

Stochastic Runge–Kutta Software Package for Stochastic Differential Equations

M. N. Gevorkyan,^{1,*} T. R. Velieva,^{1,†} A. V. Korolkova,^{1,‡} D. S. Kulyabov,^{1,2,§} and L. A. Sevastyanov^{1,3,¶}

¹*Department of Applied Probability and Informatics
Peoples' Friendship University of Russia*

Miklukho-Maklaya str. 6, Moscow, 117198, Russia

²*Laboratory of Information Technologies
Joint Institute for Nuclear Research*

Joliot-Curie 6, Dubna, Moscow region, 141980, Russia

³*Bogoliubov Laboratory of Theoretical Physics
Joint Institute for Nuclear Research*

Joliot-Curie 6, Dubna, Moscow region, 141980, Russia

As a result of the application of a technique of multistep processes stochastic models construction the range of models, implemented as a self-consistent differential equations, was obtained. These are partial differential equations (master equation, the Fokker–Planck equation) and stochastic differential equations (Langevin equation). However, analytical methods do not always allow to research these equations adequately. It is proposed to use the combined analytical and numerical approach studying these equations. For this purpose the numerical part is realized within the framework of symbolic computation. It is recommended to apply stochastic Runge–Kutta methods for numerical study of stochastic differential equations in the form of the Langevin. Under this approach, a program complex on the basis of analytical calculations metasystem Sage is developed. For model verification logarithmic walks and Black–Scholes two-dimensional model are used. To illustrate the stochastic “predator–prey” type model is used. The utility of the combined numerical-analytical approach is demonstrated.

Keywords: Runge–Kutta methods; stochastic differential equations; algebraic biology; “predator–prey” model; computer algebra software; Sage CAS

* mngevorkyan@sci.pfu.edu.ru

† trvelieva@gmail.com

‡ avkorolkova@gmail.com

§ yamadharma@gmail.com

¶ leonid.sevast@gmail.com

I. INTRODUCTION

The mathematical models adequacy may be largely increased by taking into account stochastic properties of dynamic systems. Stochastic models are widely used in chemical kinetics, hydrodynamics, population dynamics, epidemiology, filtering of signals, economics and financial mathematics as well as different fields of physics [1]. *Stochastic differential equations* (SDE) are the main mathematical apparatus of such models.

Even for the case of ordinary differential equations (ODE) an analytical solution can be obtained only for a limited class of equations. That's why a great variety of numerical methods have been developed [2, 3] for applications. In the case of SDE the importance of numerical methods greatly increases, because the exact analytical solutions have been obtained only for a very small number of stochastic models [1, 4].

Compared with numerical methods for ODEs, numerical methods for SDEs are much less developed. There are two main reasons for this: a comparative novelty of this field of applied mathematics and much more complicated mathematical apparatus. The development of new numerical methods for stochastic case in many ways is similar to deterministic methods development [4–9]. The scheme, that has been proposed by Butcher [2], gives visual representation of three main classes of numerical schemes.

The accuracy of numerical scheme may be improved in the following way: by adding additional steps (multi-step), stages (multi-stage), and the derivatives of drift vector and diffusion matrix (multi-derivative) to the scheme.

Multi-step (Runge–Kutta like) numerical methods are more suitable for the program implementation, because they can be expressed as a sequence of explicit formulas. Thus, it is natural to spread Runge–Kutta methods in the case of stochastic differential equations.

The main goal of this paper is to give a review of stochastic Runge–Kutta methods implementation made by authors for Sage computer algebra system [10]. The Python programming language and NumPy and SciPy modules were used for this purpose because Python programming language is an open and powerful framework for scientific calculations.

A few additional algorithms have to be implemented before proceeding to realisation of stochastic Runge–Kutta algorithm. These algorithms are: generation of Wiener process trajectories, approximation of multiple stochastic integrals, n-point distributed random values generation.

The authors have faced the necessity of stochastic numerical methods implementation in the course of work on the method of stochastization of one-step processes [11–13]. As The “predator-prey”-type model was taken as an example of the methodology application in order to verify the obtained SDE models by numerical experiments. The same problem had aroused when authors developed stochastic models for RED-like queuing disciplines [14].

A. Article's content

In the first part of the article a brief introduction of SDE's mathematical apparatus is given. The definition of the scalar and multidimensional Wiener process, the definition of the scalar and multidimensional Itô SDEs, strong and weak convergence of the approximating function, matrix formula for double Itô integrals approximation are introduced.

The second part of the article is devoted to the description of stochastic numerical Runge–Kutta schemes with strong and weak orders of convergence for scalar and multi-dimensional Wiener processes.

The third part provides necessary information about library functions for implementation of numerical schemes from the second part of the paper.

In the final section the introduced in previous parts of the article methods are used to derive strong and weak approximations of “predator-prey” stochastic model. Also some conclusions about dissimilarity the stochastic version of “predator-pray” models from deterministic version are made based on numerical results.

The article can be regarded as a practical introduction to the field of stochastic numerical methods. Also it could be useful as guide for the implementation of stochastic numerical schemes in any other programming languages.

B. The choice of programming language for implementing stochastic numerical methods

The following requirements to computer algebra system's programming language were taken into account:

- advanced tools for manipulations with multidimensional arrays (up to four axes) with a large number of element are needed;
- it is critical to be able to implement parallel execution of certain functions and sections of code due to the need of a large number of independent calculations according to Monte-Carlo method;
- functions to generate large arrays of random numbers are needed;

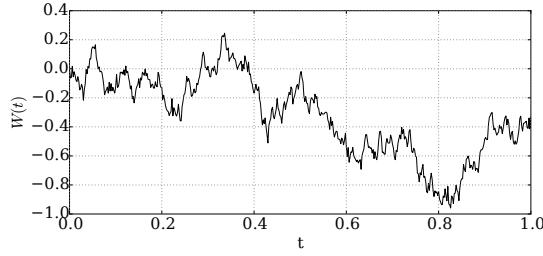


Figure 1. Wiener process path

- software should be open and free for non-commercial use.

Computer algebra system Sage generally meets all these requirements. NumPy module is used for array manipulations and SciPy module — for n-point distribution creation.

II. WIENER STOCHASTIC PROCESS

In this section only the most important definitions and notations are introduced. For brief introduction to SDE see [15, 16], for more details see [1, 17, 18].

A standard scalar Wiener process $W(t)$, $t \geq 0$ is a stochastic process which depends continuously on $t \in [t_0, T]$ and satisfies the following three conditions:

- $P\{W(0) = 0\} = 1$, in other words $W(0) = 0$ almost surely;
- $W(t)$ has independent increments i.e. $\{\Delta W_i\}_0^{N-1}$ — independently distributed random variables; $\Delta W_i = W(t_{i+1}) - W(t_i)$ and $0 \leq t_0 < t_1 < t_2 < \dots < t_N \leq T$;
- $\Delta W_i = W(t_{i+1}) - W(t_i) \sim \mathcal{N}(0, t_{i+1} - t_i)$, where $0 \leq t_{i+1} < t_i < T$, $i = 0, 1, \dots, N - 1$.

Notation $\Delta W_i \sim \mathcal{N}(0, \Delta t_i)$ denotes that ΔW_i is a normally distributed random variable with zero mean $\mathbb{E}[\Delta W_i] = \mu = 0$ and unit variance $\mathbb{D}[\Delta W_i] = \sigma^2 = \Delta t_i$.

Multidimensional Wiener process $W^\alpha(t): \Omega \times [t_0, T] \rightarrow \mathbb{R}^m$, $\forall \alpha = 1, \dots, m$, is consist of independent $W^1(t), \dots, W^m(t)$ scalar Wiener processes. The increments ΔW_i^α are mutually independent normally distributed random variables with zero mean and unit variance.

A. Computer simulation of sample Wiener process paths

In order to construct Wiener process sample path on $[0, T]$ interval divided into N parts with subinterval step $\Delta t_i = h = \text{const}$, we need to generate N normally distributed random numbers $\varepsilon_0, \dots, \varepsilon_{N-1} \sim \mathcal{N}(0, h)$ and find their cumulative sums $\varepsilon_0, \varepsilon_0 + \varepsilon_1, \varepsilon_0 + \varepsilon_1 + \varepsilon_2$ and so on. As the result two arrays (each with N elements) will be get. The first array W consist of Wiener sample path points and the second dW is an array of increments. For example of such trajectory see fig. 1.

In case of m -dimension process m sequences of N normally distributed random variables should be generated.

В описываемой нами библиотеке для генерирования выборочных винеровских траекторий написана функция:

```
(dt, t, dW, W) = wiener_process(N, dim=1, interval=(0.0, 1.0))
```

Обязательны аргумент N — число точек разбиения временного интервала `interval` (по умолчанию $[0, 1]$), а `dim` — размерность винеровского процесса. Функция возвращает кортеж из четырех элементов, где `dt` — шаг разбиения ($\Delta t_i = h = \text{const}$), `t` — одномерный `numpy` массив содержащий значения моментов времени t_1, t_2, \dots, t_N , `dW`, `W` — также `numpy` массивы размерностью $N \times m$ содержащие приращения ΔW_i^α и точки выборочной траектории W_i^α , где $i = 1, \dots, N$; $\alpha = 1, \dots, m$.

B. Itô SDE for multidimensional Wiener process

Let's consider a random process $\mathbf{x}(t) = (x^1(t), \dots, x^d(t))^T$, where $\mathbf{x}(t)$ belongs to the function space $L^2(\Omega)$ with norm $\|\cdot\|$. We assume that stochastic process $\mathbf{x}(t)$ is a solution of Itô SDE [1, 18]:

$$x^\alpha(t) = f^\alpha(t, x^\gamma(t))dt + \sum_{\beta=1}^m G_\beta^\alpha(t, x^\gamma(t))dW^\beta,$$

where $\alpha, \gamma = 1, \dots, d$, $\beta = 1, \dots, m$, vector value function $f^\alpha(t, x^\gamma(t)) = f^\alpha(t, x^1(t), \dots, x^d(t))$ is a *drift vector*, and matrix value function $G_\beta^\alpha(t, x^\gamma(t))$ is a *diffusion matrix*, $W^\alpha = (W^1, \dots, W^m)^T$ is a multidimensional Wiener process, also known as a *driver* process of SDE.

Let's introduce the discretization of interval $[t_0, T]$ by sequence $t_0 < t_1 < \dots < t_N = T$ with step $h_n = t_{n+1} - t_n$, where $n = 0, \dots, N-1$ and $h = \max\{h_{n-1}\}_1^N$ — minimal step. We also consider the step $h_n = h$ to be a constant; \mathbf{x}_n — mesh function for random process $\mathbf{x}(t)$ approximation, i.e. $\mathbf{x}_0 = \mathbf{x}(t_0)$, $\mathbf{x}_n \approx \mathbf{x}(t_n) \forall n = 1, \dots, N$.

C. Strong and weak convergences of approximation function

It is necessary to define the criterion for measuring the accuracy of process $\mathbf{x}(t)$ approximation by sequences of functions $\{\mathbf{x}_n\}_1^N$. Usually *strong* and *weak* [4, 9, 18] criteria are defined.

The sequence of approximating functions $\{\mathbf{x}_n\}_1^N$ converges with order p to exact solution $\mathbf{x}(t)$ of SDE in moment T in *strong sense* if constant $C > 0$ exists and $\delta_0 > 0$, such as $\forall h \in (0, \delta_0]$ the condition

$$\mathbb{E}(\|\mathbf{x}(T) - \mathbf{x}_N\|) \leq Ch^p$$

is fulfilled.

The sequence of approximating functions $\{\mathbf{x}_n\}_1^N$ converges with order p to solution $\mathbf{x}(t)$ of SDE in moment T in *weak sense* if constant $C_F > 0$ exists and $\delta_0 > 0$, such as $\forall h \in (0, \delta_0]$ the condition

$$|\mathbb{E}[F(\mathbf{x}(T))] - \mathbb{E}[F(\mathbf{x}_N)]| \leq C_F h^p$$

is fulfilled.

$F \in C_P^{2(p+1)}(\mathbb{R}, \mathbb{R}^d)$ is continuously differentiable (including $2(p+1)$ -th derivatives) functional with polynomial growth.

If the matrix \mathbf{G} vanishes, the condition of strong convergence is equivalent to the deterministic case convergence condition. In contrast to the deterministic case the order of strong convergence is not necessarily a natural number and may take the rational fractional values.

The convergence type selection depends entirely on the problem being solved. Numerical methods with a strong convergence are the best choice for specific trajectory approximation of a random process $\mathbf{x}(t)$ and therefore the information on the driving Wiener process is necessary. In practice, this means that the function, that implements the method with strong stochastic convergence, must get two-dimensional array with increments of ΔW_n^α , where $\alpha = 1, \dots, m$; $n = 1, \dots, N$. This array will be also used for the approximation of multiple stochastic integrals.

Weak numerical methods are suitable for approximation of characteristics of the random process $\mathbf{x}(t)$ probability distribution, because no information about the trajectory driving Wiener process is required, and random values for these methods can be generated on some other probability space, which is easy to implement in software.

D. Approximation of Itô multiple stochastic integrals

In general, for the design of numerical schemes with the order of the strong convergence greater than $p = \frac{1}{2}$, it is necessary to include single and double Itô integrals in the these schemes formula:

$$I^i(t_n, t_{n+1}) = I^i(h_n) = \int_{t_n}^{t_{n+1}} dW^i(\tau),$$

$$I^{ij}(t_n, t_{n+1}) = I^{ij}(h_n) = \int_{t_n}^{t_{n+1}} \int_{t_n}^{\tau_1} dW^i(\tau_2) dW^j(\tau_1),$$

where $i, j = 1 \dots, m$ and W^i are components of Wiener process.

So the main task is to express Itô integrals in terms of Wiener process increments $\Delta W_n^i = W^i(t_{n+1}) - W^i(t_n)$. In the case of single integral it is possible for any index i : $I^i(h_n) = \Delta W_n^i$. The case of double $I^{ij}(h_n)$ integral is more complicated and there is an exact formula only for $i = j$ case:

$$I^{ii}(h_n) = \frac{1}{2} ((\Delta W_n^i)^2 - \Delta t_n).$$

In other cases, i.e. $i \neq j$ for $I^{ij}(h_n)$, the exact expression with increments ΔW_n^α and Δt_n do not exist, so the numerical approximation shall be used.

In paper [19] author introduced a matrix form of approximation formulas. Let's denote as $\mathbf{1}_{m \times m}$ and $\mathbf{0}_{m \times m}$ the unit and zero matrices $m \times m$. Then

$$\begin{aligned} \mathbf{I}(h_n) &= \frac{\Delta \mathbf{W}_n \Delta \mathbf{W}_n - h_n \mathbf{1}_{m \times m}}{2} + \mathbf{A}(h_n), \\ \mathbf{A}(h_n) &= \frac{h}{2\pi} \sum_{k=1}^{\infty} \frac{1}{k} \left(\mathbf{V}_k (\mathbf{U}_k + \sqrt{2/h_n} \Delta \mathbf{W}_n)^T - (\mathbf{U}_k + \sqrt{2/h_n} \Delta \mathbf{W}_n) \mathbf{V}_k^T \right), \end{aligned}$$

where $\Delta \mathbf{W}_n, \mathbf{V}_k, \mathbf{U}_k$ are normally distributed independent multidimensional random values:

$$\begin{aligned} \Delta \mathbf{W}_n &= (\Delta W_n^1, \Delta W_n^2, \dots, \Delta W_n^m)^T \sim \mathcal{N}(\mathbf{0}_{m \times m}, h_n \mathbf{1}_{m \times m}), \\ \mathbf{V}_k &= (V_k^1, V_k^2, \dots, V_k^m)^T \sim \mathcal{N}(\mathbf{0}_{m \times m}, \mathbf{1}_{m \times m}), \\ \mathbf{U}_k &= (U_k^1, U_k^2, \dots, U_k^m)^T \sim \mathcal{N}(\mathbf{0}_{m \times m}, \mathbf{1}_{m \times m}). \end{aligned}$$

E. SDEs with exact solutions

For verification of written programs in the case of strong convergence SDEs with a known analytical solution will be used, so it makes possible to compute the error of trajectories approximation.

F. Logarithmic walk (one-dimensional Wiener process)

As an example of SDE with known exact solution the *logarithmic walk* model [1, 17] is used:

$$dx(t) = \mu x(t)dt + \sigma x(t)dW(t), \quad x(0) = x_0,$$

the exact solution is [1, section 4.4]:

$$x(t) = x_0 \exp((\mu - 0.5\sigma^2)t + \sigma W(t)).$$

Initial values are $\mu = 2$, $\sigma = 1$ and $x_0 = 1$.

G. Two-dimensional Black–Scholes model

For two dimensional Wiener process the two-dimensional Black–Scholes model [17] is used in following form:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t)dt + \sum_{k=1}^2 \mathbf{B}_k \mathbf{x}(t)dW^k,$$

all matrices are diagonal: $A = \text{diag}(a^1, a^2)$, $B_1 = \text{diag}(b_1^1, b_1^2)$ and $B_2 = \text{diag}(b_2^1, b_2^2)$. Drift vector $\mathbf{f}(\mathbf{x})$ and diffusion matrix $\mathbf{G}(\mathbf{x})$ can be written as

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} a^1 x^1(t) \\ a^2 x^2(t) \end{bmatrix}, \quad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} b_1^1 x^1(t) & b_1^2 x^1(t) \\ b_2^1 x^2(t) & b_2^2 x^2(t) \end{bmatrix}.$$

The exact analytical solution is given by

$$\mathbf{x}(t) = \mathbf{x}_0 \exp \left\{ \left(A - \frac{1}{2} \sum_{k=1}^2 B_k^2 \right) t + \sum_{k=1}^2 B_k W^k(t) \right\}.$$

For calculations we will take matrices: $A = \text{diag}(a_1, a_2)$, $B_1 = \text{diag}(b_1, b_2\rho)$, $B_2 = \text{diag}(0, b_2\sqrt{1-\rho})$, and following values of parameters $x_0^1 = x_0^2 = 1.0$, $a_1 = a_2 = 0.1$, $b_1 = b_2 = 0.2$, $\rho = 0.8$. Then the exact solution of the equation takes the form:

$$\begin{aligned} x^1(t) &= x_0^1 \exp((a_1 - 0.5b_1^2)t + b_1 W^1(t)) \\ x^2(t) &= x_0^2 \exp((a_2 - 0.5b_2^2)t + b_2 (\rho W^1(t) + \sqrt{1+\rho^2} W^2(t))) . \end{aligned}$$

III. STOCHASTIC RUNGE-KUTTA METHODS

In this section some of the most effective stochastic numerical Runge–Kutta methods are presented (based on the papers [4, 9, 20–24]). We restrict ourselves to the numerical schemes without derivatives of drift vector and diffusion matrix, so the effective Milstein schemes will be neglected [25–27].

In the beginning we will mention some factors that make stochastic Runge–Kutta methods more complicated in comparison to classical methods:

- When selecting a particular method we must take into account what type of convergence is necessary to provide for this specific stochastic model, and also which type of the stochastic equations is to be solved — Itô form or the Stratonovich form. This increases the number of algorithms to be programmed.
- For methods with a strong convergence of $p_s > 1$ the approximation of double stochastic integrals is required and this is a very expensive computational task.
- In the numerical scheme there are not only matrices and vectors but also tensors (four-dimensional arrays). So convolution operation on several indices is required. Implementation of convolution operation as summation by means of cycles leads to a significant drop of performance.
- In order to use the weak methods, Monte Carlo method should be applied and several series of calculations (10^7 - 10^8 calculations in each series) are required.

The program operation speed depends greatly on the necessity of multidimensional arrays convolution. NumPy module provides very useful function `einsum` (Einstein summation) which greatly improves the speed of convolution calculation.

It is also important to pay attention to the large number of zeros in generalized Butcher tables. Implementation of a separate function for specific method often helps to get a performance goal in comparison with the function that implements a universal algorithm.

A. Euler–Maruyama numerical method

The simplest numerical method for solving both scalar equations and systems of SDEs is Euler–Maruyama method, named in honor of Gishiro Maruyama (Gisiro Maruyama), who extends the classical Euler’s method from ODE to SDE case [28]. The method is easily generalized to the case of multidimensional Wiener process:

$$\begin{aligned} x_0^\alpha &= x^\alpha(t_0), \\ x_{n+1}^\alpha &= x_n^\alpha + f^\alpha(t_n, x_n^\alpha) h_n + \sum_{\gamma=1}^d G_\beta^\alpha(t_n, x_n^\gamma) \Delta W_n^\beta. \end{aligned}$$

As seen from the formulas, on each step only the increment of the driving Wiener process ΔW_n^β is required. The method has a strong $(p_d, p_s) = (1.0, 0.5)$ order. The p_d value denotes the accuracy order of the deterministic numerical method, i.e. the precision that a numerical method will give being applied to the equation with $G(t, x^\alpha(t)) \equiv 0$. The p_s value denotes the order of approximation of the stochastic part of the equation.

B. Strong stochastic Runge–Kutta methods for SDEs with scalar Wiener process

In the case of scalar SDE and Wiener process the following scheme is valid:

$$\begin{aligned} X_0^i &= x_n + \sum_{j=1}^s A_{0j}^i f(t_n + c_0^j h_n, X_0^j) h_n + \sum_{j=1}^s B_{0j}^i g(t_n + c_0^j h_n, X_1^j) \frac{I^{10}(h_n)}{\sqrt{h_n}}, \\ X_1^i &= x_n + \sum_{j=1}^s A_{1j}^i f(t_n + c_0^j h_n, X_0^j) h_n + \sum_{j=1}^s B_{1j}^i g(t_n + c_1^j h_n, X_1^j) \sqrt{h_n}, \\ x_{n+1} &= x_n + \sum_{i=1}^s a_i f(t_n + c_i^i h_n, X_0^i) h_n + \\ &\quad + \sum_{i=1}^s \left(b_i^1 I^1(h_n) + b_i^2 \frac{I^{11}(h_n)}{\sqrt{h_n}} + b_i^3 \frac{I^{10}(h_n)}{h_n} + b_i^4 \frac{I^{111}(h_n)}{h_n} \right) g(t_n + c_1^i, X_1^i). \end{aligned}$$

The method is characterized by the Butcher table [9]:

c_0^i	A_{0j}^i	B_{0j}^i	
c_1^i	A_{1j}^i	B_{1j}^i	
	a_i	b_i^1	b_i^2
		b_i^3	b_i^4

In Rößler preprint [9] there are two realisations of this scheme for $s = 4$:

0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
$\frac{3}{4}$	$\frac{3}{4} 0 0 0$	$\frac{3}{2} 0 0 0$	$\frac{3}{4} 0 0 0$	$\frac{3}{4} 0 0 0$	$\frac{3}{4} 0 0 0$	$\frac{3}{4} 0 0 0$
0	0 0 0 0	0 0 0 0	0 0 0 0	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0$	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0$	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0$
0	0 0 0 0	0 0 0 0	0 0 0 0	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0$	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0$	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0$
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
$\frac{1}{4}$	$\frac{1}{4} 0 0 0$	$\frac{1}{2} 0 0 0$	$\frac{1}{4} 0 0 0$	$\frac{1}{4} 0 0 0$	$\frac{1}{4} 0 0 0$	$\frac{1}{4} 0 0 0$
1	1 0 0 0	-1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0
$\frac{1}{4}$	0 0 $\frac{1}{4}$ 0	-5 3 $\frac{1}{2}$ 0	$\frac{1}{4}$ 0 0 $\frac{1}{4}$ 0	$\frac{1}{4}$ 0 0 $\frac{1}{4}$ 0	2 -1 $\frac{1}{2}$ 0	2 -1 $\frac{1}{2}$ 0
$\frac{1}{3}$	$\frac{2}{3}$ 0 0	-1 $\frac{4}{3}$ $\frac{2}{3}$ 0	-1 $\frac{4}{3}$ $-\frac{1}{3}$ 0	$\frac{1}{6}$ $\frac{1}{6}$ $\frac{2}{3}$ 0	-1 $\frac{4}{3}$ $\frac{2}{3}$ 0	1 $-\frac{4}{3}$ $\frac{1}{3}$ 0
		2 $-\frac{4}{3}$ $-\frac{2}{3}$ 0	-2 $\frac{5}{3}$ $-\frac{2}{3}$ 1	2 $-\frac{4}{3}$ $-\frac{2}{3}$ 0	-2 $\frac{5}{3}$ $-\frac{2}{3}$ 1	

The first scheme we will denote as **SRK1W1**, the second — as **SRK2W2**. The method **SRK1W1** has strong order $(p_d, p_s) = (2.0, 1.5)$, the method **SRK2W1** has strong order $(p_d, p_s) = (3.0, 1.5)$. Another scheme with strong order $p_s = 1.0$ can be found in [1]. This is the method with following Butcher table:

0	0 0	0 0	
0	0 0	0 0	
K1P1:	0 0	0 0	
0	0 0	0 0	
0	1 0	1 0	-1 1
		1 0	0 0

On the figure. 2 the local error of approximation of SDE exact solution for logarithmic walk model is presented. The realisation of strong stochastic Runge–Kutta method with order $p_s = 1.5$ of strong convergence was used. It is interesting to note, that the approximation of deterministic parts of the methods **SRK1W1** and **SRK2W1** don't essentially affect the error value.

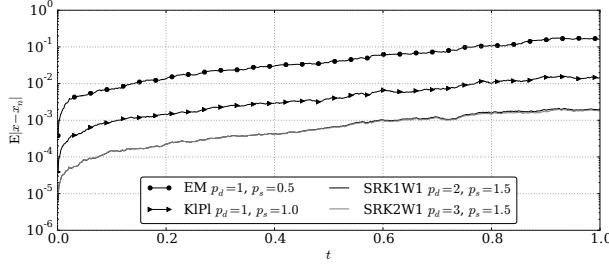


Figure 2. The errors of methods for scalar process W with $h = 10^{-3}$

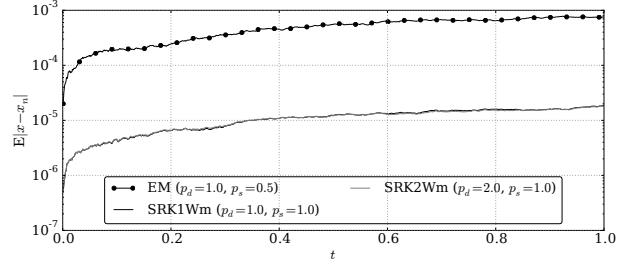


Figure 3. The errors of methods for double process W^α with $h = 10^{-3}$

C. Strong stochastic Runge–Kutta methods for SDE system with multidimensional Wiener process

For Itô SDE system with multidimensional Wiener process the stochastic Runge–Kutta scheme with strong order $p_s = 1.0$ can be obtained using single and double Itô integrals [9].

$$\begin{aligned} X^{0i\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{0j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{l=1}^m \sum_{j=1}^s B_{0j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) I^l(h_n), \\ X^{ki\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{1j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{l=1}^m \sum_{j=1}^s B_{1j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) \frac{I^{lk}(h_n)}{\sqrt{h_n}}, \\ x_{n+1}^\alpha &= x_n^\alpha + \sum_{i=1}^s a_i f^\alpha(t_n + c_0^i h_n, X^{0i\beta}) h_n + \sum_{k=1}^m \sum_{i=1}^s (b_i^1 I^k(h_n) + b_i^2 \sqrt{h_n}) G_k^\alpha(t_n + c_1^i h_n, X^{ki\beta}), \end{aligned}$$

where $n = 0, 1, \dots, N-1$; $i = 1, \dots, s$; $\beta, k = 1, \dots, m$; $\alpha = 1, \dots, d$.

Butcher table [9] has the form:

c_0^i	A_{0j}^i	B_{0j}^i	
c_1^i	A_{1j}^i	B_{1j}^i	
	a_i	b_i^1	b_i^2

There are two realisations of this scheme for $s = 3$ in Rößler preprint [9]:

SRK1Wm:	0	0	0	0	0	0		0	0	0	0	0	0
	0	0	0	0	0	0		1	1	0	0	0	0
	0	0	0	0	0	0		0	0	0	0	0	0
	0	0	0	0	0	0		0	0	0	0	0	0
	0	0	0	0	0	0		0	0	0	0	0	0
	0	0	0	0	0	0		0	0	0	0	0	0
SRK2Wm:	0	0	0	0	0	0		0	0	0	0	0	0
	0	0	0	0	1	0		1	1	0	0	1	0
	0	0	0	0	1	0		1	1	0	0	-1	0
	0	0	0	0	-1	0		1	1	0	0	1	0
	0	0	0	0	0	0		0	0	0	0	0	0
	1	0	0	1	0	0		0	0	0	0	0	0

The SRK1Wm method has the strong order $(p_d, p_s) = (1.0, 1.0)$ of convergence and the method SRK2Wm has the strong order $(p_d, p_s) = (2.0, 1.0)$ of convergence.

On figure 3 the graph of strong convergence local error for Black–Scholes model is presented. It is also important to mention, that the deterministic part of methods SRKp1W1 and SRKp2W1 does not essentially influence the error value.

D. Stochastic Runge–Kutta methodes with weak convergence of $p = 2.0$ order

Numerical methods with weak convergence are the best for approximation of characteristics of the distribution of the random process $x^\alpha(t)$. Weak numerical method does not require information about the exact trajectory of a Wiener

process W_n^α , and the random variables for these methods can be generated on another probability space. So we can use distribution which is easily generated on the computer.

For Itô SDE system with multidimensional Wiener process the following stochastic Runge-Kutta scheme with weak order $p_s = 2.0$ is valid [9].

$$\begin{aligned}
X^{0i\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{0j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{j=1}^s \sum_{l=1}^m B_{0j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) \hat{I}^l, \\
X^{ki\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{1j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{j=1}^s B_{1j}^i G_k^\alpha(t_n + c_1^j h_n, X^{kj\beta}) \sqrt{h_n}, \\
\hat{X}^{ki\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{2j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{j=1}^s \sum_{l=1, l \neq k}^m B_{2j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) \frac{\hat{I}^{kl}}{\sqrt{h_n}}, \\
x_{n+1}^\alpha &= x_n^\alpha + \sum_{i=1}^s a_i f^\alpha(t_n + c_1^i, X^{ki\beta}) h_n + \sum_{i=1}^s \sum_{k=1}^m \left(b_i^1 \hat{I}^k + b_i^2 \frac{\hat{I}^{kk}}{\sqrt{h_n}} \right) G_k^\alpha(t_n + c_1^i h_n, X^{ki\beta}) + \\
&\quad + \sum_{i=1}^s \sum_{k=1}^m \left(b_i^3 \hat{I}^k + b_i^4 \sqrt{h_n} \right) G_k^\alpha(t_n + c_2^i h_n, \hat{X}^{ki\beta}).
\end{aligned}$$

Butcher table [9] in this case appears as follows:

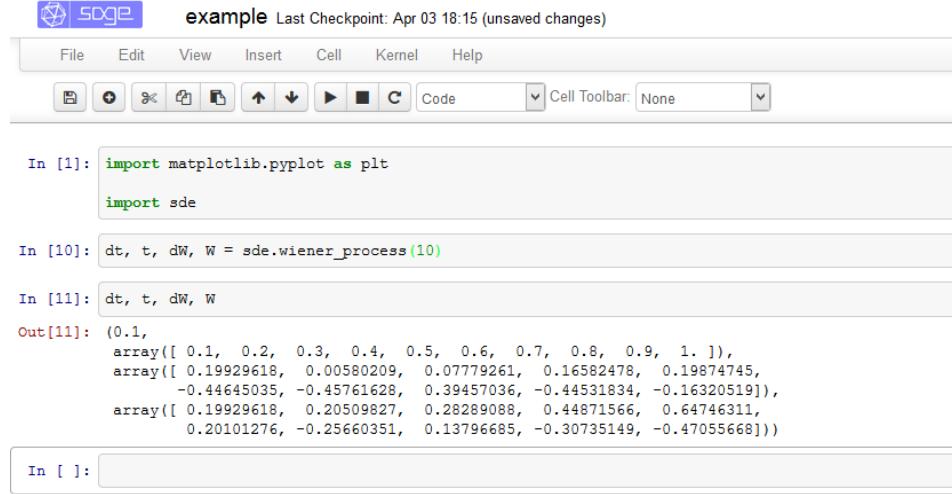
c_0^i	A_{0j}^i	B_{0j}^i
c_1^i	A_{1j}^i	B_{1j}^i
c_2^i	A_{2j}^i	B_{2j}^i
	a_i	b_i^1
		b_i^3
		b_i^4

0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	$\frac{1}{3}$	0	0	1	0	0	1	0
$\frac{5}{12}$	$\frac{25}{144}$	$\frac{35}{144}$	0	$-\frac{5}{6}$	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{2}$	0	0	1	0	0	1	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	$-\frac{1}{2}$	0	0	1	0	0	-1	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	-1	0	0	0	0	0	-1	0
$\frac{1}{10}$	$\frac{3}{14}$	$\frac{24}{35}$	1	-1	-1	0	1	-1	$\frac{1}{2}$	$\frac{1}{2}$	0
			$\frac{1}{2}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$

In the weak stochastic Runge-Kutta scheme the following random variables are used:

$$\hat{I}^{kl} = \begin{cases} \frac{1}{2}(\hat{I}^k \hat{I}^l - \sqrt{h_n} \tilde{I}^k), & k < l, \\ \frac{1}{2}(\hat{I}^k \hat{I}^l + \sqrt{h_n} \tilde{I}^l), & l < k, \\ \frac{1}{2}((\hat{I}^k)^2 - h_n). & k = l. \end{cases}$$

The random variable \hat{I}^k has three-points distribution, it can take on three fixed values: $\{-\sqrt{3h_n}, 0, \sqrt{3h_n}\}$ with probabilities $1/6$, $2/3$ and $1/6$ respectively. The variable \tilde{I}^k has two-points distribution with two values $\{-\sqrt{h_n}, \sqrt{h_n}\}$ and probabilities $1/2$ and $1/2$.



The screenshot shows the Sage notebook interface in ipython mode. The top bar includes the Sage logo, the title "example Last Checkpoint: Apr 03 18:15 (unsaved changes)", and a menu bar with File, Edit, View, Insert, Cell, Kernel, Help. Below the menu is a toolbar with various icons for file operations and cell execution. The main workspace contains the following code and its output:

```
In [1]: import matplotlib.pyplot as plt
import sde

In [10]: dt, t, dW, W = sde.wiener_process(10)

In [11]: dt, t, dW, W

Out[11]: (0.1,
array([ 0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ]),
array([ 0.19929618,  0.00580209,  0.07779261,  0.16582478,  0.19874745,
       -0.44645035, -0.45761628,  0.39457036, -0.44531834, -0.16320519]),
array([ 0.19929618,  0.20509827,  0.28289088,  0.44871566,  0.64746311,
       0.20101276, -0.25660351,  0.13796685, -0.30735149, -0.47055668]))
```

The code imports matplotlib.pyplot and sde, then generates a Wiener process trajectory with 10 points. The output consists of four arrays: dt (step size), t (time points), dW (increments), and W (trajectory points).

Figure 4. The usage of `sde` module in Sage ipython notebook mode

E. Strong and weak convergence error

To calculate the strong convergence error we have to generate a single Wiener process trajectory, and then we need to use this trajectory as driving Wiener process in order to calculate the exact solution, as well as approximate solution. After that we can find local $\|\mathbf{x}(t_n) - \mathbf{x}_n\|$, $n = 1, \dots, N$ or global $\|\mathbf{x}(T) - \mathbf{x}_N\|$ approximation errors.

Weak error calculation is not such a simple task, because we have to use Monte-Carlo method for it. For the brief reference see [4].

IV. SAGE SDE MODULE REFERENCE

Our library is a common python module. To connect it to Sage users simply perform a standard command `import sde`. Fig. 4 shows the Sage notebook in ipython mode (`-notebook=ipython` command line option)

The library contains a number of functions for internal use. The names of these functions begin with the double bottom underscore, as required by the style rules for python code PEP8.

- `(dt, t) = __time(N, interval=(0.0, 1.0))` — function divides the time interval `interval` into `N` subintervals and returns `numpy` array `t` with step `dt`;
- `(dW, W) = __scalar_wiener_process(N, dt, seed=None)` — function generates a trajectory of a scalar Wiener process `W` from `N` subintervals with step `dt`;
- `(dW, W) = __multidimensional_wiener_process(N, dim, dt, seed=None)` — similar function for generating multidimensional (`dim` dimensions) Wiener process;
- `(dW, W) = __cov_multidimensional_wiener_process(N, dim, dt, seed=None)` — another function for generating multidimensional (`dim` dimensions) Wiener process, which uses `numpy.mvtrnd`. W^1, W^2, \dots, W^m processes;
- `(dt, t, dW, W) = wiener_process(N, dim=1, interval=(0.0, 1.0), seed=None)` — the main function which should be used to generate Wiener process. Positional argument `N` denotes the number of points on the interval `interval` (default `[0, 1]`). Argument `dim` defines a dimension of Wiener process. Function returns sequence of four elements `dt`, `t`, `dW`, `W`, where `dt` is a step, ($\Delta t_i = h = \text{const}$), `t` is one-dimensional `numpy`-array of time points t_1, t_2, \dots, t_N ; `dW`, `W` are also $N \times m$ dimensional `numpy`-arrays, consisting of increments ΔW_i^α and trajectory points W_i^α , where $i = 1, \dots, N$; $\alpha = 1, \dots, m$.
- `__strong_method_selector(name)` `__weak_method_selector(name)` — set Butcher table for specific method.

The following set of functions are made for Itô integrals approximation in strong Runge-Kutta schemas. All of them get an array of increments dW as positional argument. Step size h is the second argument. All functions return a list of integral approximations for each point of driving process trajectory. For multiple integrals this list contains arrays.

- `Ito1W1(dW)` — function generates values of single Itô integral for scalar driving Wiener process;
- `Ito2W1(dW, h)` — function generates values of double Itô integral for scalar driving Wiener process;
- `Ito1Wm(dW, h)` — function generates values of single Itô integral for multidimensional driving Wiener process;
- `Ito2Wm(dW, h, n)` — function approximate values of double Itô integral. Function returns a list of arrays I^{ij} for each trajectory step. Argument n denotes the number of terms in infinite series $\mathbf{A}(h)$;
- `Ito3W1(dW, h)` — function generates values of triple Itô integral for scalar driving Wiener process.

Now we will describe core functions that implement numerical methods with strong convergence.

- `EulerMaruyama(f, g, h, x_0, dW)` `EulerMaruyamaWm(f, g, h, x_0, dW)` — EulerMaruyama method for scalar and multidimensional Wiener processes, where f — a drift vector, g — a diffusion matrix;
- `strongSRKw1(f, g, h, x_0, dW, name='SRK2W1')` — function for SDE integration with scalar Wiener process, where $f(x)$ and $g(x)$ are the same as in above functions; h — a step, x_0 — an initial value, dW — an array of Wiener process increments N ; argument `name` can take values '`SRK1W1`' and '`SRK2W2`';
- `oldstrongSRKp1Wm(f, G, h, x_0, dW, name='SRK1Wm')` — stochastic Runge–Kutta method with strong convergence order $p = 1.0$ for multidimensional Wiener process. This function is left only for performance testing for nested loops realisation of multidimensional arrays convolution ;
- `strongSRKp1Wm(f, G, h, x_0, dW, name='SRK1Wm')` — stochastic Runge–Kutta method with strong convergence order $p = 1.0$ for multidimensional Wiener process. We use NumPy method `numpy.einsum` because it gives sufficient performance goal.

The next group of functions implements methods with weak convergence.

- `n_point_distribution(values, probabilities, shape)` — n -points distribution, where `values` — an array of random variable values, `probabilities` — an array of probabilities, `shape` — arrays shape;
- `weakIto(h, dim, N)` — Itô integrals replacement for weak convergence;
- `weakSRKp2Wm(f, G, h, x_0, dW, name='SRK1Wm')` — stochastic Runge–Kutta method for multidimensional Wiener process with order $p = 2.0$.

As mentioned above, in order to apply stochastic numerical methods with weak convergence we have to use Monte-Carlo method. Thus, we must lunch a series of multiple integration of SDE using weak Runge–Kutta method. This is very expensive computational process, so the parallel computing is needed. We use standard python `multiprocessing` module for this purpose. The module has a set of functions for parallel process creation and task sharing. Since every computation is done completely independent, the processes do not require to share any data with each other.

It is important to mention that `multiprocessing` use `fork` system call to create processes. That's why all processes inherit the same number generator seed so they create exactly similar Wiener processes trajectories. To overcome this we define optional `seed` parameter for all functions, which use random number generator. It takes integer values and is used for random number generator initialization. During parallel computing a computation sequence number can be utilized as `seed|parameter`. More details and couple examples are available on repository <https://bitbucket.org/mngev/sde-numerical-integrators>.

V. STOCHASTIC “PREDATOR-PRAY” MODEL

We use our library to calculate the numerical approximation for stochastic “predators-pray” model [11–13]. “Predator-pray” model is defined by SDE with the following drift vector f and diffusion matrix G

$$f(x, y) = \begin{pmatrix} k_1ax - k_2xy \\ k_2xy - k_3y \end{pmatrix} \quad G(x, y) = \begin{pmatrix} k_1ax + k_2xy & -k_2xy \\ -k_2xy & k_2xy + k_3y \end{pmatrix}^{\frac{1}{2}},$$

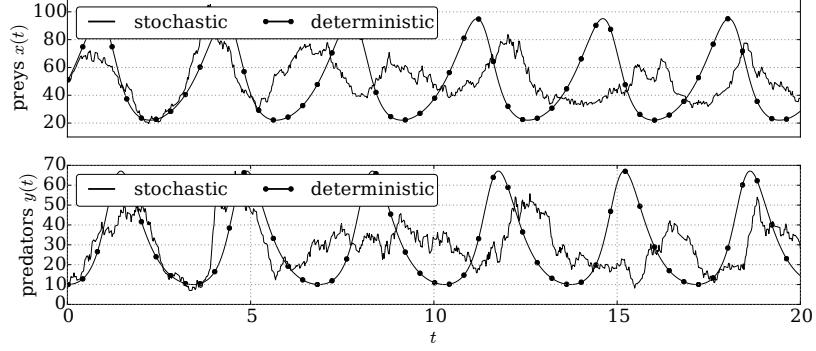


Figure 5. “Predator-pray” model numerical solutions (strong Runge-Kutta method)

where x, y are preys and predators populations respectively, a denotes preys growth, k_1 denotes preys competition, k_2 is a rate of predation, k_3 is predator’s death rate. We choose parameters values $a, k_1, k_2, k_3 = (0.5, 3.0, 0.05, 2.5)$ and initial point $(x_0, y_0) = (50.0, 30.0)$ to be closely related to the stationary point.

For stochastic “predator-pray” model the invariant $dI(x, y)$:

$$dI(x, y) = \frac{1}{2} \left(\frac{ak_1k_3}{x} + \frac{k_2k_3y}{x} + \frac{ak_1k_2x}{y} + \frac{ak_1k_3}{y} \right),$$

plays an important role. This invariant characterizes the phase volume of the system and it grows during system evolution [11]. We use strong and weak methods to check model properties.

We use weak Runge–Kutha method to calculate 500 exact realisations of SDE numerical solutions. Based on this data we find different statistical characteristics, such as median and percentiles. Time evolution of these characteristics illustrates key fiche of stochastic “predator-pray” model. Thus from fig. 8 we can see that the phase volume increases over time, also fig. 9 illustrates the same behaviour of dI value.

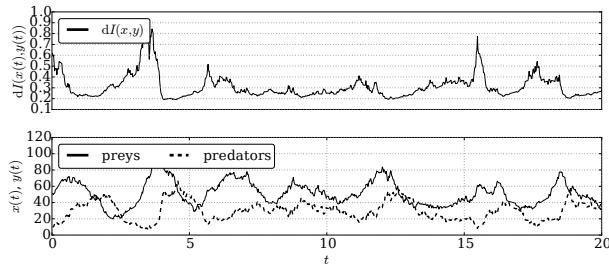


Figure 6. Invariant dI time evolution for process trajectory

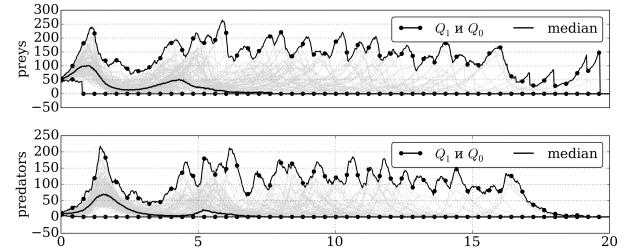


Figure 7. Multiple trajectories (gray), Q_1, Q_0 — percentiles and median.

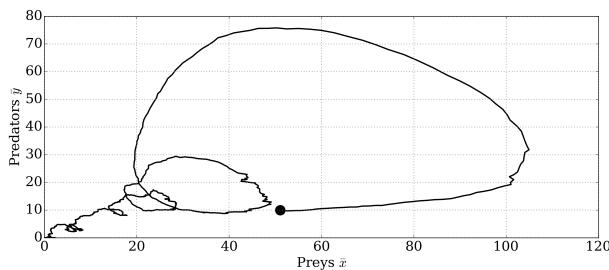


Figure 8. The phase portrait of the sample means of predators and prey quantities, based on 500 trajectories obtained by Runge–Kutta method with weak convergence.

Another key difference between deterministic and stochastic models is the tendency of imminent death of population

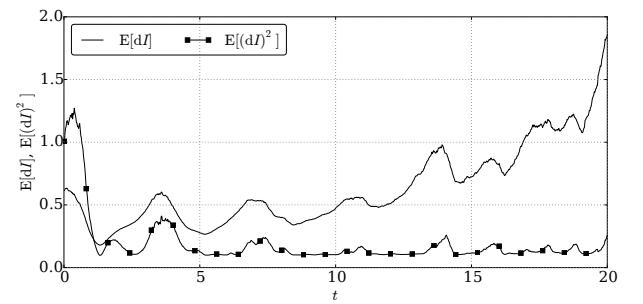


Figure 9. First and second empirical moments dI based on 500 trajectories obtained by Runge–Kutta method with weak convergence.

during evolution. On the fig. 7 grey colour denotes all trajectories realisations. It is seen that trajectories intersect the x-axis, thus the population of prays/predators dies. It makes stochastic model more realistic, because in the deterministic case the population death is not possible.

VI. CONCLUSIONS

Some realizations of stochastic Runge-Kutta methods were considered in this article. The authors gradually developed and refined the library by adding a new functionalities and optimizing existing ones. To date, the library uses numerical methods by the Rößler's article [9], as the most effective of the currently known to the authors. However, the basic functions `strongSRKw1` and `weakSRKp2Wm` are written in accordance with the general algorithm and can use any Butcher table with appropriate staging. This allows any user of the library to extend functionality by adding new methods.

Additional examples of library usage and of all files source codes are available at <https://bitbucket.org/mngev/sde-numerical-integrators>. The authors suppose to maintain the library by adding new functions.

ACKNOWLEDGMENTS

The work is partially supported by RFBR grants No's 14-01-00628, 15-07-08795, and 16-07-00556.

Calculations were carried out on computer cluster “Felix” (RUDN) and Heterogeneous computer cluster “HybriLIT” (Multifunctional center storage, processing and analysis of data JINR).

Published in: M. N. Gevorkyan, T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastyanov, Stochastic Runge–Kutta Software Package for Stochastic Differential Equations, in: Dependability Engineering and Complex Systems, Vol. 470, Springer International Publishing, 2016, pp. 169–179. doi:10.1007/978-3-319-39639-2_15.

Sources: <https://bitbucket.org/yamadharma/articles-2015-rk-stochastic>

- [1] P. E. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, 2nd Edition, Springer, Berlin Heidelberg New York, 1995.
- [2] J. C. Butcher, Numerical Methods for Ordinary Differential Equations, 2nd Edition, Wiley, New Zealand, 2003.
- [3] E. Hairer, S. P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I, 2nd Edition, Springer, Berlin, 2008.
- [4] A. Rößler, Runge-Kutta Methods for the Numerical Solution of Stochastic Differential Equations, Ph.D. thesis, Technischen Universität Darmstadt, Darmstadt (februar 2003).
- [5] K. Burrage, P. M. Burrage, High strong order explicit Runge-Kutta methods for stochastic ordinary differential equations, *Appl. Numer. Math.* (22) (1996) 81–101.
- [6] K. Burrage, P. M. Burrage, J. A. Belward, A bound on the maximum strong order of stochastic Runge-Kutta methods for stochastic ordinary differential equations., *BIT* (37) (1997) 771–780.
- [7] K. Burrage, P. M. Burrage, General order conditions for stochastic Runge-Kutta methods for both commuting and non-commuting stochastic ordinary differential equation systems, *Appl. Numer. Math.* (28) (1998) 161–177.
- [8] K. Burrage, P. M. Burrage, Order conditions of stochastic Runge-Kutta methods by B-series, *SIAM J. Numer. Anal.* (38) (2000) 1626–1646.
- [9] A. Rößler, Strong and Weak Approximation Methods for Stochastic Differential Equations – Some Recent Developments, <http://www.math.uni-hamburg.de/research/papers/prst/prst2010-02.pdf> (2010).
- [10] W. Stein, et al., Sage Mathematics Software (Version 6.5), The Sage Development Team, <http://www.sagemath.org> (2015).
- [11] A. V. Demidova, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastianov, The method of stochastization of one-step processes, in: Mathematical Modeling and Computational Physics, JINR, Dubna, 2013, p. 67.
- [12] A. V. Demidova, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastyanov, The method of constructing models of peer to peer protocols, in: 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, 2014, pp. 557–562. [arXiv:1504.00576](https://arxiv.org/abs/1504.00576), doi:10.1109/ICUMT.2014.7002162.
- [13] E. G. Eferina, A. V. Korolkova, M. N. Gevorkyan, D. S. Kulyabov, L. A. Sevastyanov, One-Step Stochastic Processes Simulation Software Package, Bulletin of Peoples' Friendship University of Russia. Series “Mathematics. Information Sciences. Physics” (3) (2014) 46–59. [arXiv:1503.07342](https://arxiv.org/abs/1503.07342).
- [14] T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, Designing installations for verification of the model of active queue management discipline RED in the GNS3, in: 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, 2014, pp. 570–577. [arXiv:1504.02324](https://arxiv.org/abs/1504.02324), doi:10.1109/ICUMT.2014.7002164.
- [15] D. J. Higham, An algorithmic introduction to numerical simulation of stochastic differential equations, *SIAM Review* 43 (3) (2001) 525–546.
- [16] S. J. A. Malham, A. Wiese, An introduction to SDE simulation (2010). [arXiv:1004.0646](https://arxiv.org/abs/1004.0646).

- [17] E. Platen, N. Bruti-Liberati, Numerical Solution of Stochastic Differential Equations with Jumps in Finance, Springer, Heidelberg Dordrecht London New York, 2010.
- [18] B. Øksendal, Stochastic differential equations. An introduction with applications, 6th Edition, Springer, Berlin Heidelberg New York, 2003.
- [19] M. Wiktorsson, Joint characteristic function and simultaneous simulation of iterated Itô integrals for multiple independent Brownian motions, *The Annals of Applied Probability* 11 (2) (2001) 470–487.
- [20] K. Debrabant, A. Rößler, Continuous weak approximation for stochastic differential equations, *Journal of Computational and Applied Mathematics* (214) (2008) 259–273.
- [21] K. Debrabant, A. Rößler, Classification of Stochastic Runge–Kutta Methods for the Weak Approximation of Stochastic Differential Equations (2013). [arXiv:1303.4510](https://arxiv.org/abs/1303.4510).
- [22] A. Tocino, R. Ardanuy, Runge–Kutta methods for numerical solution of stochastic differential equations, *Journal of Computational and Applied Mathematics* (138) (2002) 219–241.
- [23] A. R. Soheili, M. Namjoo, Strong approximation of stochastic differential equations with Runge–Kutta methods, *World Journal of Modelling and Simulation* 4 (2) (2008) 83–93.
- [24] V. Mackevičius, Second-order weak approximations for stratonovich stochastic differential equations, *Lithuanian Mathematical Journal* 34 (2) (1994) 183–200. doi:10.1007/BF02333416.
- [25] G. N. Milstein, Approximate Integration of Stochastic Differential Equations, *Theory Probab. Appl.* (19) (1974) 557–562.
- [26] G. N. Milstein, A Method of Second-Order Accuracy Integration of Stochastic Differential Equations, *Theory Probab. Appl.* (23) (1979) 396–401.
- [27] G. N. Milstein, Weak Approximation of Solutions of Systems of Stochastic Differential Equations, *Theory Probab. Appl.* (30) (1986) 750–766.
- [28] G. Maruyama, Continuous Markov processes and stochastic equations, *Rendiconti del Circolo Matematico* (4) (1955) 48–90.

Программный комплекс для решения стохастических дифференциальных уравнений стохастическим методом Рунге–Кутты

М. Н. Геворкян,^{1,*} Т. Р. Велиева,^{1,†} А. В. Королькова,^{1,‡} Д. С. Кулябов,^{1,2,§} and Л. А. Севастьянов^{1,3,¶}

¹*Кафедра прикладной информатики и теории вероятностей,
Российский университет дружбы народов,
ул. Миклухо-Маклая, д.6, Москва, Россия, 117198*

²*Лаборатория информационных технологий,
Объединённый институт ядерных исследований,
ул. Жолио-Кюри 6, Дубна, Московская область, Россия, 141980*

³*Лаборатория теоретической физики,
Объединённый институт ядерных исследований,
ул. Жолио-Кюри 6, Дубна, Московская область, Россия, 141980*

В результате применения методики построения стохастических моделей одностадийных процессов был получен спектр моделей, реализованный в виде самосогласованных дифференциальных уравнений. Это уравнения в частных производных (основное кинетическое уравнение, уравнение Фоккера–Планка) и стохастические дифференциальные уравнения (уравнение Ланжевена). Однако аналитические методы не всегда позволяют провести их адекватное исследование. Предлагается при исследовании данных уравнений использовать комбинированный аналитико-числовой подход. Для этого численная часть реализуется в рамках системы символьных вычислений. При численном исследовании стохастических дифференциальных уравнений в форме Ланжевена предлагается применять стохастические методы Рунге–Кутты. В рамках данного подхода создан программный комплекс на базе метасистемы аналитических вычислений Sage. Для верификации использованы модели логарифмического блуждания и двумерная модель Блека–Шоулза. Для иллюстрации использована стохастическая модель типа «хищник–жертва». Продемонстрировано удобство применения комбинированного численно-аналитического подхода.

Keywords: метод Рунге–Кутты; стохастические дифференциальные уравнения; модель «хищник–жертва»; символьные методы в биологии; системы компьютерной алгебры; система Sage

* mngevorkyan@sci.pfu.edu.ru

† trvelieva@gmail.com

‡ avkorolkova@gmail.com

§ yamadharma@gmail.com

¶ leonid.sevast@gmail.com

I. ВВЕДЕНИЕ

Учёт стохастических свойств при моделировании динамических систем позволяет адекватнее описывать реальные процессы. Стохастические модели применяются в химической кинетике, гидродинамике, популяционной динамике, эпидемиологии, обработке и фильтрации сигналов, экономике и финансовой математике, а также в различных разделах физики [1]. Основным математическим аппаратом для таких моделей являются стохастические дифференциальные уравнения (СДУ).

Даже в случае детерминированных дифференциальных уравнений точное аналитическое решение может быть получено лишь для ограниченного класса уравнений. На практике этот недостаток компенсируется многочисленными и хорошо разработанными численными методами [2, 3]. В случае СДУ значение численных методов повышается, так как точные аналитические решения получены для очень небольшого числа стохастических моделей [1, 4].

По сравнению с численными методами для обыкновенных дифференциальных уравнений (ОДУ), численные методы для СДУ развиты на порядок слабее. Основных причин для этого две: сравнительная новизна данной темы и значительно более сложный математический аппарат. Направления исследований и поиск новых численных методов для стохастического случая в целом совпадает с классическим случаем [4–9]. Наглядное представление об этих направлениях даёт схема, предложенная в книге Бутчера [2].

Точность аппроксимации можно повышать, используя в численной схеме значения производных от входящих в уравнение коэффициентов диффузии и сноса (*мультидифференцируемость*), вычисление каждого шага в несколько стадий (*многостадийность*), а также (*многошаговость*).

В случае детерминированных дифференциальных уравнений для программной реализации наиболее удобны многостадийные явные методы (методы типа Рунге–Кутты), так как они сводятся к последовательному вычислению ряда величин с помощью явных формул. Естественно, что одним из основных направлений исследования стало распространение теории методов Рунге–Кутты на случай СДУ.

Целью данной статьи является описание авторского программного комплекса для численного решения СДУ стохастическими методами Рунге–Кутты с попутным описанием всех используемых алгоритмов. Библиотека предназначена для CAS Sage [10] и написана на языке *python* с использованием модулей *pintpy* и *scipy*.

Для программирования с использованием стохастических численных методов необходимо было решить ряд задач по реализации вспомогательных алгоритмов, таких как генерирование траекторий винеровского процесса и многоточечных распределений, аппроксимация кратных стохастических интегралов, тестирование сильной и слабой сходимости численных методов. Только после их решения стало возможным реализовать непосредственно сами численные алгоритмы стохастического метода Рунге–Кутты.

Авторы столкнулись с необходимостью программной реализации стохастических численных методов в ходе работы над методикой согласованного введения стохастики в детерминированные модели [11–13]. В качестве примеров для применения методики были взяты модели типа хищник–жертва. Следовало верифицировать полученные результаты (стохастические модели в форме СДУ) при помощи численных экспериментов. Сходная задача возникла при работе над стохастическими моделями дисциплин обслуживания очередей типа RED на маршрутизаторах [14].

Необходимо было сравнить результаты численного решения математической модели с результатами имитационного моделирования.

A. Структура работы

В первой части статьи изложен минимум необходимых теоретических сведений: определение скалярного и многомерного процессов Винера, определение скалярного и многомерного СДУ в форме Ито, определение сильной и слабой сходимости аппроксимирующей функции, матричные формулы для аппроксимации двукратных интегралов Ито.

Вторая часть статьи посвящена описанию стохастических численных схем Рунге–Кутты с сильным и слабым порядками сходимости, как для скалярного, так и для многомерного винеровских процессов.

В третьей части дается справка по функциям библиотеки, которые реализуют вышеописанные численные схемы.

Завершает изложение пример использования реализованных алгоритмов для нахождения сильной и слабой аппроксимации решения стохастической модели хищник–жертва. На основе проведенных численных экспериментов делаются некоторые выводы о ключевых отличиях стохастической модели от детерминированной.

Статью можно рассматривать как практическое введение в область стохастических численных методов. Приводимые сведения будут полезны при реализации стохастических численных схем на других языках программирования.

B. Выбор языка программирования для реализации стохастических численных методов

При выборе системы компьютерной алгебры для реализации стохастических численных методов Рунге–Кутты авторы руководствовались следующими требованиями:

- необходимы расширенные средства для работы с многомерными массивами (до четырех размерностей) с большим количеством элементов;
- критична возможность параллельного выполнения отдельных функций и участков кода ввиду необходимости применения метода Монте-Карло с большим числом независимых испытаний;
- желательна возможность генерирования массивов случайных чисел;
- программное обеспечение должно быть открытым и бесплатным для некоммерческого использования.

В целом всем этим требованием удовлетворяет CAS Sage, написанная на языке `python`. Все необходимые функции для работы с массивами реализованы в библиотеке `numpy`, а в библиотеке `scipy` присутствуют широкий набор средств для генерирования случайных чисел. Кроме того библиотека `numpy` обеспечивает высокую производительность.

II. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ СТОХАСТИЧЕСКИХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

A. Случайный процесс Винера

Здесь излагаются лишь самые необходимые сведения. Для быстрого введения в область СДУ см. [15, 16], а для более обстоятельного знакомства см. [1, 17, 18].

Определение 1. Случайный процесс $W(t)$, $t \geq 0$ называется скалярным *процессом Винера*, если выполняются следующие условия:

- $P\{W(0) = 0\} = 1$, иначе говоря $W(0) = 0$ почти наверное;
- $W(t)$ — процесс с независимыми приращениями, то есть $\{\Delta W_i\}_0^{N-1}$ — независимые случайные величины; $\Delta W_i = W(t_{i+1}) - W(t_i)$ и $0 \leq t_0 < t_1 < t_2 < \dots < t_N \leq T$;
- $\Delta W_i = W(t_{i+1}) - W(t_i) \sim \mathcal{N}(0, t_{i+1} - t_i)$, где $0 \leq t_{i+1} < t_i < T$, $i = 0, 1, \dots, N - 1$.

Обозначение $\Delta W_i \sim \mathcal{N}(0, \Delta t_i)$ говорит о том, что ΔW_i — нормально распределенная случайная величина с математическим ожиданием $\mathbb{E}[\Delta W_i] = \mu = 0$ и дисперсией $\mathbb{D}[\Delta W_i] = \sigma^2 = \Delta t_i$.

Многомерный винеровский процесс $W^\alpha(t)$: $\Omega \times [t_0, T] \rightarrow \mathbb{R}^m$, $\forall \alpha = 1, \dots, m$, определяют как случайный процесс, составленный из совместно независимых одномерных винеровских процессов $W^1(t), \dots, W^m(t)$. Приращения ΔW_i^α являются совместно независимыми нормально распределенными случайными величинами.

B. Генерирование выборочных траекторий процесса Винера на компьютере

Для генерирования выборочной траектории одномерного винеровского процесса на отрезке $[0, T]$, разбитом на N частей с шагом $\Delta t_i = h = \text{const}$, необходимо сгенерировать N нормально распределенных случайных чисел $\varepsilon_0, \dots, \varepsilon_{N-1} \sim \mathcal{N}(0, h)$ и построить их накопленные суммы $\varepsilon_0, \varepsilon_0 + \varepsilon_1, \varepsilon_0 + \varepsilon_1 + \varepsilon_2$ и так далее. В результате получим два массива d лины N , первый W — точки выборочной траектории и второй dW — приращения этой траектории. Пример траектории приведен на рис. 1.

В случае m -мерного случайного процесса следует сгенерировать уже m последовательностей из N нормально распределенных случайных величин.

В описываемой нами библиотеке для генерирования выборочных винеровских траекторий написана функция:

```
(dt, t, dW, W) = wiener_process(N, dim=1, interval=(0.0, 1.0))
```

Обязательны аргумент N — число точек разбиения временного интервала `interval` (по умолчанию $[0, 1]$), а `dim` — размерность винеровского процесса. Функция возвращает кортеж из четырех элементов, где `dt` — шаг разбиения ($\Delta t_i = h = \text{const}$), `t` — одномерный `numpy` массив содержит значения моментов времени t_1, t_2, \dots, t_N , `dW`, `W` — также `numpy` массивы размерностью $N \times m$ содержащие приращения ΔW_i^α и точки выборочной траектории W_i^α , где $i = 1, \dots, N$; $\alpha = 1, \dots, m$.

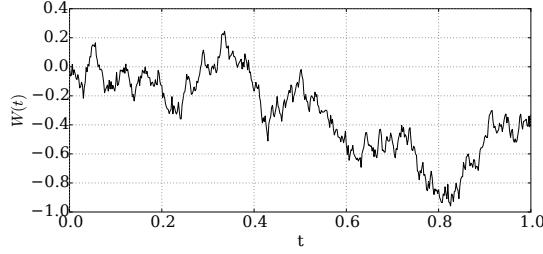


Рис. 1. Траектория винеровского процесса

C. СДУ в форме Ито для многомерного процесса Винера

Введем вероятностное пространство $(\Omega, \mathcal{A}, \mathbb{P})$, где Ω — пространство элементарных событий, \mathcal{A} — сигма-алгебра подмножеств пространства Ω , \mathbb{P} — вероятностная мера. На отрезке $[t_0, T] \in \mathbb{R}^1$ определена переменная t имеющая физический смысл времени.

Рассмотрим случайный процесс $\mathbf{x}(t) = (x^1(t), \dots, x^d(t))^T$, где $\mathbf{x}(t)$ принадлежит функциональному пространству $L^2(\Omega)$ с нормой $\|\cdot\|$. Будем считать, что случайный процесс $\mathbf{x}(t)$ является решением для СДУ в форме *Ито* [1, 18]:

$$x^\alpha(t) = f^\alpha(t, x^\gamma(t))dt + \sum_{\beta=1}^m G_\beta^\alpha(t, x^\gamma(t))dW^\beta,$$

где $\alpha, \gamma = 1, \dots, d$, $\beta = 1, \dots, m$, функция $f^\alpha(t, x^\gamma(t)) = f^\alpha(t, x^1(t), \dots, x^d(t))$ — вектор сноса, а матричнозначная функция $g_\beta^\alpha(t, x^\gamma(t))$ — матрица диффузии, $W^\alpha = (W^1, \dots, W^m)^T$ — многомерный винеровский процесс, называемый *ведущим (driver)* процессом СДУ. Кроме того $\mathbf{f}(t, \mathbf{x}(t)) = (f^1(t, \mathbf{x}), \dots, f^d(t, \mathbf{x}))^T$, а матрица \mathbf{G} имеет вид:

$$\mathbf{G} = \begin{bmatrix} g_1^1(t, \mathbf{x}) & g_2^1(t, \mathbf{x}) & \dots & g_m^1(t, \mathbf{x}) \\ g_1^2(t, \mathbf{x}) & g_2^2(t, \mathbf{x}) & \dots & g_m^2(t, \mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ g_1^d(t, \mathbf{x}) & g_2^d(t, \mathbf{x}) & \dots & g_m^d(t, \mathbf{x}) \end{bmatrix}$$

На отрезке $[t_0, T]$ введем сетку $t_0 < t_1 < \dots < t_N = T$ с шагом $h_n = t_{n+1} - t_n$, где $n = 0, \dots, N-1$ и максимальный шаг сетки $h = \max \{h_{n-1}\}_1^N$. Далее будем полагать, что если сетка равномерная, то $h_n = h = \text{const}$; \mathbf{x}_n — сеточная функция, аппроксимирующая случайный процесс $\mathbf{x}(t)$, так что $\mathbf{x}_0 = \mathbf{x}(t_0)$, $\mathbf{x}_n \approx \mathbf{x}(t_n) \forall n = 1, \dots, N$.

D. Слабая и сильная сходимости аппроксимирующей функции

Необходимо определить критерий точности аппроксимации процесса $\mathbf{x}(t)$ последовательностью функций $\{\mathbf{x}_n\}_1^N$. Таких критериев принято выделять два: *слабый* и *сильный* [4, 9, 18].

Определение 2. Последовательность аппроксимирующих функций $\{\mathbf{x}_n\}_1^N$ сходится в *сильном смысле* с порядком p к решению $\mathbf{x}(t)$ СДУ в момент времени T , если существует константа $C > 0$ и $\delta_0 > 0$, такая что $\forall h \in (0, \delta_0]$ выполняется условие

$$\mathbb{E}(\|\mathbf{x}(T) - \mathbf{x}_N\|) \leq Ch^p.$$

Определение 3. Последовательность аппроксимирующих функций $\{\mathbf{x}_n\}_1^N$ сходится в *слабом смысле* с порядком p к решению $\mathbf{x}(t)$ СДУ в момент времени T , если существует константа $C_F > 0$ и $\delta_0 > 0$, такая что $\forall h \in (0, \delta_0]$ выполняется условие

$$|\mathbb{E}[F(\mathbf{x}(T))] - \mathbb{E}[F(\mathbf{x}_N)]| \leq C_F h^p.$$

Здесь $F \in C_p^{2(p+1)}(\mathbb{R}, \mathbb{R}^d)$ — непрерывный дифференцируемый (вплоть до производных $2(p+1)$ порядка) функционал с полиномиальным ростом.

Если матрица \mathbf{G} обращается в ноль, то условие сильной сходимости равносильно условию сходимости для детерминированного случая. В отличие от детерминированного случая порядок сильной сходимости не обязательно является натуральным числом и может принимать дробно-рациональные значения.

Выбор типа сходимости целиком зависит от решаемой задачи. Численные методы с сильной сходимостью хорошо аппроксимируют конкретные траектории случайного процесса $x(t)$ и поэтому нуждаются в информации о траектории ведущего винеровского процесса. На практике это означает, что функции, реализующей стохастический метод с сильной сходимостью, необходимо передать двумерный массив со значениями приращений ΔW_n^α , где $\alpha = 1, \dots, m$; $n = 1, \dots, N$. Этот массив будет использоваться для аппроксимации кратных стохастических интегралов, которые входят в любую численную схему порядка $p > \frac{1}{2}$.

В свою очередь слабые численные методы хорошо аппроксимируют характеристики вероятностного распределения случайного процесса $x(t)$. Они не нуждаются в информации о траектории соответствующего винеровского процесса, и случайные величины для этих методов могут быть сгенерированы на другом вероятностном пространстве, которое легко реализовать программно.

Е. Вычисление и аппроксимация кратных интегралов Ито

В общем случае для конструирования численных схем, порядок сильной сходимости которых был бы больше $p = \frac{1}{2}$, необходимо включение в формулы этих схем однократных и двукратных интегралов Ито:

$$\begin{aligned} I^i(t_n, t_{n+1}) &= I^i(h_n) = \int_{t_n}^{t_{n+1}} dW^i(\tau), \\ I^{ij}(t_n, t_{n+1}) &= I^{ij}(h_n) = \int_{t_n}^{t_{n+1}} \int_{t_n}^{\tau_1} dW^i(\tau_2) dW^j(\tau_1), \end{aligned}$$

где $i, j = 1, \dots, m$ и W^i — компоненты многомерного винеровского процесса.

Возникает задача выражения этих интегралов через приращения $\Delta W_n^i = W^i(t_{n+1}) - W^i(t_n)$. В случае однократного интеграла это можно сделать для любого индекса i : $I^i(h_n) = \Delta W_n^i$. В случае же двукратного интеграла $I^{ij}(h_n)$ точная формула имеет место лишь при $i = j$:

$$I^{ii}(h_n) = \frac{1}{2} ((\Delta W_n^i)^2 - \Delta t_n).$$

В остальных же случаях при $i \neq j$ выразить $I^{ij}(h_n)$ через приращения ΔW_n^α и Δt_n в конечном виде не представляется возможным, поэтому остается лишь использовать численную аппроксимацию.

В книге [1] приведены следующие формулы для аппроксимации двукратного интеграла Ито I^{ij} :

$$\begin{aligned} I^{ij}(h_n) &= \frac{\Delta W_n^i \Delta W_n^j - h_n \delta^{ij}}{2} + A^{ij}(h_n), \\ A^{ij}(h_n) &= \frac{h}{2\pi} \sum_{k=1}^{\infty} \frac{1}{k} \left(V_k^i(U_k^j + \sqrt{2/h_n} \Delta W_n^j) - V_k^j(U_k^i + \sqrt{2/h_n} \Delta W_n^i) \right), \end{aligned}$$

$\Delta W_n^i \sim \mathcal{N}(0, h)$, $V_k^i \sim \mathcal{N}(0, 1)$, $U_k^i \sim \mathcal{N}(0, 1)$, $i = 1, \dots, m$; $k = 1, \dots, \infty$; $i = 1, \dots, N$. Из формул видно, что в случае $i = j$ получаем конечное выражение для I^{ii} , но в случае $i \neq j$ приходится суммировать бесконечный ряд A^{ij} . Данный алгоритм дает ошибку аппроксимации порядка $O(h^2/n)$, где n — число оставленных слагаемых бесконечного ряда A^{ij} .

В статье [19] предложен матричный вид аппроксимирующих формул. Пусть $\mathbf{1}_{m \times m}$, $\mathbf{0}_{m \times m}$ — единичная и нулевая матрицы $m \times m$. Тогда

$$\begin{aligned} \mathbf{I}(h_n) &= \frac{\Delta \mathbf{W}_n \Delta \mathbf{W}_n - h_n \mathbf{1}_{m \times m}}{2} + \mathbf{A}(h_n), \\ \mathbf{A}(h_n) &= \frac{h}{2\pi} \sum_{k=1}^{\infty} \frac{1}{k} \left(\mathbf{V}_k (\mathbf{U}_k + \sqrt{2/h_n} \Delta \mathbf{W}_n)^T - (\mathbf{U}_k + \sqrt{2/h_n} \Delta \mathbf{W}_n) \mathbf{V}_k^T \right), \end{aligned}$$

где $\Delta \mathbf{W}_n, \mathbf{V}_k, \mathbf{U}_k$ — независимые нормально распределенные многомерные случайные величины:

$$\begin{aligned}\Delta \mathbf{W}_n &= (\Delta W_n^1, \Delta W_n^2, \dots, \Delta W_n^m)^T \sim \mathcal{N}(\mathbf{0}_{m \times m}, h_n \mathbf{1}_{m \times m}), \\ \mathbf{V}_k &= (V_k^1, V_k^2, \dots, V_k^m)^T \sim \mathcal{N}(\mathbf{0}_{m \times m}, \mathbf{1}_{m \times m}), \quad \mathbf{U}_k = (U_k^1, U_k^2, \dots, U_k^m)^T \sim \mathcal{N}(\mathbf{0}_{m \times m}, \mathbf{1}_{m \times m}).\end{aligned}$$

В нашей библиотеке написан ряд функций для аппроксимации интегралов Ито. Следующие две функции предназначены для вычисления интегралов Ито кратности 1 и 2 для одномерного процесса Винера:

$$\text{I1} = \text{Ito1W1}(\text{dW}); \quad \text{I2} = \text{Ito2W1}(\text{dW}, \text{h})$$

они возвращают `пятерку` массивы разной размерности. Функция `Ito1W1` тривиальна и возвращает массив приращений винеровского процесса, в функция `Ito2W1` возвращает `пятерку` массив размерности (`N`, 2, 2), что соответствует интегралам $I^{ij}(h_n) \forall n = 1 \dots N, i, j = 0, 1$.

Следующие две функции предназначены для вычисления аппроксимации интегралов Ито кратности 1 и 2 для многомерного процесса Винера:

$$\text{I1m} = \text{Ito1Wm}(\text{dW}, \text{h}); \quad \text{I2m} = \text{Ito2Wm}(\text{dW}, \text{h}, \text{n}).$$

Они возвращают `пятерку` массивы разной размерности. Функция `Ito1Wm` проводит вычисления на основе точных формул и возвращает свой массив размерности (`N`, `m+1`), а функция `Ito2Wm` возвращает `пятерку` массив размерности (`N`, `m+1`, `m+1`), что соответствует интегралам $I^{ij}(h_n) \forall n = 1 \dots N, i, j = 0, \dots, m$. Аргумент `n` задает число членов в бесконечном ряде A^{ij} , который используется для аппроксимации стохастического интеграла.

F. СДУ имеющие аналитическое решение

Для верификации написанных программ в случае сильной сходимости будем использовать СДУ с известным аналитическим решением, что даст возможность вычислить погрешности аппроксимации реализаций траекторий ведущих винеровских процессов.

1. Логарифмическое блуждание (одномерный винеровский процесс)

В качестве скалярного СДУ с известным аналитическим решением возьмем уравнение логарифмического блуждания [1, 17]:

$$dx(t) = \mu x(t)dt + \sigma x(t)dW(t), \quad x(0) = x_0,$$

точное решение которого имеет вид [1, пункт 4.4]:

$$x(t) = x_0 \exp((\mu - 0.5\sigma^2)t + \sigma W(t)).$$

При вычислениях будем использовать $\mu = 2$, $\sigma = 1$ и $x_0 = 1$.

2. Двумерная модель Блека–Шоулза

Для тестирования численных методов с сильной сходимостью для систем СДУ с многомерным винеровским процессом возьмем уравнение двумерной модели Блека–Шоулза [17], которое обычно записывается в следующей форме:

$$\mathbf{x}(t) = A\mathbf{x}(t)dt + \sum_{k=1}^2 B_k \mathbf{x}(t)dW^k,$$

где все матрицы диагональны: $A = \text{diag}(a^1, a^2)$, $B_1 = \text{diag}(b_1^1, b_1^2)$ и $B_2 = \text{diag}(b_2^1, b_2^2)$. После ряда преобразования можно выразить функции $\mathbf{f}(\mathbf{x})$ и $\mathbf{G}(\mathbf{x})$ в явном виде

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} a^1 x^1(t) \\ a^2 x^2(t) \end{bmatrix}, \quad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} b_1^1 x^1(t) & b_2^1 x^1(t) \\ b_1^2 x^2(t) & b_2^2 x^2(t) \end{bmatrix}.$$

Аналитическое решение можно записать в виде экспоненты, которая берется поэлементно

$$\mathbf{x}(t) = \mathbf{x}_0 \exp \left\{ \left(A - \frac{1}{2} \sum_{k=1}^2 B_k^2 \right) t + \sum_{k=1}^2 B_k W^k(t) \right\}.$$

Для проведения вычислений возьмем следующие матрицы: $A = \text{diag}(a_1, a_2)$, $B_1 = \text{diag}(b_1, b_2\rho)$ и $B_2 = \text{diag}(0, b_2\sqrt{1-\rho})$ и следующие значения параметров $x_0^1 = x_0^2 = 1.0$, $a_1 = a_2 = 0.1$, $b_1 = b_2 = 0.2$, $\rho = 0.8$. Тогда точное решение уравнения примет вид:

$$\begin{aligned} x^1(t) &= x_0^1 \exp((a_1 - 0.5b_1^2)t + b_1 W^1(t)) \\ x^2(t) &= x_0^2 \exp((a_2 - 0.5b_2^2)t + b_2 (\rho W^1(t) + \sqrt{1+\rho^2} W^2(t))) . \end{aligned}$$

III. СТОХАСТИЧЕСКИЕ МЕТОДЫ РУНГЕ–КУТТЫ

В данном разделе мы приведем несколько наиболее эффективных стохастических численных схем Рунге–Кутты, основываясь на результатах работ [4, 9, 20–24]. Ограничимся численными схемами без производных от вектора сноса и матрицы диффузии, поэтому эффективные схемы Мильштейна останутся вне нашего внимания [25–27].

Вначале укажем ряд факторов, которые делают стохастические методы Рунге–Кутты существенно сложнее классических методов:

- При выборе конкретного метода надо учитывать, какой тип сходимости необходимо обеспечить для данной конкретной задачи, а также какое из стохастических уравнений необходимо решать — в форме Ито или в форме Стратоновича. Это увеличивает количество алгоритмов, которые нужно запрограммировать.
- Для методов с сильной сходимостью $p_s > 1$. На каждом шаге необходимо решать ресурсоемкую задачу по аппроксимации двукратных стохастических интегралов.
- В численной схеме присутствуют не только матрицы и векторы, но и тензоры (четырехмерные массивы), с которыми необходимо совершать операцию свертки по нескольким индексам. Реализация свертки через суммирование с помощью обычных циклов приводит к существенному падению производительности.
- Для использования слабых методов необходимо применять метод Монте–Карло, проводя несколько серий по 10^7 – 10^8 испытаний в каждой.

Наиболее существенно на скорости работы программы оказывается необходимость свертки многомерных массивов. Для ускорения этой операции при реализации численных схем была использована функция `einsum` из библиотеки `numpy` версии 1.6 и старше. В случае использования обычных последовательных циклов производительность вычислений существенно падает.

```
numpy.einsum(subscripts, *operands, out=None, dtype=None, order='K', casting='safe')
```

Также следует обратить внимание на большое количество нулей в обобщенных таблицах Бутчера. Реализация отдельной функции для конкретной реализации метода часто помогает получить выигрыши в производительности в несколько раз по сравнению с функцией, которая реализует универсальный алгоритм.

A. Численный метод Эйлера–Маруямы

Простейшим численным методом для решения как скалярных уравнений, так и систем СДУ, является метод Эйлера–Маруямы, названный так в честь Гиширо Маруямы (Gisiro Maruyama), который распространил классический метод Эйлера для ОДУ на случай СДУ [28]. Метод легко обобщается на случай многомерного винеровского процесса:

$$\begin{aligned} x_0^\alpha &= x^\alpha(t_0), \\ x_{n+1}^\alpha &= x_n^\alpha + f^\alpha(t_n, x_n^\alpha) h_n + \sum_{\gamma=1}^d G_\beta^\alpha(t_n, x_n^\gamma) \Delta W_n^\beta. \end{aligned}$$

Как видно из формул, на каждом шаге для вычисления следующего значения аппроксимирующей функции требуется лишь соответствующее данному шагу приращение процесса ΔW_n^β . Метод имеет сильный порядок $(p_d, p_s) = (1.0, 0.5)$. Величина p_d обозначает порядок точности детерминированной части численного метода, то есть той точности, которую будет давать численный метод при применении его к уравнению функции $G(t, x^\alpha(t)) \equiv 0$. Величина p_s обозначает порядок приближения стохастической части уравнения.

B. Стохастические методы Рунге–Кутты сильной сходимости $p = 1.5$ для одномерного винеровского процесса

В случае скалярных уравнения и винеровского процесса справедлива следующая численная схема:

$$\begin{aligned} X_0^i &= x_n + \sum_{j=1}^s A_{0j}^i f(t_n + c_0^j h_n, X_0^j) h_n + \sum_{j=1}^s B_{0j}^i g(t_n + c_1^j h_n, X_1^j) \frac{I^{10}(h_n)}{\sqrt{h_n}}, \\ X_1^i &= x_n + \sum_{j=1}^s A_{1j}^i f(t_n + c_0^j h_n, X_0^j) h_n + \sum_{j=1}^s B_{1j}^i g(t_n + c_1^j h_n, X_1^j) \sqrt{h_n}, \\ x_{n+1} &= x_n + \sum_{i=1}^s a_i f(t_n + c^i h_n, X_0^i) h_n + \\ &\quad + \sum_{i=1}^s \left(b_i^1 I^1(h_n) + b_i^2 \frac{I^{11}(h_n)}{\sqrt{h_n}} + b_i^3 \frac{I^{10}(h_n)}{h_n} + b_i^4 \frac{I^{111}(h_n)}{h_n} \right) g(t_n + c_1^i, X_1^i). \end{aligned}$$

Обобщённая таблица Бутчера который имеет вид [9]:

c_0^i	A_{0j}^i	B_{0j}^i	
c_1^i	A_{1j}^i	B_{1j}^i	
	a_i	b_i^1	b_i^2
		b_i^3	b_i^4

В препринте Росслера [9] для данной схемы приведены две таблицы Бутчера для метода четвертой стадийности $s = 4$:

0	0 0 0 0	0 0 0 0	0 0 0 0 0 0 0 0 0
$\frac{3}{4}$	$\frac{3}{4} 0 0 0$	$\frac{3}{2} 0 0 0$	$\frac{3}{4} 0 0 0 0 0 0 0 0$
0	0 0 0 0	0 0 0 0	$\frac{1}{2} \frac{1}{4} \frac{1}{4} 0 0 1 \frac{1}{2} 0 0$
0	0 0 0 0	0 0 0 0	$0 0 0 0 0 0 0 0 0$
0	0 0 0 0	0 0 0 0	0 0 0 0 0 0 0 0 0
$\frac{1}{4}$	$\frac{1}{4} 0 0 0$	$\frac{1}{2} 0 0 0$	$\frac{1}{4} \frac{1}{4} 0 0 0 -\frac{1}{2} 0 0 0$
1	1 0 0 0	-1 0 0 0	1 1 0 0 0 1 0 0 0
$\frac{1}{4}$	0 0 $\frac{1}{4}$ 0	-5 3 $\frac{1}{2}$ 0	$\frac{1}{4} 0 0 \frac{1}{4} 0 2 -1 \frac{1}{2} 0$
$\frac{1}{3} \frac{2}{3} 0 0$	-1 $\frac{4}{3} \frac{2}{3} 0$	-1 $\frac{4}{3} -\frac{1}{3} 0$	$\frac{1}{6} \frac{1}{6} \frac{2}{3} 0 -1 \frac{4}{3} \frac{2}{3} 0 1 -\frac{4}{3} \frac{1}{3} 0$
	2 $-\frac{4}{3} -\frac{2}{3} 0$	-2 $\frac{5}{3} -\frac{2}{3} 1$	2 $-\frac{4}{3} -\frac{2}{3} 0$ -2 $\frac{5}{3} -\frac{2}{3} 1$

Численную схему, реализуемую первой таблицей, обозначим как **SRK1W1**, а вторую — как **SRK2W2**. Метод **SRK1W1** имеет сильный порядок $(p_d, p_s) = (2.0, 1.5)$, а метод **SRK2W2** — сильный порядок $(p_d, p_s) = (3.0, 1.5)$. Еще один метод с сильным порядком $p_s = 1.0$ приведен в книге [1] и его таблица Бутчера имеет вид:

K1P1:	0	0 0	0 0	
	0	0 0	0 0	
	0	0 0	0 0	
	0	0 0	0 0	
	0 1 0	1 0 -1 1		
	1 0	0 0		

На рис. 2 приведены графики локальной погрешности аппроксимации точного решения стохастического уравнения модели логарифмического блуждания. Для решения использовались реализации стохастического метода Рунге–Кутты сильной сходимости $p_s = 1.5$. Отметим, что разный порядок аппроксимации детерминированной части уравнения в методах SRK1W1 и SRK2W1 несущественно влияет на общую погрешность вычисления.

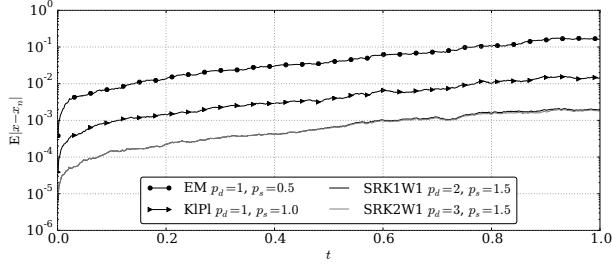


Рис. 2. Сильные погрешности методов для скалярного процесса W с $h = 10^{-3}$

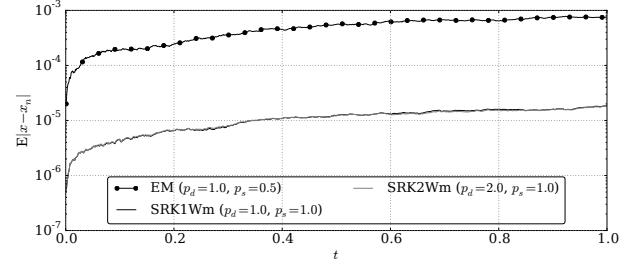


Рис. 3. Сильные погрешности методов для двумерного процесса W^α с $h = 10^{-3}$

В описываемой библиотеке написано несколько функций для решения СДУ со скалярным винеровским процессом

```
x_num = EulerMaruyama(f, g, h, x_0, dW) x_num = strongSRKW1(f, g,
h,
x_0, dW, name='SRK2W1')
```

где $f(x)$ и $g(x)$ — функции, принимающие в качестве аргумента и возвращающие действительное число; h — шаг сетки, x_0 — начальное значение, dW — массив приращений винеровского процесса N ; аргумент `name` может принимать значения '`SRK1W1`' и '`SRK2W2`'.

C. Стохастические методы Рунге–Кутты сильной сходимости $p = 1.0$ для многомерного винеровского потока

Для системы СДУ Ито с многомерным винеровским процессом можно построить стохастическую численную схему Рунге–Кутты сильного порядка $p_s = 1.0$ с использованием однократных и двукратных интегралов Ито [9].

$$\begin{aligned} X^{0i\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{0j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{l=1}^m \sum_{j=1}^s B_{0j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) I^l(h_n), \\ X^{ki\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{1j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{l=1}^m \sum_{j=1}^s B_{1j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) \frac{I^{lk}(h_n)}{\sqrt{h_n}}, \\ x_{n+1}^\alpha &= x_n^\alpha + \sum_{i=1}^s a_i f^\alpha(t_n + c_0^i h_n, X^{0i\beta}) h_n + \sum_{k=1}^m \sum_{i=1}^s (b_i^1 I^k(h_n) + b_i^2 \sqrt{h_n}) G_k^\alpha(t_n + c_1^i h_n, X^{ki\beta}), \end{aligned}$$

где $n = 0, 1, \dots, N - 1$; $i = 1, \dots, s$; $\beta, k = 1, \dots, m$; $\alpha = 1, \dots, d$.

Обобщённый таблица Бутчера имеет вид [9]:

c_0^i	A_{0j}^i	B_{0j}^i
c_1^i	A_{1j}^i	B_{1j}^i
	a_i	b_i^1

В препринте Росслера [9] указаны две таблицы Бутчера для метода третьей стадийности $s = 3$:

SRK1Wm:	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
SRK2Wm:	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	1	1	0	0	1	0
	0	0	0	0	1	0	1	1	0	0	-1	0
	0	0	0	0	-1	0	1	1	0	0	-1	0
	1	0	0	1	0	0	0	1	1	0	0	0

Метод `SRK1Wm` имеет сильный порядок сходимости $(p_d, p_s) = (1.0, 1.0)$, а метод `SRK2Wm` — сильный порядок сходимости $(p_d, p_s) = (2.0, 1.0)$.

На рис. 3 приведены графики локальной погрешности аппроксимации точного решения двумерной модели Блека–Шоулза. Для решения использовались реализации стохастического метода Рунге–Кутты сильной сходимости $p_s = 1.0$ для многомерного винеровского процесса. Отметим, что и здесь разный порядок аппроксимации детерминированной части уравнения в методах `SRK1Wm` и `SRK2Wm` несущественно сказывается на общей погрешности вычисления.

В описываемой библиотеке для решения систем СДУ с многомерным процессом Винера (предполагается, что $d = m$) написано несколько функций

```
x_num = strongSRKp1Wm(f, G, h, x_0, dW, name='SRK1Wm') x_num =
EulerMaruyamaWm(f, G, h, x_0, dW)
```

где $f(x)$ и $G(x)$ — функции, принимающие в качестве аргумента массив x и возвращающие пятеру-массивы размерности m и (m, m) ; h — шаг сетки, x_0 — начальное значение (одномерный массив размерности m), dW — массив приращений винеровского процесса (N, m) ; аргумент `name` может принимать значения '`SRK1Wm`' и '`SRK2Wm`'.

D. Стохастические методы Рунге–Кутты слабой сходимости $p = 2.0$

Численные методы со слабой сходимостью хорошо аппроксимируют характеристики распределения случайного процесса $x^\alpha(t)$. Слабый численный метод не нуждается в информации о траектории винеровского процесса W_n^α , и случайные величины для этих методов могут быть сгенерированы на другом вероятностном пространстве. Поэтому можно использовать легко моделируемое на компьютере распределение.

$$\begin{aligned} X^{0i\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{0j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{j=1}^s \sum_{l=1}^m B_{0j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) \hat{I}^l, \\ X^{ki\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{1j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{j=1}^s B_{1j}^i G_k^\alpha(t_n + c_1^j h_n, X^{kj\beta}) \sqrt{h_n}, \\ \hat{X}^{ki\alpha} &= x_n^\alpha + \sum_{j=1}^s A_{2j}^i f^\alpha(t_n + c_0^j h_n, X^{0j\beta}) h_n + \sum_{j=1}^s \sum_{l=1, l \neq k}^m B_{2j}^i G_l^\alpha(t_n + c_1^j h_n, X^{lj\beta}) \frac{\hat{I}^{kl}}{\sqrt{h_n}}, \\ x_{n+1}^\alpha &= x_n^\alpha + \sum_{i=1}^s a_i f^\alpha(t_n + c_1^i h_n, X^{ki\beta}) h_n + \sum_{i=1}^s \sum_{k=1}^m \left(b_i^1 \hat{I}^k + b_i^2 \frac{\hat{I}^{kk}}{\sqrt{h_n}} \right) G_k^\alpha(t_n + c_1^i h_n, X^{ki\beta}) + \\ &\quad + \sum_{i=1}^s \sum_{k=1}^m \left(b_i^3 \hat{I}^k + b_i^4 \sqrt{h_n} \right) G_k^\alpha(t_n + c_2^i h_n, \hat{X}^{ki\beta}). \end{aligned}$$

Обобщенная таблица Бутчера в этом случае имеет вид [9]:

c_0^i	A_{0j}^i	B_{0j}^i	
c_1^i	A_{1j}^i	B_{1j}^i	
c_2^i	A_{2j}^i	B_{2j}^i	
	a_i	b_i^1	b_i^2
		b_i^3	b_i^4

0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	$\frac{1}{3}$	0	0	1	1	0	0	0
$\frac{5}{12}$	$\frac{25}{144}$	$\frac{35}{144}$	0	$-\frac{5}{6}$	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{2}$	0	0	1	1	0	0	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	$-\frac{1}{2}$	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	-1	0	0	0	0	0	-1	0
$\frac{1}{10}$	$\frac{3}{14}$	$\frac{24}{35}$		1	-1	-1	0	1	-1	$\frac{1}{2}$	$\frac{1}{2}$
				$\frac{1}{2}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{4}$
										0	$\frac{1}{2}$
											$-\frac{1}{2}$
											0
											$-\frac{1}{2}$

В численной схеме слабого стахостического метода Рунге–Кутты используются следующие случайные величины:

$$\hat{I}^{kl} = \begin{cases} \frac{1}{2}(\hat{I}^k \hat{I}^l - \sqrt{h_n} \tilde{I}^k), & k < l, \\ \frac{1}{2}(\hat{I}^k \hat{I}^l + \sqrt{h_n} \tilde{I}^l), & l < k, \\ \frac{1}{2}((\hat{I}^k)^2 - h_n). & k = l. \end{cases}$$

Величина \hat{I}^k имеет трехточечное распределение. Это означает, что \hat{I}^k может принимать три значения $\{-\sqrt{3h_n}, 0, \sqrt{3h_n}\}$ с вероятностями $1/6, 2/3$ и $1/6$ соответственно. В свою очередь величина \tilde{I}^k имеет двухточечное распределение $\{-\sqrt{h_n}, \sqrt{h_n}\}$ с вероятностями $1/2$ и $1/2$.

В описываемой библиотеке для генерирования массива случайных чисел, распределенных по n -точечному распределению, написана функция

```
n_point_distribution(values, probabilities, shape)
```

где аргумент `values` задает список или массив из n точек x_1, \dots, x_n , аргумент `probabilities` задает массив/список вероятностей p_1, \dots, p_n , а аргумент `shape` форму требуемого массива генерируемых случайных чисел. Для генерирования величин \hat{I}^{kl} предназначена функция

```
(I_hat, I) = weakIt0(h, dim, N)
```

где h — шаг сетки, `dim` — размерность винеровского процесса и N — число точек разбиения отрезка интегрирования. Возвращаемые величины \hat{I}^k и \hat{I}^{kl} .

E. Оценка сильной погрешности

Для оценки сильной сходимости достаточно сгенерировать одну траекторию ведущего процесса Винера и для этой траектории рассчитать точное решение с помощью аналитической формулы, а также приближенное решения с помощью численных методов. После этого можно найти как локальную $\|\mathbf{x}(t_n) - \mathbf{x}_n\|$, $n = 1, \dots, N$, так и глобальную $\|\mathbf{x}(T) - \mathbf{x}_N\|$ ошибки аппроксимации.

F. Оценка слабой погрешности

Оценка слабой сходимости — значительно более трудоемкая задача, для решения которой используется метод Монте-Карло.

Проводится M испытаний, то есть к исходному СДУ M раз применяется изучаемый численный метод. В результате получаем множество независимых одинаково распределенных случайных величин \mathbf{x}_{nm} , где $m = 1, \dots, M$ — номер испытания методом Монте-Карло, а $n = 1, \dots, N$ — номер шага метода. Так как обычно интересуются глобальной погрешностью метода, то рассматривают значения аппроксимирующей функции на конце отрезка интегрирования $[t_0, T]$, где $t_N = T$, или же в любой фиксированной промежуточной точке отрезка.

Необходимо аппроксимировать функционалы вида $\mathbb{E}[f(\mathbf{x}(t))]$ с помощью выборочного среднего (эмпирического математического ожидания) от $\{\mathbf{x}_m\}_1^M$:

$$\bar{f}(\mathbf{x}_N) = \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}_m).$$

Метод Монте-Карло основывается на центральной предельной теореме, которая позволяет дать оценку абсолютного значения отклонения выборочного среднего от математического ожидания величины \mathbf{x} :

$$\frac{1}{M} \sum_{m=1}^M f(\mathbf{x}_m) - \mathbb{E}[f(\mathbf{x})] \xrightarrow[M \rightarrow \infty]{} \mathcal{N}\left(0, \frac{\mathbb{D}[f(\mathbf{x})]}{M}\right)$$

или иначе

$$\mathbb{D}[\bar{f}(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})]] = \frac{\mathbb{D}[f(\mathbf{x})]}{M}$$

$$\mathbb{D}[\bar{f}(\mathbf{x}) - \mathbb{E}[f(x)]] = \mathbb{E}[(\bar{f}(\mathbf{x}) - \mathbb{E}[f(x)] - 0)^2] = \mathbb{E}[\bar{f}(\mathbf{x}) - \mathbb{E}[f(x)]]^2.$$

Так как выполняется неравенство $\mathbb{D}[f(\mathbf{x})] = \mathbb{E}[f(\mathbf{x})]^2 - (\mathbb{E}[f(\mathbf{x})])^2 \leq \mathbb{E}[f(\mathbf{x})]^2$, то справедлива следующая оценка среднеквадратичной сходимости метода Монте-Карло:

$$\sqrt{\mathbb{E}[\bar{f}(\mathbf{x}) - \mathbb{E}[f(x)]]^2} \leq \frac{\sqrt{\mathbb{E}[f(\mathbf{x})]^2}}{\sqrt{M}}.$$

Таким образом, Метод Монте-Карло сходится как $1/\sqrt{M}$, и, кроме того, на его сходимость влияет значение второго момента функции $f(\mathbf{x})$. Если это значение велико, то число испытаний методом Монте-Карло необходимо увеличить.

Чтобы оценить глобальную слабую погрешность вычислений, необходимо учесть как погрешность метода Монте-Карло, так и погрешность самой численной схемы:

$$\text{err} = \left| \mathbb{E}[f(\mathbf{x}(T))] - \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}_{Nm}) \right| \leq |\mathbb{E}[f(\mathbf{x}(T))] - \mathbb{E}[f(\mathbf{x}_N)]| + \left| \mathbb{E}[f(\mathbf{x}_N)] - \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}_{Nm}) \right|.$$

Общая ошибка складывается из *ошибки дискретизации* численной схемы и из *статистической ошибки*. Для проверки ошибки дискретизации с помощью численных экспериментов необходимо, чтобы статистическая ошибка была меньше ошибки дискретизации. На практике это означает, что число M экспериментов методом Монте-Карло должно быть велико до такой степени, чтобы обеспечить статистическую погрешность, меньшую ошибки дискретизации.

Для построения доверительного интервала следует провести K серий экспериментов методом Монте-Карло по N испытаний в каждом. Для каждой серии следует вычислить *абсолютную среднюю ошибку*:

$$\text{MAE} = |\mu| = |\bar{f}(\mathbf{x}_N) - \mathbb{E}[f(\mathbf{x}(T))]|.$$

Тогда получим последовательность из K величин $|\mu_k|$, $k = 1, \dots, K$. Далее вычисляем эмпирическую дисперсию абсолютной средней ошибки σ_μ^2 по формуле

$$\sigma_\mu^2 = \frac{1}{K} \sum_{k=1}^K (\mu_k - \mu)^2,$$

где

$$\mu = \frac{1}{MK} \sum_{k=1}^K \sum_{m=1}^M f(\mathbf{x}_{Nm}) - \mathbb{E}[f(\mathbf{x}(T))].$$

Доверительный интервал вычисляется по следующей формуле: $[\mu - \Delta\mu, \mu + \Delta\mu]$, где

$$\Delta\mu = t_{1-\frac{\alpha}{2}, K} \sqrt{\frac{\sigma_\mu^2}{K}}$$

$t_{1-\frac{\alpha}{2}, K}$ — $(1 - \frac{1}{\alpha})$ -квантиль распределения Стьюдента с K степенью свободы.

The screenshot shows a Sage notebook window titled "example Last Checkpoint: Apr 03 18:15 (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Help. Below the menu is a toolbar with various icons. The code area contains the following:

```
In [1]: import matplotlib.pyplot as plt
import sde

In [10]: dt, t, dW, W = sde.wiener_process(10)

In [11]: dt, t, dW, W

Out[11]: (0.1,
           array([ 0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ]),
           array([ 0.19929618,  0.00580209,  0.07779261,  0.16582478,  0.19874745,
                  -0.44645035, -0.45761628,  0.39457036, -0.44531834, -0.16320519]),
           array([ 0.19929618,  0.20509827,  0.28289088,  0.44871566,  0.64746311,
                  0.20101276, -0.25660351,  0.13796685, -0.30735149, -0.47055668]))
```

The bottom cell is labeled "In []:".

Рис. 4. Использование модуля sde в Sage в режиме ipython notebook

IV. ОПИСАНИЕ БИБЛИОТЕКИ SDE

Для использования библиотеки следует ее подключить как обычный python-модуль командой `import sde`. На рис. 4 показан Sage блокнот в режиме ipython (опция `-notebook=ipython` при запуске).

В библиотеке содержится ряд функций для внутреннего использования. Названия таких функций начинаются с двойного нижнего подчеркивания, как того требуют стилевые правила для кода python PEP8.

- `(dt, t) = __time(N, interval=(0.0, 1.0))` — внутренняя функция, которая разбивает временной интервал `interval` на `N` частей и возвращает numpy массив `t` с шагом `dt`;
- `(dW, W) = __scalar_wiener_process(N, dt, seed=None)` — функция генерирует траекторию скалярного винеровского процесса `W` из `N` частей с шагом `dt`;
- `(dW, W) = __multidimensional_wiener_process(N, dim, dt, seed=None)` — аналогичная функция для генерации многомерного (размерностью `dim`) процесса Винера;
- `(dW, W) = __cov_multidimensional_wiener_process(N, dim, dt, seed=None)` — функция генерирует многомерный (размерность `dim`) винеровский процесс с использованием функции `numpy.multivariate_normal`. Процессы W^1, W^2, \dots, W^m могут коррелировать, если внутри функции задать матрицу ковариации `cov` не диагональной;
- `(dt, t, dW, W) = wiener_process(N, dim=1, interval=(0.0, 1.0), seed=None)` — основная функция, которую следует использовать для генерирования винеровского процесса. Обязательный аргумент `N` — число точек разбиения временного интервала `interval` (по умолчанию `[0, 1]`). `dim` задает размерность винеровского процесса. Функция возвращает кортеж из четырех элементов, где `dt` — шаг разбиения ($\Delta t_i = h = \text{const}$), `t` — одномерный numpy массив, содержащий значения моментов времени t_1, t_2, \dots, t_N ; `dW, W` — также numpy массивы размерностью $N \times m$, содержащие приращения ΔW_i^α и точки выборочной траектории W_i^α , где $i = 1, \dots, N$; $\alpha = 1, \dots, m$.
- `__strong_method_selector(name)` `__weak_method_selector(name)` — заполнение массивов таблицы Бутчера для конкретного метода со слабой и сильной сходимостью.

Следующий ряд функций служит для аппроксимирования интегралов Ито для методов с сильной сходимостью. Все они принимают в качестве обязательного аргумента массив приращений винеровского процесса `dW`, шаг разбиения временного интервала `h` и возвращают список значений интегралов на каждом шаге траектории. Для кратных интегралов возвращается список массивов:

- `Ito1W1(dW)` — генерирование значений однократного интеграла Ито для скалярного винеровского процесса;

- `Ito2W1(dW, h)` — генерирование значений двукратного интеграла Ито для скалярного винеровского процесса;
- `Ito1Wm(dW, h)` — генерирование значений однократного интеграла Ито для многомерного винеровского процесса;
- `Ito2Wm(dW, h, n)` — аппроксимация значений двукратных интегралов Ито. Возвращает список массивов с элементами I^{ij} для каждого шага траектории. Аргумент `n` отвечает за число слагаемых в бесконечном ряде $\mathbf{A}(h)$;
- `Ito3W1(dW, h)` — генерирование трехкратного интеграла Ито для скалярного винеровского процесса.

Теперь перейдем к основным функциям, реализующим численные методы с сильной сходимостью:

- `EulerMaruyama(f, g, h, x_0, dW)` `EulerMaruyamaWm(f, g, h, x_0, dW)` — метод Эйлера–Маруямы для скалярного и многомерного винеровских процессов, где `f` — вектор сноса, `g` — матрица диффузии;
- `strongSRKw1(f, g, h, x_0, dW, name='SRK2W1')` — функция для решения СДУ со скалярным винеровским процессом, где $f(x)$ и $g(x)$ те же, что и в предыдущем методе; `h` — шаг сетки, `x_0` — начальное значение, `dW` — массив приращений винеровского процесса `N`; аргумент `name` может принимать значения '`SRK1W1`' и '`SRK2W2`';
- `oldstrongSRKp1Wm(f, G, h, x_0, dW, name='SRK1Wm')` — стохастический метод Рунге–Кутты сильного порядка $p = 1.0$ для многомерного винеровского процесса. Функция реализует алгоритм с использованием вложенных циклов, поэтому работает крайне медленно и оставлена лишь для тестирования;
- `strongSRKp1Wm(f, G, h, x_0, dW, name='SRK1Wm')` — стохастический метод Рунге–Кутты сильного порядка $p = 1.0$ для многомерного винеровского процесса. Используется метод `numpy.einsum`, о котором говорилось выше. Благодаря этому методу вычисление свертки четырехмерного массива значительно ускоряется.

Следующая группа функций реализует методы со слабой сходимостью:

- `n_point_distribution(values, probabilities, shape)` — многоточечное распределение, в котором `values` — массив принимаемых случайной величиной значений, `probabilities` — массив вероятностей выпадения этих значений, `shape` — форма этих массивов;
- `weakIto(h, dim, N)` — замена интегралов Ито в случае слабой сходимости;
- `weakSRKp2Wm(f, G, h, x_0, dW, name='SRK1Wm')` — стохастический метод Рунге–Кутты слабого порядка $p = 2.0$ для многомерного винеровского процесса, где массив `dW` используется только для вычисления размерностей и количества точек в траектории, а сам алгоритм метода для вычислений его не использует.

Как уже упоминалось выше, для использования стохастических численных методов со слабой сходимостью следует применять метод Монте–Карло, который заключается в многократном применении одного и того же метода к СДУ. Это очень ресурсоемкий процесс, который, однако, легко распараллеливается как на много-процессорном компьютере с общей памятью, так и на кластере с несколькими узлами. Для этой цели можно использовать модуль `multiprocessing` из стандартной библиотеки языка python. Этот модуль позволяет автоматически создать необходимое число процессов и распределить работу между ними. Так как каждое вычисление производится полностью независимо, то процессам не нужно обмениваться между собой никакими данными.

Важно заметить, что `multiprocessing` для создания процессов в рамках операционных систем Unix использует системный вызов `fork`, из-за чего все порожденные процессы наследуют один и тот же генератор случайных чисел, в результате чего генерируют совершенно одинаковые винеровские траектории. Во избежание этого во всех функциях библиотеки, которые так или иначе связаны с генерацией случайных чисел, предусмотрен необязательный аргумент `seed`, который может принимать целочисленные значения и использоваться для инициализации генератора. При параллельных вычислениях в качестве аргумента `seed` можно передать порядковый номер вычисления. Более подробное описание доступно по ссылке <https://bitbucket.org/mngev/sde-numerical-integrators>.

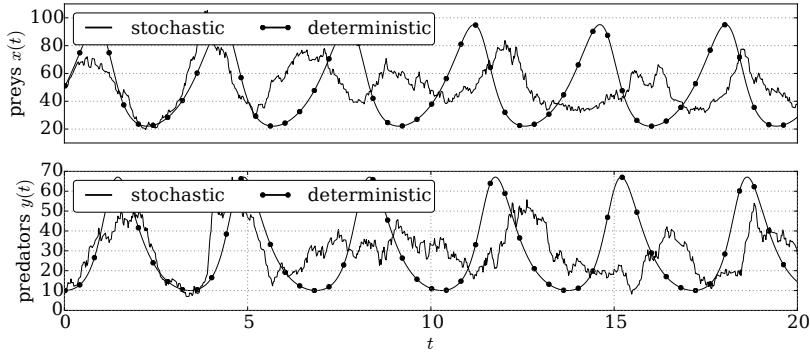


Рис. 5. Графики решений уравнений детерминированной модели хищник-жертва и соответствующей стохастической модели с помощью стохастического метода Рунге–Кутты с сильной сходимостью

V. СТОХАСТИЧЕСКАЯ МОДЕЛЬ ХИЩНИК-ЖЕРТВА

Описанный выше комплекс программ был применен для численных экспериментов со стахостичекой моделью хищник-жертва [11–13].

Модель задается СДУ с вектором сноса f и матрицей диффузии G следующего вида:

$$f(x, y) = \begin{pmatrix} k_1ax - k_2xy \\ k_2xy - k_3y \end{pmatrix} \quad G(x, y) = \begin{pmatrix} k_1ax + k_2xy & -k_2xy \\ -k_2xy & k_2xy + k_3y \end{pmatrix}^{\frac{1}{2}},$$

где x — численность жертв, y — численность хищников, a — рост жертв, k_1 — конкуренция жертв, k_2 — частота хищничества, k_3 — смертность хищников. Выберем параметры $a, k_1, k_2, k_3 = (0.5, 3.0, 0.05, 2.5)$ и начальную точку, близкую к стационарной $(x_0, y_0) = (50.0, 30.0)$.

Для стохастического случая важное значение имеет величина

$$dI(x, y) = \frac{1}{2} \left(\frac{ak_1k_3}{x} + \frac{k_2k_3y}{x} + \frac{ak_1k_2x}{y} + \frac{ak_1k_3}{y} \right),$$

которая характеризует фазовый объем системы и в среднем возрастает в течение эволюции системы [11]. Реализованные в библиотеке слабые и сильные методы были применены к данной задаче для проверки свойств модели с помощью численного моделирования.

С помощью слабого метода Рунге–Кутты было вычислено 500 конкретных реализаций траекторий решения СДУ. На основе этих данных были найдены различные статистические характеристики, такие как медиана и процентили. Эволюция этих характеристик с течением времени иллюстрирует характерные особенности стохастической системы хищник-жертва. Так из рис. 8 видно неуклонное возрастание фазового объема со временем, также как из рис. 9, где изображена эволюция инварианта dI со временем. Видно, что dI в среднем возрастает, хотя и не монотонно.

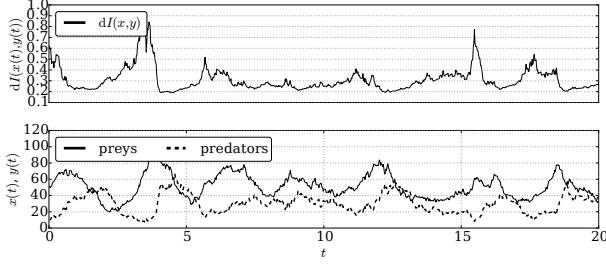


Рис. 6. Эволюция инварианта dI во времени для конкретной реализации траектории стохастической модели хищник-жертва, полученной методом Рунге–Кутты с сильной сходимостью

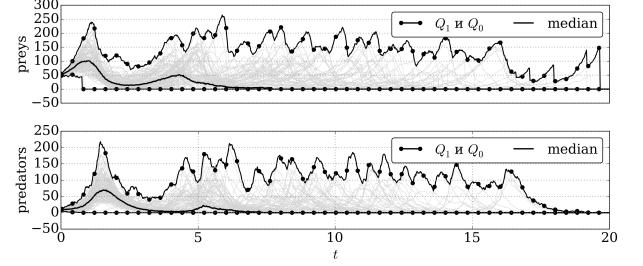


Рис. 7. Множество траекторий решений стохастической модели хищник-жертва (серый) и соответствующие данному множеству траекторий процентиль Q_1, Q_0 и медиана

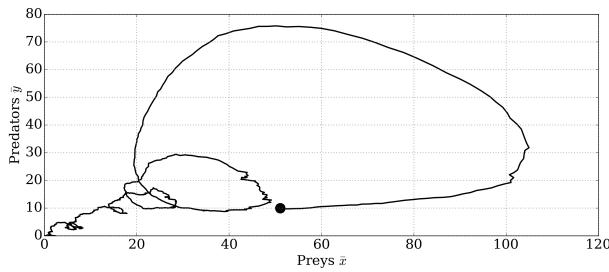


Рис. 8. Фазовый портрет выборочных средних от численности хищников и жертв на основании 500 траекторий, полученных методом Рунге–Кутты со слабой сходимостью

Другое характерное отличие стохастической от классической детерминированной модели проявляется в тенденции к неизбежной гибели популяций в процессе эволюции. На рис. 7 серым цветом обозначены все полученные траектории. Видно, что по мере эволюции системы кривые траекторий пересекают ось абсцисс, что означает гибель одного из представителя популяций. Эта особенность стохастической модели выгодно отличает ее от детерминированной, в которой гибель одного из видов не возможна в принципе.

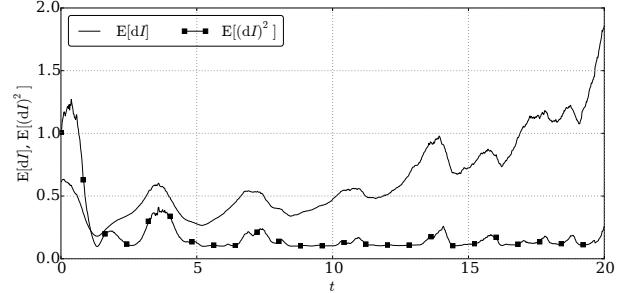


Рис. 9. Первый и второй эмпирические моменты инварианта dI на основании 500 траекторий, полученных методом Рунге–Кутты со слабой сходимостью

VI. ЗАКЛЮЧЕНИЕ

Мы рассмотрели несколько конкретных реализаций стохастического метода Рунге–Кутты. На данный момент в библиотеке используются методы, найденные А. Росслером [9] как наиболее эффективные из известных на данный момент авторам. Однако основные функции `strongSRKW1` и `weakSRKp2Wm` написаны в соответствии с общим алгоритмом и могут использовать любую таблицу Бутчера соответствующей стадийности. Это позволяет пользователю библиотеки расширить ее функционал, добавив новые методы.

Исходный код библиотеки доступен по адресу <https://bitbucket.org/mngev/sde-numerical-integrators>. Там же находятся блокноты со всеми вышеизложенными примерами применения библиотечных функций для решения СДУ. Авторы предполагают активно развивать и дорабатывать библиотеку, добавляя в нее новые функции.

БЛАГОДАРНОСТИ

Работа частично поддержана грантами РФФИ №14-01-00628, 15-07-08795 и 16-07-00556.

Расчёты проведены на вычислительном кластере «Felix» РУДН и на Гетерогенном вычислительном кластере «HybriLIT» Многофункционального центра хранения, обработки и анализа данных ОИЯИ.

Опубликовано в: *Gevorkyan M. N., Velieva T. R., Korolkova A. V., Kulyabov D. S., Sevastyanov L. A. Stochastic Runge–Kutta Software Package for Stochastic Differential Equations // Dependability Engineering and Complex Systems. — Springer International Publishing, 2016. — Vol. 470. — P. 169–179. — doi:10.1007/978-3-319-39639-2_15.*

Исходные тексты: <https://bitbucket.org/yamadharma/articles-2015-rk-stochastic>

- [1] Kloeden P. E., Platen E. Numerical Solution of Stochastic Differential Equations. — 2 edition. — Berlin Heidelberg New York : Springer, 1995. — ISBN: 3-540-54062-8.
- [2] Butcher J. C. Numerical Methods for Ordinary Differential Equations. — 2 edition. — New Zealand : Wiley, 2003. — ISBN: 0-471-96758-0.
- [3] Хайпер Э., Нёрсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи / Под ред. С. С. Филиппова. — 1 изд. — Москва : Мир, 1990. — ISBN: 5-03-001179-X.
- [4] Rößler A. Runge-Kutta Methods for the Numerical Solution of Stochastic Differential Equations : Ph. D. thesis / Andreas Rößler ; Technischen Universität Darmstadt. — Darmstadt, 2003. — februar.
- [5] Burrage K., Burrage P. M. High strong order explicit Runge-Kutta methods for stochastic ordinary differential equations // Appl. Numer. Math. — 1996. — no. 22. — P. 81–101.

- [6] Burrage K., Burrage P. M., Belward J. A. A bound on the maximum strong order of stochastic Runge-Kutta methods for stochastic ordinary differential equations. // BIT. — 1997. — no. 37. — P. 771–780.
- [7] Burrage K., Burrage P. M. General order conditions for stochastic Runge-Kutta methods for both commuting and non-commuting stochastic ordinary differential equation systems // Appl. Numer. Math. — 1998. — no. 28. — P. 161–177.
- [8] Burrage K., Burrage P. M. Order conditions of stochastic Runge-Kutta methods by B-series // SIAM J. Numer. Anal. — 2000. — no. 38. — P. 1626–1646.
- [9] Rößler A. Strong and Weak Approximation Methods for Stochastic Differential Equations – Some Recent Developments / Department Mathematik. Schwerpunkt Mathematische Statistik und Stochastische Prozesse. — 2010. — <http://www.math.uni-hamburg.de/research/papers/prst/prst2010-02.pdf>.
- [10] Stein W. et al. — Sage Mathematics Software (Version 6.5). — The Sage Development Team, 2015. — <http://www.sagemath.org>.
- [11] Demidova A. V., Korolkova A. V., Kulyabov D. S., Sevastianov L. A. The method of stochastization of one-step processes // Mathematical Modeling and Computational Physics. — Dubna : JINR, 2013. — P. 67.
- [12] Demidova A. V., Korolkova A. V., Kulyabov D. S., Sevastyanov L. A. The method of constructing models of peer to peer protocols // 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). — IEEE, 2014. — P. 557–562. — arXiv : 1504.00576.
- [13] Eferina E. G., Korolkova A. V., Gevorkyan M. N. et al. One-Step Stochastic Processes Simulation Software Package // Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics". — 2014. — no. 3. — P. 46–59. — arXiv : 1503.07342.
- [14] Velieva T. R., Korolkova A. V., Kulyabov D. S. Designing installations for verification of the model of active queue management discipline RED in the GNS3 // 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). — IEEE, 2014. — P. 570–577. — arXiv : 1504.02324.
- [15] Higham D. J. An algorithmic introduction to numerical simulation of stochastic differential equations // SIAM Review. — 2001. — Vol. 43, no. 3. — P. 525–546.
- [16] Malham S. J. A., Wiese A. An introduction to SDE simulation / Maxwell Institute for Mathematical Sciences and School of Mathematical and Computer Sciences. — 2010. — arXiv : 1004.0646.
- [17] Platen E., Bruti-Liberati N. Numerical Solution of Stochastic Differential Equations with Jumps in Finance. — Heidelberg Dordrecht London New York : Springer, 2010. — ISBN: 978-3-642-12057-2.
- [18] Оксендалль Б. Стохастические дифференциальные уравнения. Введение в теорию и приложения. — 5 изд. — Москва : Мир, ACT, 2003. — ISBN: 5-03-003477-3.
- [19] 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.
- [20] Debrabant K., Rößler A. Continuous weak approximation for stochastic differential equations // Journal of Computational and Applied Mathematics. — 2008. — no. 214. — P. 259–273.
- [21] Debrabant K., Rößler A. Classification of Stochastic Runge–Kutta Methods for the Weak Approximation of Stochastic Differential Equations / Technische Universität Darmstadt, Fachbereich Mathematik. — 2013. — arXiv : 1303.4510.
- [22] Tocino A., Ardanuy R. Runge–Kutta methods for numerical solution of stochastic differential equations // Journal of Computational and Applied Mathematics. — 2002. — no. 138. — P. 219–241.
- [23] Soheili A. R., Namjoo M. Strong approximation of stochastic differential equations with Runge–Kutta methods // World Journal of Modelling and Simulation. — 2008. — Vol. 4, no. 2. — P. 83–93.
- [24] Mackevičius V. Second-order weak approximations for stratonovich stochastic differential equations // Lithuanian Mathematical Journal. — 1994. — Vol. 34, no. 2. — P. 183–200.
- [25] Milstein G. N. Approximate Integration of Stochastic Differential Equations // Theory Probab. Appl. — 1974. — no. 19. — P. 557–562.
- [26] Milstein G. N. A Method of Second-Order Accuracy Integration of Stochastic Differential Equations // Theory Probab. Appl. — 1979. — no. 23. — P. 396–401.
- [27] Milstein G. N. Weak Approximation of Solutions of Systems of Stochastic Differential Equations // Theory Probab. Appl. — 1986. — no. 30. — P. 750–766.
- [28] Maruyama G. Continuous Markov processes and stochastic equations // Rendiconti del Circolo Matematico. — 1955. — no. 4. — P. 48–90.