ISSN 0361-7688, Programming and Computer Software, 2018, Vol. 44, No. 2, pp. 86–93. © Pleiades Publishing, Ltd., 2018.
Original Russian Text © M.N. Gevorkyan, A.V. Demidova, T.R. Velieva, A.V. Korol'kova, D.S. Kulyabov, L.A. Sevast'yanov, 2018, published in Programmirovanie, 2018, Vol. 44, No. 2.

Implementing a Method for Stochastization of One-Step Processes in a Computer Algebra System

M. N. Gevorkyan[a],*, A. V. Demidova[a],, T. R. Velieva[a],***, A. V. Korol'kova[a],****, D. S. Kulyabov[a,b],*****, and L. A. Sevast'yanov[a,c],********

[a] *Department of Applied Probability and Informatics, Peoples' Friendship University of Russia (RUDN), ul. Miklukho-Maklaya 6, Moscow, 117198 Russia*

[b] *Laboratory of Information Technologies, Joint Institute for Nuclear Research, ul. Zholio-Kyuri 6, Dubna, Moscow oblast, 141980 Russia*

[c] *Bogoliubov Laboratory of Theoretical Physics, Joint Institute for Nuclear Research, ul. Zholio-Kyuri 6, Dubna, Moscow oblast, 141980 Russia*

**e-mail: gevorkyan_mn@rudn.university*

***e-mail: demidova_av@rudn.university*

****e-mail: velieva_tr@rudn.university*

*****e-mail: korolkova_av@rudn.university*

******e-mail: kulyabov_ds@rudn.university*

*******e-mail: sevastianov_la@rudn.university*

Received October 11, 2017

Abstract—When modeling such phenomena as population dynamics, controllable flows, etc., a problem arises of adapting the existing models to a phenomenon under study. For this purpose, we propose to derive new models from the first principles by stochastization of one-step processes. Research can be represented as an iterative process that consists in obtaining a model and its further refinement. The number of such iterations can be extremely large. This work is aimed at software implementation (by means of computer algebra) of a method for stochastization of one-step processes. As a basis of the software implementation, we use the *SymPy* computer algebra system. Based on a developed algorithm, we derive stochastic differential equations and their interaction schemes. The operation of the program is demonstrated on the Verhulst and Lotka–Volterra models.

DOI: 10.1134/S0361768818020044

1. INTRODUCTION

Many physical and technical phenomena can be described in the framework of the statistical approach. Usually, a model is selected that is sufficiently complete to reflect a phenomenon under study, and, then, this model is refined. A question arises as to how this refinement should be carried out, because this process is quite ambiguous. Models implement certain first principles. We believe that, if a model is constructed based on the first principles, then introduced modifications will express the internal structure of the model. To describe phenomena under study (data communication networks, control systems, and population dynamics), we use the model of one-step processes [1, 2].

We developed a stochastization technique that, based on the first principles, yields a stochastic model and a deterministic model corresponding to it [3–7]. Research process is iterative: from a deterministic model, we obtain a primary model, from which we obtain a stochastic model that correlates with a deterministic model, and, based on this correlation, a refined primary model. Then, the process repeats.

Stochastization formalism can be implemented in different ways. Currently, we use a state vectors representation (combinatorial approach) and an occupation numbers representation (operator approach) [8–11].

In the case of the combinatorial approach, all operations are carried out in the state space of the system. During all model manipulations, we deal with a particular system. As a result, we obtain a description in the form of a differential equation. This approach is convenient when designing a model because it enables comparison with other models.

In the operator approach, we abstract ourselves from a particular implementation of a system under study and deal with abstract operators. The space of

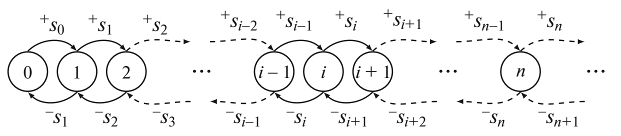

Fig. 1. One-step process.

state vectors is used only at the end of computations. Moreover, a particular operator algebra is selected based on the symmetry of a problem. This approach is convenient for theoretical constructions.

Often, it is required to find a stochastic model equivalent to a previously designed deterministic model. For this purpose, we use the combinatorial approach. In this representation, the stochastic model has the form of a differential equation, which facilitates its comparison with the original model.

This paper is organized as follows. Section 2 introduces basic notations and conventions. A method for stochastization of one-step processes, as well as its components, is described in Section 3. Section 4 describes a software complex that implements the stochastization method. In Subsection 4.1, we compare some computer algebra systems to select a basis for the software implementation. Subsection 4.2 considers the main code fragments of the stochastization program. Practical application of the method is demonstrated in Subsections 4.3 and 4.4.

2. NOTATION AND CONVENTIONS

1. For tensors, we use the abstract index notation [12], i.e., a tensor as a whole is denoted by a simple superscript (e.g., x^i), while tensor components are denoted by underlined superscripts (e.g., $x^{\underline{i}}$).

2. We adhere to the following conventions. Latin indices from the middle of the alphabet (i, j, and k) are associated with the vector state space of the system. Latin indices from the beginning of the alphabet (a) are associated with the Wiener-process space. Greek indices (α) denote the number of different interactions in kinetic equations.

3. TECHNIQUE FOR STOCHASTIZATION OF DETERMINISTIC MODELS

We developed an empirical technique for stochastization of one-step processes. It is formalized in such a way that, to use this technique, it is sufficient that the initial problem be represented as a Markov process. However, only some of its steps are explicitly algorithmized.

At the first step, the model is reduced to a one-step process (see Fig. 1). Then, this process needs to be represented in the form of interaction schemes [5, 13]. Analogs of interaction schemes are equations of chemical kinetics, particle response, etc.

For interaction schemes, we developed an algorithm (more precisely, a family of algorithms where different algorithms correspond to different approaches) that derives the master equation from interaction schemes. This equation [1, 2], however, usually has a fairly complex structure, which hinders its solution and investigation. At the next step, we obtain approximate models in the form of the Fokker–Planck and Langevin equations.

The proposed approach assumes that research is an iterative process: approximate models are refined, leading to the refinement of the original interaction schemes.

To construct models, we use the following iterative algorithm (algorithm 1).

```
Initial parameters: deterministic equation
Result: stochastic equation
until correspondence between equations
 is established do
    interaction scheme ← deterministic equation;
    stochastic equation ← interaction scheme;
    if stochastic equation
    satisfies the model, then
        terminate stochastization;
    otherwise
        reiterate
    end
end
```

Algorithm 1: Algorithm of research.

Unfortunately, the most part of this algorithm is not formalized. Derivation of interaction schemes and determination of correspondence between equations are quite difficult to formalize. And yet, we managed to formalize the derivation of stochastic equations from interaction schemes (see algorithms 2 and 3).

This process is not complex but it is rather time-consuming. Below, we describe its main stages in more detail.

3.1. Interaction Schemes

The state of a system is described by a state vector $\varphi^i \in \mathbb{R}^n$, where n is the dimension of the system. The operators $I^i_j \in \mathbb{N}^n_0 \times \mathbb{N}^n_0$ and $F^i_j \in \mathbb{N}^n_0 \times \mathbb{N}^n_0$ define the system state before and after interaction, respectively. The component indices of the system dimension take the values $\underline{i}, \underline{j} = \overline{1, n}$. As a result of interaction, the system transits from one state to another.

There are s types of different interactions in the system. Hence, instead of the operators I^i_j and F^i_j, we consider the operators $I_j^{i\alpha} \in \mathbb{N}_0^n \times \mathbb{N}_0^n \times \mathbb{N}_+^s$ and $F_j^{i\alpha} \in \mathbb{N}_0^n \times \mathbb{N}_0^n \times \mathbb{N}_+^s$. The component indices for the number of interactions take the values $\underline{\alpha} = \overline{1, s}$.

```
Initial parameters: interaction scheme (1)
Result: Langevin equations (12) and (13)
begin
    System state operators
    $I_j^{i\alpha}$, $F_j^{i\alpha}$ ← interaction scheme (1);
    State transition operator
    $r_j^{i\alpha}$ ← $I_j^{i\alpha}$, $F_j^{i\alpha}$ (2);
    Transition rates
    ${}^{+}s_{\underline{\alpha}}$, ${}^{-}s_{\underline{\alpha}}$ ← $I_j^{i\alpha}$, $F_j^{i\alpha}$, $r_j^{i\alpha}$ (7);
    master equation ← ${}^{+}s_{\underline{\alpha}}$, ${}^{-}s_{\underline{\alpha}}$, $r_j^{i\alpha}$ (6);
    Fokker–Planck equation ← master equation (8), (9);
    Langevin equation ← Fokker–Planck equation (12), (13);
end
```

Algorithm 2: Stochastization algorithm.

```
Initial parameters: interaction scheme (1)
Result: Langevin equations (12) and (13)
begin
    System state operators
    $I_j^{i\alpha}$, $F_j^{i\alpha}$ ← interaction scheme (1);
    State transition operator
    $r_j^{i\alpha}$ ← $I_j^{i\alpha}$, $F_j^{i\alpha}$ (2);
    Transition rates
    ${}^{+}_{\mathrm{fp}}s_{\underline{\alpha}}$, ${}^{-}_{\mathrm{fp}}s_{\underline{\alpha}}$ ← $I_j^{i\alpha}$, $F_j^{i\alpha}$, $r_j^{i\alpha}$ (11);
    Langevin equation ← ${}^{+}_{\mathrm{fp}}s_{\underline{\alpha}}$, ${}^{-}_{\mathrm{fp}}s_{\underline{\alpha}}$, $r_j^{i\alpha}$ (12), (13);
end
```

Algorithm 3: Simplified stochastization algorithm.

Interaction among system elements is described by interaction schemes.

$$I_j^{i\underline{\alpha}}\varphi^j \underset{{}^{-}k_{\underline{\alpha}}}{\overset{{}^{+}k_{\underline{\alpha}}}{\rightleftharpoons}} F_j^{i\underline{\alpha}}\varphi^j, \quad \underline{\alpha} = \overline{1, s}. \tag{1}$$

Here, Greek indices specify the number of interactions and Latin indices denote the dimension of the system. The coefficients ${}^{+}k_{\underline{\alpha}}$ and ${}^{-}k_{\underline{\alpha}}$ represent the intensity (rate) of interaction.

The state of the system is given by the operator

$$r_j^{i\underline{\alpha}} = F_j^{i\underline{\alpha}} - I_j^{i\underline{\alpha}}. \tag{2}$$

Thus, one interaction step $\underline{\alpha}$ in the forward and backward directions can be represented (respectively) as

$$\varphi^i \to \varphi^i + r_j^{i\underline{\alpha}}\varphi^j,$$
$$\varphi^i \to \varphi^i - r_j^{i\underline{\alpha}}\varphi^j.$$

Most often, the model is constructed in such a way that the tensors $I_j^{i\underline{\alpha}}$ and $F_j^{i\underline{\alpha}}$ are diagonal in terms of Latin indices. Hence, we can explicitly use the diagonal sections of these matrices and rewrite (1) in a more common form (as a system of linear equations) [2]:

$$I_j^{i\underline{\alpha}}\varphi^j\delta_i \underset{{}^{-}k_{\underline{\alpha}}}{\overset{{}^{+}k_{\underline{\alpha}}}{\rightleftharpoons}} F_j^{i\underline{\alpha}}\varphi^j\delta_i, \tag{3}$$

where $\delta_{\underline{i}} = (1,\dots,1)$.

We also use the following designations:

$$I^{i\underline{\alpha}} := I_j^{i\underline{\alpha}}\delta^j, \quad F^{i\underline{\alpha}} := F_j^{i\underline{\alpha}}\delta^j, \quad r^{i\underline{\alpha}} := r_j^{i\underline{\alpha}}\delta^j. \tag{4}$$

3.2. Master Equation

For one-step processes, as a kinetic equation, we consider the master equation [1, 2]

$$\frac{\partial p(\varphi_2, t_2 | \varphi_1, t_1)}{\partial t} = \int [w(\varphi_2 | \psi, t_2) p(\psi, t_2 | \varphi_1, t_1) - w(\psi | \varphi_2, t_2) p(\varphi_2, t_2 | \varphi_1, t_1)] \mathrm{d}\psi,$$

where $w(\varphi|\psi, t)$ is the probability of transition from state ψ to state φ in a unit time.

The master equation can be regarded as an implementation of the Kolmogorov equation. However, the master equation is more convenient and has a direct physical interpretation [2].

Having fixed the initial values φ_1, t_1, we can write this equation for a subensemble:

$$\frac{\partial p(\varphi, t)}{\partial t} = \int [w(\varphi | \psi, t) p(\psi, t) - w(\psi | \varphi, t) p(\varphi, t)] \mathrm{d}\psi. \tag{5}$$

For a system described by one-step processes, there are two types of transitions from one state to another that occur as a result of interaction among system elements in the forward direction $(\varphi^i + r_j^{i\underline{\alpha}}\varphi^j)$ with probability ${}^{+}s_{\underline{\alpha}}(\varphi^k)$ and in the backward direction $(\varphi^i - r_j^{i\underline{\alpha}}\varphi^j)$ with probability ${}^{-}s_{\underline{\alpha}}(\varphi^k)$ (see Fig. 1). In this case, the transition probability matrix can be written as

$$w_{\underline{\alpha}}(\varphi^i | \psi^i, t) = {}^{+}s_{\underline{\alpha}}\delta_{\varphi^i, \psi^i + 1} + {}^{-}s_{\underline{\alpha}}\delta_{\varphi^i, \psi^i - 1}, \quad \underline{\alpha} = \overline{1, s},$$

where $\delta_{i,j}$ is the Kronecker delta.

Thus, the general form of master equation (5) for the state vector φ^i, which is modified step by step with the step length $r_j^{i\underline{\alpha}}\varphi^j$, is as follows:

$$\frac{\partial p(\varphi^i,t)}{\partial t} = \sum_{\underline{\alpha}=1}^{s}\{{}^-s_{\underline{\alpha}}(\varphi^i + r^{i\underline{\alpha}},t)p(\varphi^i + r^{i\underline{\alpha}},t) + {}^+s_{\underline{\alpha}}(\varphi^i - r^{i\underline{\alpha}},t)p(\varphi^i - r^{i\underline{\alpha}},t) - [{}^+s_{\underline{\alpha}}(\varphi^i) + {}^-s_{\underline{\alpha}}(\varphi^i)]p(\varphi^i,t)\}. \tag{6}$$

The functions ${}^+s_{\underline{\alpha}}$ and ${}^-s_{\underline{\alpha}}$ for Eq. (6) are obtained using the combinatorial approach.

The unit-time transition rates ${}^+s_{\underline{\alpha}}$ and ${}^-s_{\underline{\alpha}}$ are proportional to the number of ways for selecting the number of arrangements from $\varphi^{\underline{i}}$ to $I^{\underline{i}\underline{\alpha}}$ (denoted as $A_{\varphi^{\underline{i}}}^{I^{\underline{i}\underline{\alpha}}}$) and from $\varphi^{\underline{i}}$ to $F^{\underline{i}\underline{\alpha}}$ (denoted as $A_{\varphi^{\underline{i}}}^{F^{\underline{i}\underline{\alpha}}}$), respectively; these rates are given by the expressions

$$\begin{aligned} {}^+s_{\underline{\alpha}} &= {}^+k_{\underline{\alpha}}\prod_{\underline{i}=1}^{n} A_{\varphi^{\underline{i}}}^{I^{\underline{i}\underline{\alpha}}} = {}^+k_{\underline{\alpha}}\prod_{\underline{i}=1}^{n}\frac{\varphi^{\underline{i}}!}{(\varphi^{\underline{i}} - I^{\underline{i}\underline{\alpha}})!}, \\ {}^-s_{\underline{\alpha}} &= {}^-k_{\underline{\alpha}}\prod_{\underline{i}=1}^{n} A_{\varphi^{\underline{i}}}^{F^{\underline{i}\underline{\alpha}}} = {}^-k_{\underline{\alpha}}\prod_{\underline{i}=1}^{n}\frac{\varphi^{\underline{i}}!}{(\varphi^{\underline{i}} - F^{\underline{i}\underline{\alpha}})!}. \end{aligned} \tag{7}$$

3.3. Fokker–Planck Equation

The Fokker–Planck equation is a special case of the master equation and can be regarded as its approximate form. It can be derived by expanding the master equation in a series up to the second-order terms inclusive. For this purpose, we can use the Kramers–Moyal expansion [1] (for simplicity, it is written for a one-dimensional case):

$$\frac{\partial p(\varphi,t)}{\partial t} = \sum_{n=1}^{\infty}\frac{(-1)^n}{n!}\frac{\partial^n}{\partial\varphi^n}\left[\xi^n(\varphi)p(\varphi,t)\right], \tag{8}$$

where

$$\xi^n(\varphi) = \int_{-\infty}^{\infty}(\psi - \varphi)^n w(\psi|\varphi)\mathrm{d}\psi.$$

By dropping the terms of the order higher than the second, we obtain the Fokker–Planck equation

$$\frac{\partial p(\varphi,t)}{\partial t} = -\frac{\partial}{\partial\varphi}[A(\varphi)p(\varphi,t)] + \frac{\partial^2}{\partial\varphi^2}[B(\varphi)p(\varphi,t)],$$

or, in a multidimensional case,

$$\frac{\partial p(\varphi^k,t)}{\partial t} = -\frac{\partial}{\partial\varphi^i}\left[A^i(\varphi^k)p(\varphi^k,t)\right] + \frac{1}{2}\frac{\partial^2}{\partial\varphi^i\partial\varphi^j}\left[B^{ij}(\varphi^k)p(\varphi^k,t)\right], \tag{9}$$

where

$$\begin{aligned} A^i &:= A^i(\varphi^k) = r^{i\underline{\alpha}}[{}^+_{\mathrm{fp}}s_{\underline{\alpha}} - {}^-_{\mathrm{fp}}s_{\underline{\alpha}}], \\ B^{ij} &:= B^{ij}(\varphi^k) = r^{i\underline{\alpha}}r^{j\underline{\alpha}}[{}^+_{\mathrm{fp}}s_{\underline{\alpha}} - {}^-_{\mathrm{fp}}s_{\underline{\alpha}}]. \end{aligned} \tag{10}$$

Using the Kramers–Moyal expansion, we can replace the combinations of the form $\varphi(\varphi - 1)\cdots(\varphi - (n-1))$ by $(\varphi)^n$ in (7); as a result, for the Fokker–Planck equation, we obtain

$$\begin{aligned} {}^+_{\mathrm{fp}}s_{\underline{\alpha}} &= {}^+k_{\underline{\alpha}}\prod_{\underline{i}=1}^{n}(\varphi^{\underline{i}})^{I^{\underline{i}\underline{\alpha}}}, \\ {}^-_{\mathrm{fp}}s_{\underline{\alpha}} &= {}^-k_{\underline{\alpha}}\prod_{\underline{i}=1}^{n}(\varphi^{\underline{i}})^{F^{\underline{i}\underline{\alpha}}}. \end{aligned} \tag{11}$$

It can be seen from (10) that the coefficients of the Fokker–Planck equation can be derived directly from (2) and (7); i.e., in this case, there is no need to write the master equation.

3.4. Langevin Equation

The Langevin equation

$$\mathrm{d}\varphi^i = a^i\mathrm{d}t + b^i_a\mathrm{d}W^a \tag{12}$$

corresponds to the Fokker–Planck equation; here, $a^i := a^i(\varphi^k)$, $b^i_a := b^i_a(\varphi^k)$, $\varphi^i \in \mathbb{R}^n$ is the state vector of the system, and $W^a \in \mathbb{R}^m$ is the m-dimensional Wiener process. The Wiener process is implemented as $\mathrm{d}W = \varepsilon\sqrt{\mathrm{d}t}$, where $\varepsilon \sim N(0,1)$ is the normal distribution with mean 0 and variance 1. Here, Latin indices from the middle of the alphabet denote the values associated with state vectors (space dimension is n), while Latin indices from the beginning of the alphabet denote the values associated with the vector of the Wiener process (space dimension is $m \le n$).

In this case, the coefficients of Eqs. (9) and (12) are related as follows:

$$A^i = a^i, \quad B^{ij} = b^i_a b^{ja}. \tag{13}$$

It can be seen that the second term of the Langevin equation is a square root, which has a complicated form in the multidimensional case. Note, however, that it is the squared second term of the Langevin equation that is used in many relationships, which is why there is usually no need to compute the root explicitly.

4. IMPLEMENTING THE MODEL OF ONE-STEP STOCHASTIC PROCESSES IN A COMPUTER ALGEBRA SYSTEM

4.1. Justifying the Selection of a Computer Algebra System

To implement the algorithms considered above, we had to solve the problem of selecting a computer algebra system. Our requirements quite correspond to the requirements for a universal computer algebra system; however, the spectrum of such systems is fairly wide. So, our selection criteria were as follows:

- the system must be shareware;

• the system must be interactive and support the read–eval–print loop (REPL) paradigm;

• it is desirable that the system be supported and improved by a community of developers: it makes no sense to create a product in a language that may soon turn out to be dead;

• symbolic computations constitute only one stage of the method, and the equations derived also need to be investigated, most often, by numerical methods; therefore, the system must support various output formats and, even better, provide seamless integration of a target system with other software products;

• it is also desirable that the system enable implementation of numerical methods.

There are not so many universal shareware computer algebra systems. Let us consider several main candidates.

Maxima [14, 15] is a classical system that, however, stopped in its development at the level of the late 1990s. New versions are released often but only to improve stability and correct errors. New capabilities are introduced extremely slowly. Moreover, the interaction with other programs is limited.

Axiom [16, 17] is distinguished for its mathematical approach to computer algebra. This system supports the Hindley–Milner type system [18, 19] and has a powerful internal extension language. However, because of the unresolved problems with copyright, the system is "in fever." Several versions of the system originated, with each modification having its own plan for development. It is still not clear how and when this situation will end. Moreover, the interoperability of this system is almost zero.

In our opinion, the most promising system is SymPy [20, 21]. This system was originally developed as a symbolic computation library for Python. Quite unexpectedly, Python has become a universal glue language. Its application in various projects caused an explosive growth of related tools and libraries. And SymPy was developing alongside it. Currently, SymPy is quite a powerful computer algebra system. However, most of our criteria are satisfied not by the system itself but by its libraries. And yet, SymPy seems to satisfy all criteria listed above:

• as an interactive shell, it is convenient to use the Jupyter notebook (which is a component of the iPython system [22]) that supports the REPL paradigm;

• Python is actually used as a glue language that allows one to integrate different software products, while the SciPy library [23] supports many output formats;

• SymPy's output data can be naturally transferred for numerical computations to the NumPy library [24].

Thus, to implement the method for stochastization of one-step processes, we selected the SymPy system.

4.2. Software Implementation of the Stochastization Algorithm

The algorithm for deriving a stochastic differential equation from an interaction scheme (see algorithms 2 and 3) is implemented as a sequence of operations on vector data. The initial data (interaction schemes) are represented as follows:

• a symbolic vector X represents the state vector φ;

• a symbolic vector K represents the interaction rates $^{+}k_{\underline{\alpha}}$ and $^{-}k_{\underline{\alpha}}$ (1);

• numerical matrices I and F represent the initial and final states in formula (4).

Main computations are carried out by four functions.

The first function simply finds the element $\dfrac{\varphi^{\underline{i}}!}{(\varphi^{\underline{i}} - I^{\underline{i}\alpha})!}$ from formula (7):

```
def P(x, n):
  """x is a symbol, n is an integer"""
  return sp.prod([x-i for i in range(n)])
```

The second function uses the first one to compute $^{+}s_{\underline{\alpha}}$ and $^{-}s_{\underline{\alpha}}$; this function receives symbolic vectors $X = (x^1, x^2, \dots, x^n)^T$ and $K = (^{-}k_1, \dots, ^{-}k_s)$ as its arguments and returns (as a result) a list $^{+}s_1, \dots, ^{+}s_s$ (the code is presented only for forward reactions):

```
def S(X, K, I):
  res = []
  for i in range(len(K)):
    # Compute the elements of the product
    Ps = [P(x, int(n)) for (x, n) in ↩
  zip(X, I[i,:])]
    # Find the product itself
    res.append(K[i]* sp.prod(Ps))
  # Obtain the list [s_1, ↩
    s_2, s_3, …, s_s]
  return res
```

The following are the main functions of the algorithm: `drift_vector(X, K, I, F)` yields a drift vector and `diffusion_matrix(X, K, I, F)` yields a diffusion matrix (in the `SymPy` symbolic format).

```
def drift_vector(X, K, I, F):
  """Drift vector"""
  res = sp.zeros(r=len(X), c=1)
  R = F.T - I.T
  for i in range(len(K)):
    res += R[:, i] * S(X, K, I)[i]
  return res
def diffusion_matrix(X, K, I, F):
  """Diffusion matrix"""
  res = sp.zeros(r=len(X), c=len(X))
  R = F.T - I.T
  R = sp.Matrix(R)
  for i in range(len(K)):
    res += R[:, i] * R[:, i].T * S(X, K, I)[i]
  return res
```

The output data of the program can be exported in the LATEX format by using built-in methods that allow A^i and B^{ij} to be translated into the LATEX code. For this purpose, it is sufficient to call the combination of functions `print(sympy.latex(A))`, where the variable `A` contains the result of the function `drift_vector` and the function `latex()` exports it into the LATEX code.

When using the `Jupyter` interactive shell, the correct display of the TEX notation needs to be pre-tuned. For this purpose, it is required to import the `Latex` module at the beginning of the `Jupyter` notebook:

```
from IPython.display import Latex
```

and then call the function

```
sympy.init_printing(use_unicode=True)
```

In addition, the expressions obtained can be exported in the format supported by a wide spectrum of programming languages, including universal (C/C++, Fortran) and domain-specific (Matematica, Julia) languages. These functions are available as methods of the `sympy.printing` class. For example, the following code yields a C++ function `vector<double> f(vector<double> x)`:

```
print("""vector<double>      f(vector
<double> x) {{
  vector<double> res = {{ {0} }};
  return res;
  }}""".format(", ↩
    ".join([sp.printing.ccode(f) for f
      in f])))
```

4.3. Implementation Example: Verhulst Model

For demonstration of the method, let us consider the Verhulst model [25–27]. In population dynamics, this model describes limited population growth.

The deterministic model has the form

$$\dot{\varphi} = \lambda\varphi - \beta\varphi - \gamma\varphi^2, \tag{14}$$

where λ is the reproduction rate, β is the death rate, and γ is the population decline rate.

Based on (14), we write interaction scheme (3):

$$\begin{gathered} \varphi \underset{\gamma}{\overset{\lambda}{\rightleftarrows}} 2\varphi, \\ 0 \xleftarrow{\beta} \varphi. \end{gathered} \tag{15}$$

Here, the first direct relationship means reproduction of an individual, the first backward relationship means conflict between individuals, and the second backward relationship means death of an individual.

This model is one-dimensional ($n = 1$). The number of interactions is $s = 2$.

Based on (15), we write the matrices $I^{\underline{i\alpha}}$ and $F^{\underline{i\alpha}}$:

$$I^{\underline{i\alpha}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad F^{\underline{i\alpha}} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

The following data are inputted to the program:

```
I = sp.Matrix([[1], [1]])
F = sp.Matrix([[2], [0]])
X = sp.Matrix(['phi'])
K = sp.Matrix(['k_{0}'.format(i+1) for i in ↩
    range(2)])
```

Since the problem has one dimension, the drift vector and the diffusion matrix are scalars:

$$A(\varphi) = \lambda\varphi - \beta\varphi - \gamma\varphi^2,$$
$$B(\varphi) = \lambda\varphi + \beta\varphi - \gamma\varphi^2.$$

The stochastic differential equation corresponding to Eq. (14) has the form

$$\mathrm{d}\varphi(t) = (\lambda\varphi - \beta\varphi - \gamma\varphi^2)\mathrm{d}t + \sqrt{(\lambda\varphi + \beta\varphi - \gamma\varphi^2)}\mathrm{d}W(t).$$

Thus, our goal is achieved: the model is stochastized. It should be noted that, for manual computations, even this simple one-dimensional model is rather cumbersome.

4.4. Implementation Example: Predator–Prey Model

Systems that describe interaction between two types of population, predators and preys, are well investigated, and there are many various models for these systems. The model developed (independently of each other) by A. Lotka [28, 29] and V. Volterra [30] is believed to be the first predator–prey model.

The deterministic system can be written as

$$\begin{cases} \dot{x} = k_1 x - k_2 xy, \\ \dot{y} = k_2 xy - k_3 y, \end{cases} \tag{16}$$

where x and y are interacting individuals (prey and predator, respectively) and k_1, k_2, k_3 are interaction rates.

The state vector is introduced as $\varphi^{\underline{i}} = (x, y)^T$. Based on (16), we write interaction scheme (3):

$$\begin{gathered} x \xrightarrow{k_1} 2x, \\ x + y \xrightarrow{k_2} 2y, \\ y \xrightarrow{k_3} 0. \end{gathered} \tag{17}$$

Scheme (17) has the standard interpretation. The first relationship means that the prey consumes a unit of food to immediately reproduce itself. The second relationship means that the predator eats the prey to immediately reproduce itself; this is the only possibility for prey to die. The last relationship is the only possibility for predator to die.

For this model, the system dimension is $n = 2$ and the number of interactions is $s = 3$.

Based on (17), we write the matrices $I^{\underline{i\alpha}}$ and $F^{\underline{i\alpha}}$:

$$I^{\underline{i\alpha}} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad F^{\underline{i\alpha}} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix}.$$

The following data are inputted to the program:

```
I = sp.Matrix([[1, 1, 0], [0, 1, 1]])
F = sp.Matrix([[2, 0, 0], [0, 2, 0]])
X = sp.Matrix(['x', 'y'])
K = sp.Matrix(['k_{0}'.format(i+1) for i in ↩
    range(3)])
```

As a result, the program computes the drift vector and the diffusion matrix:

$$A^{\underline{i}}(x,y) = \begin{pmatrix} k_1x - k_2xy \\ k_2xy - k_3y \end{pmatrix},$$
$$B^{\underline{ij}}(x,y) = \begin{pmatrix} k_1x + k_2xy & -k_2xy \\ -k_2xy & k_2xy + k_3y \end{pmatrix}.$$

These relationships allow us to write the Fokker–Planck equation and the stochastic differential equation corresponding to it. For this purpose, it is sufficient to extract the square root of the matrix $B^{\underline{ij}}$:

$$\begin{gathered} \mathrm{d}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} k_1x - k_2xy \\ k_2xy - k_3y \end{pmatrix}\mathrm{d}t + b^{\underline{i}}_{\underline{a}}\begin{pmatrix} \mathrm{d}W^1 \\ \mathrm{d}W^2 \end{pmatrix}, \\ b^{\underline{i}}_{\underline{a}} b^{\underline{ja}} = B^{\underline{ij}} = \begin{pmatrix} k_1x + k_2xy & -k_2xy \\ -k_2xy & k_2xy + k_3y \end{pmatrix}. \end{gathered} \tag{18}$$

It can be seen that the deterministic part of system (18) coincides with the original system (16). Since the root of a matrix in an analytical form is quite difficult to extract, it would be optimal to investigate this system numerically.

5. CONCLUSIONS

Research process often has an iterative character. Computational results are estimated based on certain criteria. Computations have to be carried out repeatedly. Computer algebra systems make it possible to automate this process.

In this paper, we have considered the simplest implementation of the algorithm for stochastization of one-step processes by using interaction schemes. For this purpose, we have analyzed several computer algebra systems based on the proposed criteria. According to these criteria, as a computer algebra system for implementation of the stochastization method, we have selected the *SymPy* system. Some essential code fragments of the implemented stochastization method have been presented. The operation of the software complex has been demonstrated on the Verhulst and Lotka–Volterra models.

For all its seeming simplicity, this software product significantly improves labor productivity of the researcher. Thus, it can be concluded that computer algebra systems, in addition to numerical computation systems, have become a necessary tool for the researcher.

ACKNOWLEDGMENTS

This work was supported in part by the Russian Foundation for Basic Research (project nos. 15-07-08795 and 16-07-00556) and the RUDN University Program 5-100.

REFERENCES

1. Gardiner, C., *Stochastic Methods: A Handbook for the Natural and Social Sciences,* Springer, 2009, 4th ed.
2. Van Kampen, N.G., *Stochastic Processes in Physics and Chemistry,* Elsevier, 2011.
3. Korolkova, A.V., Eferina, E.G., Laneev, E.B., Gudkova, I.A., Sevastianov, L.A., and Kulyabov, D.S., Stochastization of one-step processes in the occupations number representation, *Proc. 30th Eur. Conf. Modeling and Simulation,* 2016, pp. 698–704.
4. Eferina, E.G., Hnatich, M., Korolkova, A.V., Kulyabov, D.S., Sevastianov, L.A., and Velieva, T.R., Diagram representation for the stochastization of single-step processes, *Communications in Computer and Information Science,* Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V., Eds., Springer, 2016, vol. 678, pp. 483–497.
5. Hnatifič, M., Eferina, E.G., Korolkova, A.V., Kulyabov, D.S., and Sevastyanov, L.A., Operator approach to the master equation for the one-step process, *EPJ Web of Conferences,* 2015, vol. 108, pp. 58–59.
6. Gevorkyan, M.N., Demidova, A.V., Zaryadov, I.S., Sobolewski, R., Korolkova, A.V., Kulyabov, D.S., and Sevastianov, L.A., Approaches to stochastic modeling of wind turbines, *Proc. 31st Eur. Conf. Modeling and Simulation (ECMS),* Varadi, K., Vidovics-Dancs, A., Radics, J.P., Paprika, Z.Z., Zwierczyk, P.T., and Horak, P., Eds., Budapest: European Council for Modeling and Simulation, 2017, pp. 622–627.
7. Gevorkyan, M.N., Velieva, T.R., Korolkova, A.V., Kulyabov, D.S., and Sevastyanov, L.A., Stochastic Runge–Kutta software package for stochastic differential equations, Dependability Engineering and Complex Systems, Springer, 2016, vol. 470, pp. 169–179.
8. Grassberger, P. and Scheunert, M., Fock-space methods for identical classical objects, *Fortschritte der Physik,* 1980, vol. 28, no. 10, pp. 547–578.
9. Täuber, U.C., Field-theory approaches to nonequilibrium dynamics, *Ageing Glass Transition,* 2005, vol. 716, pp. 295–348.
10. Janssen, H.-K. and Täuber, U.C., The field theory approach to percolation processes, *Ann. Phys.,* 2005, vol. 315, no. 1, pp. 147–192.
11. Mobilia, M., Georgiev, I.T., and Täuber, U.C., Fluctuations and correlations in lattice models for predator–

prey interaction, *Phys. Rev. E,* 2006, vol. 73, no. 4, p. 040903.

12. Penrose, R. and Rindler, W., *Spinors and Space-Time: Volume 1, Two-Spinor Calculus and Relativistic Fields,* Cambridge Univ. Press, 1987.
13. Demidova, A.V., Korolkova, A.V., Kulyabov, D.S., and Sevastyanov, L.A., The method of constructing models of peer to peer protocols, *Proc. 6th Int. Congr. Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT),* IEEE Computer Society, 2015, pp. 557–562.
14. Bainov, D.D. and Hristova, S.G., *Differential Equations with Maxima,* Chapman and Hall/CRC, 2011.
15. Timberlake, T. K. and Mixon, J. W., Classical mechanics with maxima, *Undergraduate Lecture Notes in Physics,* New York: Springer, 2016.
16. Jenks, R.D. and Sutor, R.S., *AXIOM: The Scientific Computation System,* Springer, 1992.
17. Eferina, E.G., Korolkova, A.V., Gevorkyan, M.N., Kulyabov, D.S., and Sevastyanov, L.A., One-step stochastic processes simulation software package, *Bull. Peoples' Friendship Univ. Russia, Ser. Math., Inf. Sci., Phys.,* 2014, no. 3, pp. 46–59.
18. Hindley, R., The principal type-scheme of an object in combinatory logic, *Trans. Am. Math. Soc.,* 1969, vol. 146, p. 29.
19. Milner, R., A theory of type polymorphism in programming, *J. Comput. Syst. Sci.,* 1978, vol. 17, no. 3, pp. 348–375.
20. Lamy, R., *Instant SymPy Starter,* Packt Publishing, 2013, p. 52.
21. Eferina, E.G. and Kulyabov, D.S., Implementation of diagram technique for statistical systems in Sympy, *Proc. 6th Int. Conf. Problems of Mathematical Physics and Mathematical Modeling,* Moscow: NRNU MEPhI, 2017, pp. 125–127.
22. Perez, F. and Granger, B.E., IPython: A system for interactive scientific computing, *Comput. Sci. Eng.,* 2007, vol. **9,** no. 3, pp. 21–29.
23. Oliphant, T.E., Python for scientific computing, *Comput. Sci. Eng.,* 2007, vol. 9, no. 3, pp. 10–20.
24. Oliphant, T.E., *Guide to NumPy,* CreateSpace Independent Publishing, 2015, 2nd ed.
25. Verhulst, P.F., Notice sur la loi que la population suit dans son accroissement, 1838, vol. 10, pp. 113–117.
26. Feller, W., Die Grundlagen der Volter-raschen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung, *Acta Biotheoretica,* 1939, vol. 5, no. 1, pp. 11–40.
27. Feller, W., On the theory of stochastic processes, with particular reference to applications, *Proc. 1st Berkeley Symp. Mathematical Statistics and Probability,* 1949, pp. 403–432.
28. Lotka, A.J., Contribution to the theory of periodic reaction, *J. Phys. Chem. A,* 1910, vol. 14, no. 3, pp. 271–274.
29. Lotka, A.J., *Elements of Physical Biology,* Williams and Wilkins Company, 1925.
30. Volterra, V., Variations and fluctuations of the number of individuals in animal species living together, *Journal du Conseil permanent International pour l' Exploration de la Mer,* 1928, vol. 3, no. 1, pp. 3–51.

Translated by Yu. Kornienko