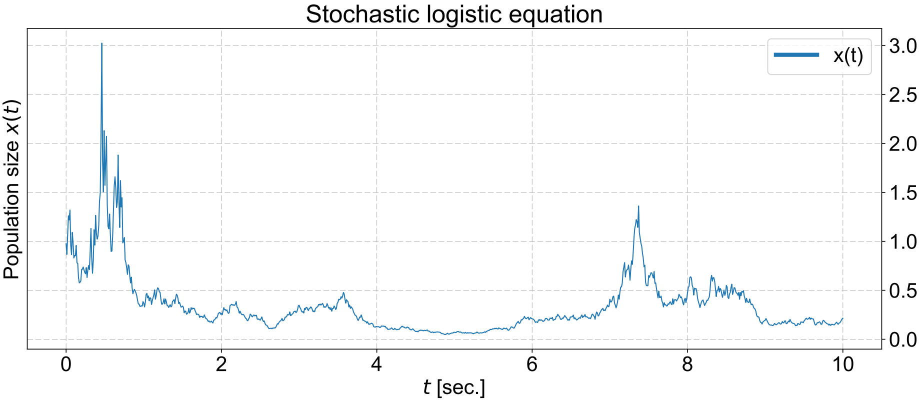
и

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

The deterministic model



The stochastic model



Automation of the algorithm

For algorithm automation we choose `SymPy` for Python language. All we need are symbolic vectors X, K and numerical matrices N and M.

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

This functions calculate drift vector and diffusion matrix.

Multidimensional example automation

We can generate random matrices and automatically derive equations:

$$\mathbf{N} = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} k_1 x_2 x_3 - 2k_2 x_1 x_2 \left(x_1 - 1\right) - k_3 x_1 \left(x_1 - 1\right) \\ k_1 x_2 x_3 \\ k_2 x_1 x_2 \left(x_1 - 1\right) \end{bmatrix}$$

$$\mathbf{G}(\mathbf{x}) = \begin{bmatrix} k_1 x_2 x_3 + 4k_2 x_1 x_2 \left(x_1 - 1\right) + k_3 x_1 \left(x_1 - 1\right) & k_1 x_2 x_3 & -2k_2 x_1 x_2 \left(x_1 - 1\right) \\ k_1 x_2 x_3 & k_1 x_2 x_3 & 0 \\ -2k_2 x_1 x_2 \left(x_1 - 1\right) & 0 & k_2 x_1 x_2 \left(x_1 - 1\right) \end{bmatrix}$$

Brusselator

A simple model of Belousov–Zhabotinsky reaction, proposed I. R. Prigogine.

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

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

Brusselator

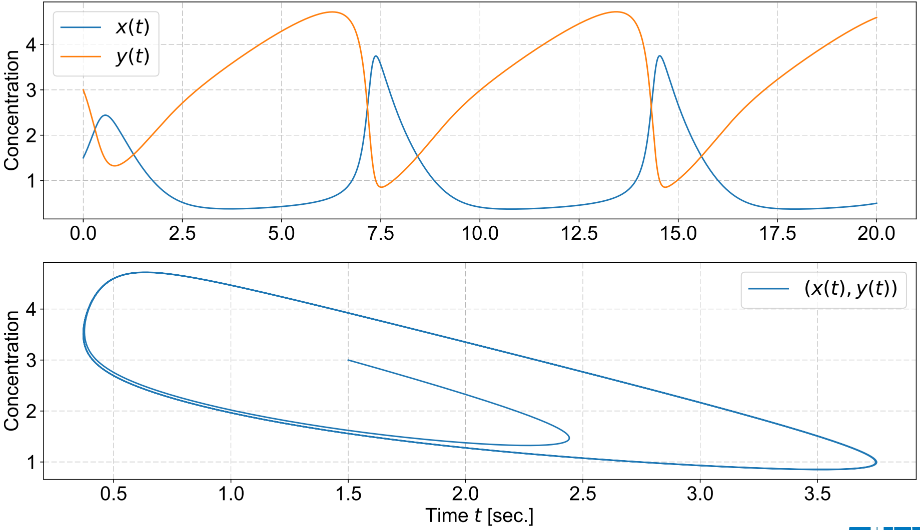
Drift vector:

$$\begin{bmatrix} ak_1 - bk_3x + k_2xy\left(x-1\right) - k_4x \\ bk_3x - k_2xy\left(x-1\right) \\ -ak_1 \\ -bk_3x \\ bk_3x \\ k_4x \end{bmatrix}$$

Matrix diffusion:

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

$(x_0, y_0) = (1.5, 3.0)$


x(t)
y(t)
Concentration
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
1
2
3
4
(x(t), y(t))
Concentration
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Time t [sec.]

Conclusion

- Using SymPy package we automated derivation of ODE and SDE systems from kinetic reactions schemes.
- The program has been tested for variety of examples.
- The generated expressions are translated to Python and Julia languages for numerical solution.

Bibliography

Kloeden P. E., Platen E.
Numerical Solution of Stochastic Differential Equations.
Berlin Heidelberg New-York: Springer, 1995.

Rößler A.
Strong and Weak Approximation Methods for Stochastic Differential Equations.
preprint, 2010.

Wiktorsson M.
Joint characteristic function and simultaneous simulation of iterated Itô integrals for multiple independent Brownian motions.
The Annals of Applied Probability. — 2001. — Vol. 11, no. 2. — P. 470–487.

Rößler A.
Runge-Kutta Methods for the Numerical Solution of Stochastic Differential Equations.
preprint, 2003.