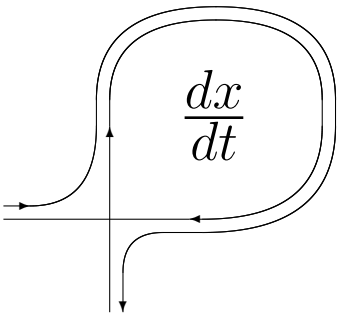


DIFFERENTIAL EQUATIONS
AND
CONTROL PROCESSES
N. 1, 2021
Electronic Journal,
reg. N ΦC77-39410 at 15.04.2010
ISSN 1817-2172

http:// diffjournal.spbu.ru/
e-mail: jodiff@mail.ru

Computer software for the investigation of differential equations, dynamical systems, and control processes
Stochastic differential equations
Numerical methods
Computer modeling in dynamical and control systems

SDE-MATH: A software package for the implementation of strong high-order numerical methods for Itô SDEs with multidimensional non-commutative noise based on multiple Fourier–Legendre series

Mikhail D. Kuznetsov[1], Dmitriy F. Kuznetsov[2]

[1]Faculty of Computer Technologies and Informatics, St. Petersburg Electrotechnical University, Saint-Petersburg, Russia
[2]Institute of Applied Mathematics and Mechanics, Peter the Great St. Petersburg Polytechnic University
e-mail: sde_kuznetsov@inbox.ru

Abstract. The article is devoted to the implementation of strong numerical methods with convergence orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô stochastic differential equations with multidimensional non-commutative noise based on multiple Fourier–Legendre series and unified Taylor–Itô and Taylor–Stratonovich expansions. Algorithms for the implementation of these methods are constructed and a package of programs in the Python programming language is presented. An important part of this software package concerning the

mean-square approximation of iterated Itô and Stratonovich stochastic integrals of multiplicities 1 to 6 with respect to components of the multidimensional Wiener process is based on the method of generalized multiple Fourier series. More precisely, we used multiple Fourier–Legendre series converging in the sense of norm in Hilbert space for the mean-square approximation of iterated Itô and Stratonovich stochastic integrals.

Key words: Software package, Python programming language, numerical method, strong convergence, Itô stochastic differential equation, multidimensional Wiener process, non-commutative noise, unified Taylor–Itô expansion, unified Taylor–Stratonovich expansion, Milstein scheme, high-order strong numerical scheme, iterated Itô stochastic integral, iterated Stratonovich stochastic integral, mean-square approximation, generalized multiple Fourier series, multiple Fourier–Legendre series, Legendre polynomial.

Contents

1 Introduction **94**

2 Theoretical Results Underlying the SDE-MATH Software Package **98**

2.1 Strong Numerical Methods with Convergence Orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs Based on the Unified Taylor–Itô Expansion 98

2.2 Strong Numerical Methods with Convergence Orders 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs Based on the Unified Taylor–Stratonovich Expansion 106

2.3 Method of Expansion and Approximation of Iterated Itô and Stratonovich Stochastic Integrals Based on Generalized Multiple Fourier Series 113

2.4 Approximations of Iterated Itô Stochastic Integrals from the Numerical Schemes (11)–(16) Using Legendre Polynomials 123

2.5 Optimization of Approximations of Iterated Itô Stochastic Integrals from the Numerical Schemes (12)–(16) 132

2.6 Approximations of Iterated Stratonovich Stochastic Integrals from the Numerical Schemes (24)–(28) Using Legendre Polynomials 141

2.7 Numerical Algorithm for Linear Stationary Systems of Itô SDEs Based on Spectral Decomposition 147

3 The Structure of the SDE-MATH Software Package 148

3.1 Development Tools . . . 148

3.2 Dependency Libraries . . . 149

3.3 Architecture . . . 149

3.3.1 Integration with SymPy . . . 151

3.3.2 Purpose of NumPy . . . 151

3.3.3 Purpose of SQLite Database . . . 152

3.3.4 Purpose of Matplotlib . . . 153

3.4 Implementation Plan . . . 153

3.4.1 Calculation of the Fourier–Legendre Coefficients . . . 153

3.4.2 Differential Operators L, $\bar{L}$, $G_0^{(i)}$, $i = 1, \ldots, m$ . . . 154

3.4.3 Approximations of Iterated Stochastic Integrals . . . 154

3.4.4 Strong Numerical Schemes for Itô SDEs . . . 154

3.4.5 Graphical User Interface . . . 154

4 Software Package Graphical User Interface 154

4.1 Information Model of The Graphical User Interface . . . 154

4.1.1 Processing Screens . . . 156

4.1.2 Greetings Dialog . . . 156

4.1.3 Main Menu Dialog . . . 156

4.1.4 Visualization Tool . . . 156

4.1.5 Data Input Dialogs . . . 156

4.2 The User Experience and Implementation Results . . . 157

5 The Results Obtained Using the SDE-MATH Software Package 175

5.1 The Calculated Fourier–Legendre Coefficients . . . 175

5.2 Accuracy Settings . . . 176

5.3 Testing Example (Nonlinear System of Itô SDEs) . . . 179

5.4 Visualization and Numerical Results for Nonlinear System of Itô SDEs Obtained via the SDE-MATH Software Package . . . 180

5.5 Example of Linear System of Itô SDEs (Solar Activity) . . . 211

5.6 Visualization and Numerical Results for Solar Activity Model 211

5.7 Example of Abstract Linear System of Itô SDEs 214

5.8 Visualization and Numerical Results for Abstract Linear System of Itô SDEs Obtained via the SDE-MATH Software Package 215

6 Source Codes of the SDE-MATH Software Package in the Python Programming Language 217

6.1 Source Codes of Graphical User Interface 217

6.1.1 Source Codes of Main Menu 217

6.1.2 Source Codes of Charts Window 230

6.1.3 Source Codes of Input for Nonlinear Systems of Itô SDEs 240

6.1.4 Source Codes of Input for Linear Systems of Itô SDEs 255

6.2 Source Codes for Nonlinear Systems of Itô SDEs 275

6.2.1 Source Codes for Calculation of the Fourier–Legendre Coefficients 275

6.2.2 Source Codes for Supplementary Differential Operators and Functions . . 299

6.2.3 Source Codes for Iterated Itô Stochastic Integrals Approximations Subprograms 303

6.2.4 Source Codes for Iterated Stratonovich Stochastic Integrals Approximations Subprograms 329

6.2.5 Source Codes for Calculation of the Numbers q, q_1,..., q_{15} 351

6.2.6 Source Codes for Strong Taylor–Itô Numerical Schemes with Convergence Orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs 359

6.2.7 Source Codes for Strong Taylor–Stratonovich Numerical Schemes with Convergence Orders 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs 381

6.3 Source Codes for Linear Stationary Systems of Itô SDEs 400

6.4 Source Codes for Utilities and Initialization 407

7 Future Work 412

References 413

1 Introduction

As known, Itô stochastic differential equations (SDEs) have appeared in the theory of random processes relatively recently [1] (1951). Nevertheless, to date, a large number of mathematical models for dynamical systems of different physical nature under the influence of random perturbations have been built on the basis of such equations [2]-[16]. Among them we note mathematical models in stochastic financial mathematics [5]-[7], [10]-[12], geophysics [2], [4], genetics [13], hydrology [2], epidemiology [9], chemical kinetics [2], [9], biology [8], [15], seismology [2], electrodynamics [16] and many other fields [2], [9], [14]. In addition, Itô SDEs arise when solving a number of mathematical problems, such as filtration [2], [3], [17]-[21], stochastic control [2], [17], stochastic stability [2], parameter estimation of stochastic systems [2], [3], [22].

Exact solutions of Itô SDEs are known in rare cases. For this reason, it becomes necessary to construct numerical methods for Itô SDEs. Moreover, the problem of numerical solution of Itô SDEs often occurs even in cases when the exact solution of Itô SDE is known. This means that in some cases, knowing the exact solution of the Itô SDE does not allow us to simulate it numerically in a simple way.

This article is devoted to the implementation of high-order strong numerical methods for systems of Itô SDEs with multidimensional non-commutative noise. More precisely, we consider strong numerical methods with convergence orders 1.0, 1.5, 2.0, 2.5, and 3.0. The article also considers the Euler method, which under suitable conditions [2] has the order 0.5 of strong convergence. To construct the mentioned numerical methods in this article, we use the so-called unified Taylor–Itô and Taylor–Stratonovich expansions [24], [25] (also see [26], Chapter 4). The important components of these expansions are the iterated Itô and Stratonovich stochastic integrals, which are functionals of a complex structure with respect to the components of a multidimensional Wiener process.

It should be noted that it is impossible to construct a numerical method for Itô SDE in a general case (multidimensional non-commutative noise) that includes only increments of the multidimensional Wiener processes, but has a higher order of convergence (in the mean-square sense) than the Euler method (simplest numerical method for Itô SDEs). This result is known as the "Clark–Cameron paradox" [23] (1980) and well explains the need to use high-order numerical methods for Itô SDEs, since the accuracy of the Euler method is insufficient for solving a number of practical problems related to Itô SDEs [2].

According to the "Clark–Cameron paradox" [23], avoidance of the problem of mean-square approximation of the mentioned iterated stochastic integrals is impossible in the general case when constructing high-order strong numerical methods for Itô SDEs.

The problem of mean-square approximation of iterated Itô and Stratonovich stochastic integrals in the context of the numerical integration of Itô SDEs was considered in a number of works [2], [3], [7], [8], [27]-[39].

It should be explained why the results of these works are insufficient for constructing effective procedures for the implementation of strong numerical methods of order 1.5 and higher for Itô SDEs.

There exists an approach to the mean-square approximation of iterated stochastic integrals based on integral sums [27], [34], [35]. Note that one of the variants of this method is based on reducing the problem of mean-square approximation of iterated stochastic integrals to the numerical integration of systems of linear Itô SDEs by the Euler method [39]. However, this approach [27], [34], [35], [39] implies the partitioning of the interval of integration for iterated stochastic integrals. It should be noted that the length of this interval is an integration step for numerical methods for Itô SDEs, which is already a fairly small value even without additional partitioning. Computational experiments show that the numerical modeling of iterated stochastic integrals by the method of integral sums [27], [34], [35], [39] leads to unacceptably high computational cost and accumulation of computation errors [42].

More efficient approach of the mean-square approximation of iterated stochastic integrals is based on the expansion of the so-called Brownian bridge process into the trigonometric Fourier series with random terms (version of the so-called Karhunen–Loève expansion) [2], [3], [7], [27], [28], [33], [34], [37], [38]. However, in [27], [33], [34], [38], this approach was used to approximate only iterated stochastic integrals of multiplicities 1 and 2, which makes it possible to implement numerical method with order 1.0 of strong convergence for Itô SDEs (Milstein method [27]). In papers [2], [3], [7], [28], the approximation of iterated stochastic integrals of multiplicities 1 to 3 was considered by the above approach, which makes it possible to implement numerical methods with orders 1.0 and 1.5 of strong convergence for Itô SDEs. However, formulas concerning integrals of multiplicity 3 turned out to be too complicated and did not find wide application in practice. Moreover, these formulas (for iterated stochastic integrals of multiplicity 3) were obtained without strict theoretical justification and exclude the possibility of effective estimation of the mean-square error of

approximation (see discussion in [26] (Sections 2.6.2, 6.2) for details).

It should be noted that in papers [29], [30], a similar approach was used to approximate iterated stochastic integrals of multiplicities 1 to 3 based on the series expansion of the Wiener process using trigonometric functions and Haar functions. In [40] orthonormal expansions of functions in terms of Walsh series were used to represent the iterated stochastic integrals.

Note that the iterated stochastic integrals under consideration are the random variables with unknown density functions. The only exception is the iterated Itô stochastic integral with multiplicity 2 [31]. However, the knowledge of density function of the mentioned stochastic integral gives no simple way of its approximation [31].

In this work, we use a more efficient method of the mean-square approximation of iterated Itô and Stratonovich stochastic integrals than the methods considered above. This method (the so-called method of generalized multiple Fourier series) is based on the theory constructed in Chapters 1, 2, and 5 of monograph [26] (also see bibliography therein). The method of generalized multiple Fourier series made it possible in this work to successfully implement the procedures for the mean-square approximation of iterated Itô and Stratonovich stochastic integrals of multiplicities 1 to 6. In this case, we use multiple Fourier–Legendre series, that is, we have chosen Legendre polynomials as a basis system of functions for approximating iterated stochastic integrals. It is important to note that the Legendre polynomials were first applied in the context of this problem in [43] (1997), while in the works of other authors Legendre polynomials were not considered as a system of basis functions for approximating iterated stochastic integrals (an exception is work [36]). As shown in [44], the Legendre polynomials are an optimal system of basis functions for approximating iterated Itô and Stratonovich stochastic integrals.

In this article, to build the SDE-MATH software package in the Python programming language, we use a database with 270,000 exactly calculated Fourier–Legendre coefficients to approximate iterated Itô and Stratonovich stochastic integrals of multiplicities 1 to 6. It should be noted that the procedures for the mean-square approximation of iterated stochastic integrals of multiplicities 4, 5, and 6 constructed in this work have no analogues in the literature. At the same time, we propose a much more convenient procedure for the mean-square approximation of iterated stochastic integrals of multiplicity 3 than in works [2], [3], [7], [28]. This procedure provides an accurate calculation of the mean-square error of approximation of the mentioned stochastic integrals.

Another important feature of the presented software package is the use of unified Taylor–Itô and Taylor–Stratonovich expansions [24], [25] (also see [26], Chapter 4) for constructing strong numerical methods with convergence orders 1.5, 2.0, 2.5, and 3.0 for Itô SDEs. Unified Taylor–Itô and Taylor—Stratonovich expansions make it possible (in contrast with its classical analogues [2]) to use the minimal sets of iterated Itô and Stratonovich stochastic integrals. This property well explains the motive for using the mentioned unified expansions.

The results of this work on the approximation of iterated stochastic integrals can be used to numerically solve various types of SDEs. For example, for semilinear SPDEs with multiplicative trace class noise [26] (Chapter 7), [45], [46]. This is due to the fact that iterated stochastic integrals are a universal tool for constructing high-order strong numerical methods for various types of SDEs. In recent years, the mentioned numerical methods have been constructed for SDEs with jumps [7], SPDEs with multiplicative trace class noise [47]-[49], McKean SDEs [50], SDEs with switchings [51], mean-field SDEs [52], Itô–Volterra stochastic integral equations [49], etc.

There are many publications in which codes of programs in various programming languages are given for the numerical solution of SDEs [3], [9], [14], [53]-[61]. Among them, we note the software described in [3], [54], [56], [60]. Some of the mentioned works [3], [54], [56], [57], [60] are based on the results of monograph [2] on the approximation of iterated stochastic integrals (see above discusson on the disadvantages of approach [2]). Other publications [9], [14], [53], [55] do not use the modeling of iterated stochastic integrals for the case of multidimensional non-commutative noise at all.

In this article, we develop software for the numerical integration of Itô SDEs based on theoretical results and MATLAB codes from monographs [58], [61] for modeling iterated stochastic integrals of multiplicities 1 to 6 (the case of multidimensional non-commutative noise). In addition, we provide software (as a part of the SDE-MATH software package) for the numerical integration of linear stationary systems of Itô SDEs based on the results of article [62] and MATLAB codes from monographs [58], [61].

In Section 7 we discuss possible directions for the development of the SDE-MATH software package. In particular, the parallelization of computations, the implementation of methods of the Runge–Kutta type [2], [7], [42], [61] and multistep numerical methods for Itô SDEs [2], [7], [42], [61], the development of a part of the software package for solving filtering problem and stochastic optimal control problem [2], as well as improvement of the graphical user interface.

2 Theoretical Results Underlying the SDE-MATH Software Package

2.1 Strong Numerical Methods with Convergence Orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs Based on the Unified Taylor–Itô Expansion

Let $(\Omega,\ \mathrm{F},\ \mathsf{P})$ be a complete probability space and let $\{\mathrm{F}_t, t \in [0,T]\}$ be a nondecreasing right-continuous family of σ-algebras of F. Let $\mathbf{w}_t$ be a standard m-dimensional Wiener stochastic process with independent components $\mathbf{w}_t^{(i)}$ $(i = 1,\ldots,m)$, which is F_t-measurable for any $t \in [0,T]$. Consider an Itô SDE in the integral form

$$\mathbf{x}_t = \mathbf{x}_0 + \int\limits_0^t \mathbf{a}(\mathbf{x}_\tau,\tau)d\tau + \sum_{i=1}^m \int\limits_0^t B_i(\mathbf{x}_\tau,\tau)d\mathbf{w}_\tau^{(i)}, \quad \mathbf{x}_0 = \mathbf{x}(0,\omega), \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is a solution of the Itô SDE (1), the nonrandom functions $\mathbf{a}(\mathbf{x},t) : \mathbb{R}^n \times [0,T] \to \mathbb{R}^n$, $B(\mathbf{x},t) : \mathbb{R}^n \times [0,T] \to \mathbb{R}^{n\times m}$ guarantee the existence and uniqueness up to stochastic equivalence of a solution of (1) [63], the second integral on the right-hand side of (1) is interpreted as an Itô stochastic integral, $B_i(\mathbf{x},t)$ is the ith colomn of the matrix function $B(\mathbf{x},t)$, $\mathbf{x}_0$ is an n-dimensional and F_0-measurable random variable, $\mathsf{M}\{|\mathbf{x}_0|^2\} < \infty$ (M is an expectation operator). We assume that $\mathbf{x}_0$ and $\mathbf{w}_t - \mathbf{w}_0$ are independent when $t > 0$.

It is well known that one of the effective approaches to the numerical integration of Itô SDEs is an approach based on the Taylor–Itô and Taylor–Stratonovich expansions [2], [7], [42]. The essential feature of such expansions are the so-called iterated Itô and Stratonovich stochastic integrals, which have the form

$$J[\psi^{(k)}]_{T,t} = \int\limits_t^T \psi_k(t_k)\ldots\int\limits_t^{t_2} \psi_1(t_1)d\mathbf{w}_{t_1}^{(i_1)}\ldots d\mathbf{w}_{t_k}^{(i_k)}, \tag{2}$$

$$J^*[\psi^{(k)}]_{T,t} = \int\limits_t^{*T} \psi_k(t_k)\ldots\int\limits_t^{*t_2} \psi_1(t_1)d\mathbf{w}_{t_1}^{(i_1)}\ldots d\mathbf{w}_{t_k}^{(i_k)}, \tag{3}$$

where every $\psi_l(\tau)$ $(l = 1,\ldots,k)$ is a continuous nonrandom function on $[t,T]$,

$\mathbf{w}_\tau^{(i)}$ $(i = 1, \ldots, m)$ are independent standard Wiener processes and $\mathbf{w}_\tau^{(0)} \stackrel{\mathsf{def}}{=} \tau$,

$$\int \text{ and } \overset{*}{\int}$$

denote Itô and Stratonovich stochastic integrals, respectively; $i_1, \ldots, i_k = 0, 1, \ldots, m$.

Note that $\psi_l(\tau) \equiv 1$ $(l = 1, \ldots, k)$ and $i_1, \ldots, i_k = 0, 1, \ldots, m$ in the classical Taylor–Itô and Taylor–Stratonovich expansions [2]. At the same time $\psi_l(\tau) \equiv (t - \tau)^{q_l}$ $(l = 1, \ldots, k;\ q_1, \ldots, q_k = 0, 1, 2, \ldots)$ and $i_1, \ldots, i_k = 1, \ldots, m$ in the unified Taylor–Itô and Taylor–Stratonovich expansions [24], [25] (also see [26], Chapter 4).

Let $C^{2,1}(\mathbb{R}^n \times [0, T])$ be the space of functions $R(\mathbf{x}, t) : \mathbb{R}^n \times [0, T] \to \mathbb{R}^1$ with the following property: these functions are twice continuously differentiable in $\mathbf{x}$ and have one continuous derivative in t. Let us consider the following differential operators on the space $C^{2,1}(\mathbb{R}^n \times [0, T])$

$$L = \frac{\partial}{\partial t} + \sum_{i=1}^{n} a^{(i)}(\mathbf{x}, t) \frac{\partial}{\partial \mathbf{x}^{(i)}} + \frac{1}{2} \sum_{j=1}^{m} \sum_{l,i=1}^{n} B^{(lj)}(\mathbf{x}, t) B^{(ij)}(\mathbf{x}, t) \frac{\partial^2}{\partial \mathbf{x}^{(l)} \partial \mathbf{x}^{(i)}}, \qquad (4)$$

$$G_0^{(i)} = \sum_{j=1}^{n} B^{(ji)}(\mathbf{x}, t) \frac{\partial}{\partial \mathbf{x}^{(j)}}, \quad i = 1, \ldots, m, \qquad (5)$$

where $a^{(i)}(\mathbf{x}, t)$ is the ith component of the vector function $a(\mathbf{x}, t)$ and $B^{(ij)}(\mathbf{x}, t)$ is the ijth component of the matrix function $B(\mathbf{x}, t)$.

Consider the following sequence of differential operators

$$G_p^{(i)} = \frac{1}{p} \left(G_{p-1}^{(i)} L - L G_{p-1}^{(i)} \right), \quad p = 1, 2, \ldots, \quad i = 1, \ldots, m,$$

where L and $G_0^{(i)}$, $i = 1, \ldots, m$ are defined by (4), (5).

For the further consideration, we need to introduce the following set of iterated Itô stochastic integrals

$$I_{(l_1 \ldots l_k) s,t}^{(i_1 \ldots i_k)} = \int_t^s (t - t_k)^{l_k} \ldots \int_t^{t_2} (t - t_1)^{l_1} d\mathbf{w}_{t_1}^{(i_1)} \ldots d\mathbf{w}_{t_k}^{(i_k)}, \qquad (6)$$

where $l_1, \ldots, l_k = 0, 1, \ldots$ and $i_1, \ldots, i_k = 1, \ldots, m$.

Assume that $R(\mathbf{x},t)$, $\mathbf{a}(\mathbf{x},t)$, and $B_i(\mathbf{x},t)$, $i=1,\ldots,m$ are enough smooth functions with respect to the variables $\mathbf{x}$ and t. Then for all $s,t\in[0,T]$ such that $s>t$ we can write the following unified Taylor–Itô expansion [24] (also see [26], Chapter 4)

$$R(\mathbf{x}_s,s)=$$

$$=R(\mathbf{x}_t,t)+\sum_{q=1}^{r}\sum_{(k,j,l_1,\ldots,l_k)\in \mathrm{D}_q}\frac{(s-t)^j}{j!}\sum_{i_1,\ldots,i_k=1}^{m}G_{l_1}^{(i_1)}\ldots G_{l_k}^{(i_k)}L^jR(\mathbf{x}_t,t)\ I_{(l_1\ldots l_k)s,t}^{(i_1\ldots i_k)}+$$

$$+\left(H_{r+1}\right)_{s,t}\quad \text{w. p. 1}, \tag{7}$$

where

$$L^jR(\mathbf{x},t)\stackrel{\mathsf{def}}{=}\begin{cases}\underbrace{L\ldots L}_{j}R(\mathbf{x},t) & \text{for } j\geq 1\\ \\ R(\mathbf{x},t) & \text{for } j=0\end{cases},$$

$$\mathrm{D}_q=\left\{(k,j,l_1,\ldots,l_k):\ k+2\left(j+\sum_{p=1}^{k}l_p\right)=q;\ k,j,l_1,\ldots,l_k=0,1,\ldots\right\}, \tag{8}$$

and $\left(H_{r+1}\right)_{s,t}$ is the remainder term in integral form [26].

Consider the partition $\{\tau_p\}_{p=0}^{N}$ of the interval $[0,T]$ such that

$$0=\tau_0<\tau_1<\ldots<\tau_N=T,\quad \Delta_N=\max_{0\leq j\leq N-1}\left|\tau_{j+1}-\tau_j\right|. \tag{9}$$

Let $\mathbf{y}_{\tau_j}\stackrel{\mathsf{def}}{=}\mathbf{y}_j$, $j=0,1,\ldots,N$ be a time discrete approximation of the process $\mathbf{x}_t$, $t\in[0,T]$, which is a solution of the Itô SDE (1).

Definiton 1 [2]. *We will say that a time discrete approximation* $\mathbf{y}_j$, $j=0,1,\ldots,N$, *corresponding to the maximal step of discretization* Δ_N, *converges strongly with order* $\gamma>0$ *at time moment* T *to the process* $\mathbf{x}_t$, $t\in[0,T]$, *if there exists a constant* $C>0$, *which does not depend on* Δ_N, *and a* $\delta>0$ *such that* $\mathsf{M}\{|\mathbf{x}_T-\mathbf{y}_T|\}\leq C(\Delta_N)^{\gamma}$ *for each* $\Delta_N\in(0,\delta)$.

From (7) for $s=\tau_{p+1}$ and $t=\tau_p$ we obtain the following representation of explicit one-step strong numerical scheme for the Itô SDE (1)

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{q=1}^{r}\sum_{(k,j,l_1,\ldots,l_k)\in \mathrm{D}_q}\frac{(\tau_{p+1}-\tau_p)^j}{j!}\sum_{i_1,\ldots,i_k=1}^{m}G_{l_1}^{(i_1)}\ldots G_{l_k}^{(i_k)}L^j\mathbf{y}_p\ \hat{I}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}^{(i_1\ldots i_k)}+$$

$$+\mathbf{1}_{\{r=2d-1,d\in\mathbb{N}\}}\frac{(\tau_{p+1}-\tau_p)^{(r+1)/2}}{((r+1)/2)!}L^{(r+1)/2}\mathbf{y}_p, \tag{10}$$

where $\hat{I}^{(i_1\ldots i_k)}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}$ is an approximation of the iterated Itô stochastic integral (6) and $\mathbf{1}_A$ is the indicator of the set A. Note that we understand the equality (10) componentwise with respect to the components $\mathbf{y}_p^{(i)}$ of the column $\mathbf{y}_p$. Also for simplicity we put $\tau_p = p\Delta$, $\Delta = T/N$, $p = 0, 1, \ldots, N$.

Under the appropriate conditions [2] the numerical scheme (10) has strong order $r/2$ ($r \in \mathbb{N}$) of convergence.

Below we consider particular cases of the numerical scheme (10) for $r = 1, 2, 3, 4, 5$, and 6, i.e. explicit one-step strong numerical schemes with convergence orders $0.5, 1.0, 1.5, 2.0, 2.5$, and 3.0 for the Itô SDE (1) [26], [64], [65]. At that for simplicity we will write $\mathbf{a}$, $L\mathbf{a}$, B_i, $G_0^{(i)}B_j$ etc. instead of $\mathbf{a}(\mathbf{y}_p,\tau_p)$, $L\mathbf{a}(\mathbf{y}_p,\tau_p)$, $B_i(\mathbf{y}_p,\tau_p)$, $G_0^{(i)}B_j(\mathbf{y}_p,\tau_p)$ etc. correspondingly. Moreover, the operators L and $G_0^{(i)}$, $i = 1, \ldots, m$ are defined by (4), (5).

Scheme with strong order 0.5 (Euler scheme)

$$\mathbf{y}_{p+1} = \mathbf{y}_p + \sum_{i_1=1}^{m} B_{i_1}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p} + \Delta\mathbf{a}. \tag{11}$$

Scheme with strong order 1.0 (Milstein scheme)

$$\mathbf{y}_{p+1} = \mathbf{y}_p + \sum_{i_1=1}^{m} B_{i_1}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p} + \Delta\mathbf{a} + \sum_{i_1,i_2=1}^{m} G_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}. \tag{12}$$

Scheme with strong order 1.5

$$\begin{aligned}\mathbf{y}_{p+1} = \mathbf{y}_p &+ \sum_{i_1=1}^{m} B_{i_1}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p} + \Delta\mathbf{a} + \sum_{i_1,i_2=1}^{m} G_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p} + \\ &+ \sum_{i_1=1}^{m}\left[G_0^{(i_1)}\mathbf{a}\left(\Delta\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p} + \hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right) - LB_{i_1}\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right] + \\ &+ \sum_{i_1,i_2,i_3=1}^{m} G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p} + \frac{\Delta^2}{2}L\mathbf{a}.\end{aligned} \tag{13}$$

Scheme with strong order 2.0

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\mathbf{a}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$

$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\mathbf{a}\left(\Delta\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-LB_{i_1}\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}L\mathbf{a}+$$

$$+\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}LB_{i_2}\left(\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)-LG_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\right.$$

$$\left.+G_0^{(i_1)}G_0^{(i_2)}\mathbf{a}\left(\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}. \qquad (14)$$

Scheme with strong order 2.5

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\mathbf{a}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$

$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\mathbf{a}\left(\Delta\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-LB_{i_1}\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}L\mathbf{a}+$$

$$+\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}LB_{i_2}\left(\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)-LG_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\right.$$

$$\left.+G_0^{(i_1)}G_0^{(i_2)}\mathbf{a}\left(\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}+$$

$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}L\mathbf{a}\left(\frac{1}{2}\hat{I}^{(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}\right)+\right.$$

$$\left.+\frac{1}{2}LLB_{i_1}\hat{I}^{(i_1)}_{(2)\tau_{p+1},\tau_p}-LG_0^{(i_1)}\mathbf{a}\left(\hat{I}^{(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}\left[G_0^{(i_1)}LG_0^{(i_2)}B_{i_3}\left(\hat{I}^{(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}\right)+\right.$$

$$+G_0^{(i_1)}G_0^{(i_2)}LB_{i_3}\left(\hat{I}^{(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}\mathbf{a}\left(\Delta\hat{I}^{(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\hat{I}^{(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)-$$

$$\left.-LG_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4,i_5=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}B_{i_5}\hat{I}^{(i_1i_2i_3i_4i_5)}_{(00000)\tau_{p+1},\tau_p}+$$

$$+\frac{\Delta^3}{6}LL\mathbf{a}. \tag{15}$$

Scheme with strong order 3.0

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\mathbf{a}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$

$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\mathbf{a}\left(\Delta\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-LB_{i_1}\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}L\mathbf{a}+$$

$$+\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}LB_{i_2}\left(\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)-LG_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\right.$$

$$\left.+G_0^{(i_1)}G_0^{(i_2)}\mathbf{a}\left(\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}+\mathbf{q}_{p+1,p}+\mathbf{r}_{p+1,p}, \tag{16}$$

where

$$\mathbf{q}_{p+1,p}=\sum_{i_1=1}^{m}\Bigg[G_0^{(i_1)}L\mathbf{a}\left(\frac{1}{2}\hat{I}^{(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\hat{I}^{(i_1)}_{(0)\tau_{p+1},\tau_p}\right)+$$

$$+\frac{1}{2}LLB_{i_1}\hat{I}^{(i_1)}_{(2)\tau_{p+1},\tau_p}-LG_0^{(i_1)}\mathbf{a}\left(\hat{I}^{(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1)}_{(1)\tau_{p+1},\tau_p}\right)\Bigg]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}\Bigg[G_0^{(i_1)}LG_0^{(i_2)}B_{i_3}\left(\hat{I}^{(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}LB_{i_3}\left(\hat{I}^{(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}\mathbf{a}\left(\Delta\hat{I}^{(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\hat{I}^{(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)-$$

$$-LG_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}\Bigg]+$$

$$+\sum_{i_1,i_2,i_3,i_4,i_5=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}B_{i_5}\hat{I}^{(i_1i_2i_3i_4i_5)}_{(00000)\tau_{p+1},\tau_p}+$$

$$+\frac{\Delta^3}{6}LL\mathbf{a},$$

and

$$\mathbf{r}_{p+1,p}=\sum_{i_1,i_2=1}^{m}\Bigg[G_0^{(i_1)}G_0^{(i_2)}L\mathbf{a}\left(\frac{1}{2}\hat{I}^{(i_1i_2)}_{(02)\tau_{p+1},\tau_p}+\Delta\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\hat{I}^{(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)+$$

$$+\frac{1}{2}LLG_0^{(i_1)}B_{i_2}\hat{I}^{(i_1i_2)}_{(20)\tau_{p+1},\tau_p}+$$

$$+G_0^{(i_1)}LG_0^{(i_2)}\mathbf{a}\left(\hat{I}^{(i_1i_2)}_{(11)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(02)\tau_{p+1},\tau_p}+\Delta\left(\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)\right)+$$

$$+LG_0^{(i_1)}LB_{i_2}\left(\hat{I}^{(i_1i_2)}_{(11)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(20)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}LLB_{i_2}\left(\frac{1}{2}\hat{I}^{(i_1i_2)}_{(02)\tau_{p+1},\tau_p}+\frac{1}{2}\hat{I}^{(i_1i_2)}_{(20)\tau_{p+1},\tau_p}-\hat{I}^{(i_1i_2)}_{(11)\tau_{p+1},\tau_p}\right)-$$

$$-LG_0^{(i_1)}G_0^{(i_2)}\mathbf{a}\left(\Delta\hat{I}^{(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\hat{I}^{(i_1i_2)}_{(11)\tau_{p+1},\tau_p}\right)\Bigg]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m}\Bigg[G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}\mathbf{a}\left(\Delta\hat{I}_{(0000)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}+\hat{I}_{(0001)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}LG_0^{(i_3)}B_{i_4}\left(\hat{I}_{(0100)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}-\hat{I}_{(0010)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}\right)-$$

$$-LG_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}_{(1000)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}+$$

$$+G_0^{(i_1)}LG_0^{(i_2)}G_0^{(i_3)}B_{i_4}\left(\hat{I}_{(1000)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}-\hat{I}_{(0100)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}LB_{i_4}\left(\hat{I}_{(0010)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}-\hat{I}_{(0001)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4)}\right)\Bigg]+$$

$$+\sum_{i_1,i_2,i_3,i_4,i_5,i_6=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}G_0^{(i_5)}B_{i_6}\hat{I}_{(000000)\tau_{p+1},\tau_p}^{(i_1i_2i_3i_4i_5i_6)}.$$

Under the suitable conditions [2] the numerical schemes (12)–(16) have strong orders 1.0, 1.5, 2.0, 2.5, and 3.0 of convergence correspondingly. Among these conditions we consider only the condition for approximations of iterated Itô stochastic integrals from (12)–(16) [2] (also see [42])

$$\mathsf{M}\left\{\left(I_{(l_1\ldots l_k)\tau_{p+1},\tau_p}^{(i_1\ldots i_k)}-\hat{I}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}^{(i_1\ldots i_k)}\right)^2\right\}\le C\Delta^{r+1},\qquad(17)$$

where constant C is independent of Δ and $r/2$ are the strong convergence orders for the numerical schemes (12)–(16), i.e. $r/2=1.0,1.5,2.0,2.5$, and 3.0.

Note that the numerical schemes (12)–(16) are unrealizable in practice without procedures for the numerical simulation of iterated Itô stochastic integrals from (10). In Section 2.3 we give a brief overview of the effective method of the mean-square approximation of iterated Itô and Stratonovich stochastic integrals of arbitrary multiplicity k ($k\in\mathbb{N}$).

2.2 Strong Numerical Methods with Convergence Orders 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs Based on the Unified Taylor–Stratonovich Expansion

Let us consider the following differential operator on the space $C^{2,1}(\mathbb{R}^n\times[0,T])$

$$\bar{L} = L - \frac{1}{2}\sum_{i=1}^{m} G_0^{(i)} G_0^{(i)}, \tag{18}$$

where operators L and $G_0^{(i)}$, $i = 1, \ldots, m$ are defined by (4), (5).

Define the following sequence of differential operators

$$\bar{G}_p^{(i)} \stackrel{\mathsf{def}}{=} \frac{1}{p}\left(\bar{G}_{p-1}^{(i)}\bar{L} - \bar{L}\bar{G}_{p-1}^{(i)}\right), \quad p = 1, 2, \ldots, \quad i = 1, \ldots, m, \tag{19}$$

where $\bar{G}_0^{(i)} \stackrel{\mathsf{def}}{=} G_0^{(i)}$, $i = 1, \ldots, m$. The operators $\bar{L}$ and $G_0^{(i)}$, $i = 1, \ldots, m$ are defined by (18) and (5) correspondingly.

For the further consideration, we need to introduce the following set of iterated Stratonovich stochastic integrals

$$I_{(l_1 \ldots l_k)s,t}^{*(i_1 \ldots i_k)} = \int_t^{*s} (t - t_k)^{l_k} \ldots \int_t^{*t_2} (t - t_1)^{l_1} d\mathbf{w}_{t_1}^{(i_1)} \ldots d\mathbf{w}_{t_k}^{(i_k)}, \tag{20}$$

where $l_1, \ldots, l_k = 0, 1, \ldots$ and $i_1, \ldots, i_k = 1, \ldots, m$.

Assume that $R(\mathbf{x}, t)$, $\mathbf{a}(\mathbf{x}, t)$, and $B_i(\mathbf{x}, t)$, $i = 1, \ldots, m$ are enough smooth functions with respect to the variables $\mathbf{x}$ and t. Then for all $s, t \in [0, T]$ such that $s > t$ we can write the following unified Taylor–Stratonovich expansion [25] (also see [26], Chapter 4)

$$R(\mathbf{x}_s, s) =$$

$$= R(\mathbf{x}_t, t) + \sum_{q=1}^{r} \sum_{(k, j, l_1, \ldots, l_k) \in \mathrm{D}_q} \frac{(s-t)^j}{j!} \sum_{i_1, \ldots, i_k = 1}^{m} \bar{G}_{l_1}^{(i_1)} \ldots \bar{G}_{l_k}^{(i_k)} \bar{L}^j R(\mathbf{x}_t, t)\ I_{(l_1 \ldots l_k)s,t}^{*(i_1 \ldots i_k)} +$$

$$+ \left(\bar{H}_{r+1}\right)_{s,t} \quad \text{w. p. 1}, \tag{21}$$

where

$$\bar{L}^j R(\mathbf{x}, t) \stackrel{\mathsf{def}}{=} \begin{cases} \underbrace{\bar{L} \ldots \bar{L}}_{j} R(\mathbf{x}, t) & \text{for } j \geq 1 \\ \\ R(\mathbf{x}, t) & \text{for } j = 0 \end{cases},$$

the set D_q is defined by the equality (8) and $\left(\bar{H}_{r+1}\right)_{s,t}$ is the remainder term in integral form [25] (also see [26], Chapter 4).

Consider the partition (9) of the interval $[0,T]$. From (21) for $s=\tau_{p+1}$ and $t=\tau_p$ we obtain the following representation of explicit one-step strong numerical scheme for the Itô SDE (1)

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{q=1}^{r}\sum_{(k,j,l_1,\ldots,l_k)\in \mathrm{D}_q}\frac{(\tau_{p+1}-\tau_p)^j}{j!}\sum_{i_1,\ldots,i_k=1}^{m}\bar{G}_{l_1}^{(i_1)}\ldots\bar{G}_{l_k}^{(i_k)}\bar{L}^j\mathbf{y}_p\,\hat{I}^{*(i_1\ldots i_k)}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}+$$

$$+\mathbf{1}_{\{r=2d-1,d\in N\}}\frac{(\tau_{p+1}-\tau_p)^{(r+1)/2}}{((r+1)/2)!}L^{(r+1)/2}\mathbf{y}_p, \tag{22}$$

where $\hat{I}^{*(i_1\ldots i_k)}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}$ is an approximation of the iterated Stratonovich stochastic integral (20) and $\mathbf{1}_A$ is the indicator of the set A. Note that we understand the equality (22) componentwise with respect to the components $\mathbf{y}_p^{(i)}$ of the column $\mathbf{y}_p$. Also for simplicity we put $\tau_p=p\Delta$, $\Delta=T/N$, $p=0,1,\ldots,N$.

Under the appropriate conditions [2] the numerical scheme (22) has strong order $r/2$ ($r\in\mathbb{N}$) of convergence.

Denote

$$\bar{\mathbf{a}}(\mathbf{x},t)=\mathbf{a}(\mathbf{x},t)-\frac{1}{2}\sum_{j=1}^{m}G_0^{(j)}B_j(\mathbf{x},t),$$

where $B_j(\mathbf{x},t)$ is the jth column of the matrix function $B(\mathbf{x},t)$.

It is not difficult to show that (see (18))

$$\bar{L}=\frac{\partial}{\partial t}+\sum_{i=1}^{n}\bar{\mathbf{a}}^{(i)}(\mathbf{x},t)\frac{\partial}{\partial\mathbf{x}^{(i)}}, \tag{23}$$

where $\bar{\mathbf{a}}^{(i)}(\mathbf{x},t)$ is the ith component of the vector function $\bar{\mathbf{a}}(\mathbf{x},t)$.

Below we consider particular cases of the numerical scheme (22) for $r=2,3,4,5$, and 6, i.e. explicit one-step strong numerical schemes with convergence orders $1.0, 1.5, 2.0, 2.5$, and 3.0 for the Itô SDE (1) [26], [66], [67]. At that for simplicity we will write $\bar{\mathbf{a}}$, $\bar{L}\bar{\mathbf{a}}$, $L\mathbf{a}$, B_i, $G_0^{(i)}B_j$ etc. instead of $\bar{\mathbf{a}}(\mathbf{y}_p,\tau_p)$, $\bar{L}\bar{\mathbf{a}}(\mathbf{y}_p,\tau_p)$, $L\mathbf{a}(\mathbf{y}_p,\tau_p)$, $B_i(\mathbf{y}_p,\tau_p)$, $G_0^{(i)}B_j(\mathbf{y}_p,\tau_p)$ etc. correspondingly.

Scheme with strong order 1.0

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\bar{\mathbf{a}}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}. \tag{24}$$

Scheme with strong order 1.5

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\bar{\mathbf{a}}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$
$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-\bar{L}B_{i_1}\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$
$$+\sum_{i_1,i_2,i_3=1}^{m}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{*(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}L\mathbf{a}. \tag{25}$$

Scheme with strong order 2.0

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\bar{\mathbf{a}}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$
$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-\bar{L}B_{i_1}\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$
$$+\sum_{i_1,i_2,i_3=1}^{m}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{*(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\bar{L}\bar{\mathbf{a}}+$$
$$+\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}\bar{L}B_{i_2}\left(\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)-\bar{L}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\right.$$
$$\left.+G_0^{(i_1)}G_0^{(i_2)}\bar{\mathbf{a}}\left(\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)\right]+$$
$$+\sum_{i_1,i_2,i_3,i_4=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{*(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}. \tag{26}$$

Scheme with strong order 2.5

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\bar{\mathbf{a}}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$
$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-\bar{L}B_{i_1}\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m} G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{*(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\bar{L}\bar{\mathbf{a}}+$$

$$+\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}\bar{L}B_{i_2}\left(\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)-\bar{L}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\right.$$

$$\left.+G_0^{(i_1)}G_0^{(i_2)}\bar{\mathbf{a}}\left(\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m} G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{*(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}+$$

$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\bar{L}\bar{\mathbf{a}}\left(\frac{1}{2}\hat{I}^{*(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}\right)+\right.$$

$$\left.+\frac{1}{2}\bar{L}\bar{L}B_{i_1}\hat{I}^{*(i_1)}_{(2)\tau_{p+1},\tau_p}-\bar{L}G_0^{(i_1)}\bar{\mathbf{a}}\left(\hat{I}^{*(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}\left[G_0^{(i_1)}\bar{L}G_0^{(i_2)}B_{i_3}\left(\hat{I}^{*(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}\right)+\right.$$

$$+G_0^{(i_1)}G_0^{(i_2)}\bar{L}B_{i_3}\left(\hat{I}^{*(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)-$$

$$\left.-\bar{L}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{*(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4,i_5=1}^{m} G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}B_{i_5}\hat{I}^{*(i_1i_2i_3i_4i_5)}_{(00000)\tau_{p+1},\tau_p}+$$

$$+\frac{\Delta^3}{6}LL\mathbf{a}. \tag{27}$$

Scheme with strong order 3.0

$$\mathbf{y}_{p+1}=\mathbf{y}_p+\sum_{i_1=1}^{m}B_{i_1}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\Delta\bar{\mathbf{a}}+\sum_{i_1,i_2=1}^{m}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}+$$

$$+\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right)-\bar{L}B_{i_1}\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m} G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{*(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\bar{L}\bar{\mathbf{a}}+$$

$$+\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}\bar{L}B_{i_2}\left(\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)-\bar{L}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\right.$$

$$\left.+G_0^{(i_1)}G_0^{(i_2)}\bar{\mathbf{a}}\left(\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m} G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{*(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}+\mathbf{q}_{p+1,p}+\mathbf{r}_{p+1,p}, \tag{28}$$

where

$$\mathbf{q}_{p+1,p}=\sum_{i_1=1}^{m}\left[G_0^{(i_1)}\bar{L}\bar{\mathbf{a}}\left(\frac{1}{2}\hat{I}^{*(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\hat{I}^{*(i_1)}_{(0)\tau_{p+1},\tau_p}\right)+\right.$$

$$\left.+\frac{1}{2}\bar{L}\bar{L}B_{i_1}\hat{I}^{*(i_1)}_{(2)\tau_{p+1},\tau_p}-\bar{L}G_0^{(i_1)}\bar{\mathbf{a}}\left(\hat{I}^{*(i_1)}_{(2)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1)}_{(1)\tau_{p+1},\tau_p}\right)\right]+$$

$$+\sum_{i_1,i_2,i_3=1}^{m}\left[G_0^{(i_1)}\bar{L}G_0^{(i_2)}B_{i_3}\left(\hat{I}^{*(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}\right)+\right.$$

$$+G_0^{(i_1)}G_0^{(i_2)}\bar{L}B_{i_3}\left(\hat{I}^{*(i_1i_2i_3)}_{(010)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1i_2i_3)}_{(000)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1i_2i_3)}_{(001)\tau_{p+1},\tau_p}\right)-$$

$$\left.-\bar{L}G_0^{(i_1)}G_0^{(i_2)}B_{i_3}\hat{I}^{*(i_1i_2i_3)}_{(100)\tau_{p+1},\tau_p}\right]+$$

$$+\sum_{i_1,i_2,i_3,i_4,i_5=1}^{m} G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}B_{i_5}\hat{I}^{*(i_1i_2i_3i_4i_5)}_{(00000)\tau_{p+1},\tau_p}+$$

$$+\frac{\Delta^3}{6}\bar{L}\bar{L}\bar{\mathbf{a}},$$

and

$$\mathbf{r}_{p+1,p}=\sum_{i_1,i_2=1}^{m}\left[G_0^{(i_1)}G_0^{(i_2)}\bar{L}\bar{\mathbf{a}}\left(\frac{1}{2}\hat{I}^{*(i_1i_2)}_{(02)\tau_{p+1},\tau_p}+\Delta\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}+\frac{\Delta^2}{2}\hat{I}^{*(i_1i_2)}_{(00)\tau_{p+1},\tau_p}\right)+\right.$$

$$\left.+\frac{1}{2}\bar{L}\bar{L}G_0^{(i_1)}B_{i_2}\hat{I}^{*(i_1i_2)}_{(20)\tau_{p+1},\tau_p}+\right.$$

$$+G_0^{(i_1)}\bar{L}G_0^{(i_2)}\bar{\mathbf{a}}\left(\hat{I}^{*(i_1i_2)}_{(11)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(02)\tau_{p+1},\tau_p}+\Delta\left(\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(01)\tau_{p+1},\tau_p}\right)\right)+$$

$$+\bar{L}G_0^{(i_1)}\bar{L}B_{i_2}\left(\hat{I}^{*(i_1i_2)}_{(11)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(20)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}\bar{L}\bar{L}B_{i_2}\left(\frac{1}{2}\hat{I}^{*(i_1i_2)}_{(02)\tau_{p+1},\tau_p}+\frac{1}{2}\hat{I}^{*(i_1i_2)}_{(20)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2)}_{(11)\tau_{p+1},\tau_p}\right)-$$

$$-\bar{L}G_0^{(i_1)}G_0^{(i_2)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1i_2)}_{(10)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1i_2)}_{(11)\tau_{p+1},\tau_p}\right)\Bigg]+$$

$$+\sum_{i_1,i_2,i_3,i_4=1}^{m}\Bigg[G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}\bar{\mathbf{a}}\left(\Delta\hat{I}^{*(i_1i_2i_3i_4)}_{(0000)\tau_{p+1},\tau_p}+\hat{I}^{*(i_1i_2i_3i_4)}_{(0001)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}\bar{L}G_0^{(i_3)}B_{i_4}\left(\hat{I}^{*(i_1i_2i_3i_4)}_{(0100)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3i_4)}_{(0010)\tau_{p+1},\tau_p}\right)-$$

$$-\bar{L}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\hat{I}^{*(i_1i_2i_3i_4)}_{(1000)\tau_{p+1},\tau_p}+$$

$$+G_0^{(i_1)}\bar{L}G_0^{(i_2)}G_0^{(i_3)}B_{i_4}\left(\hat{I}^{*(i_1i_2i_3i_4)}_{(1000)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3i_4)}_{(0100)\tau_{p+1},\tau_p}\right)+$$

$$+G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}\bar{L}B_{i_4}\left(\hat{I}^{*(i_1i_2i_3i_4)}_{(0010)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1i_2i_3i_4)}_{(0001)\tau_{p+1},\tau_p}\right)\Bigg]+$$

$$+\sum_{i_1,i_2,i_3,i_4,i_5,i_6=1}^{m}G_0^{(i_1)}G_0^{(i_2)}G_0^{(i_3)}G_0^{(i_4)}G_0^{(i_5)}B_{i_6}\hat{I}^{*(i_1i_2i_3i_4i_5i_6)}_{(000000)\tau_{p+1},\tau_p}.$$

Under the suitable conditions [2] the numerical schemes (24)–(28) have strong orders 1.0, 1.5, 2.0, 2.5, and 3.0 of convergence correspondingly. Among these conditions we consider only the condition for approximations of iterated Stratonovich stochastic integrals from (24)–(28) [2] (also see [42])

$$\mathsf{M}\left\{\left(I^{*(i_1\ldots i_k)}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}-\hat{I}^{*(i_1\ldots i_k)}_{(l_1\ldots l_k)\tau_{p+1},\tau_p}\right)^2\right\}\le C\Delta^{r+1},\qquad(29)$$

where constant C is independent of Δ and $r/2$ are the strong convergence orders for the numerical schemes (24)–(28), i.e. $r/2=1.0,1.5,2.0,2.5$, and 3.0.

Note that the numerical schemes (24)–(28) are unrealizable in practice without procedures for the numerical simulation of iterated Stratonovich stochastic

integrals from (22). The next section is devoted to the effective method of the mean-square approximation of iterated Itô and Stratonovich stochastic integrals of arbitrary multiplicity k ($k \in \mathbb{N}$).

2.3 Method of Expansion and Approximation of Iterated Itô and Stratonovich Stochastic Integrals Based on Generalized Multiple Fourier Series

Let us consider the effective approach to expansion of iterated Itô stochastic integrals [42] (2006) (also see [26], [44]-[46], [58], [61], [67], [68]). This method is reffered to as the method of generalized multiple Fourier series.

Suppose that every $\psi_l(\tau)$ ($l = 1, \ldots, k$) is a continuous nonrandom function on $[t, T]$. Define the following function on the hypercube $[t, T]^k$

$$K(t_1, \ldots, t_k) = \begin{cases} \psi_1(t_1) \ldots \psi_k(t_k) & \text{for } \ t_1 < \ldots < t_k \\ & \\ 0 & \text{otherwise} \end{cases}, \quad t_1, \ldots, t_k \in [t, T], \tag{30}$$

where $k \geq 2$ and $K(t_1) \equiv \psi_1(t_1)$ for $t_1 \in [t, T]$.

Suppose that $\{\phi_j(x)\}_{j=0}^{\infty}$ is a complete orthonormal system of functions in the space $L_2([t, T])$.

The function $K(t_1, \ldots, t_k)$ is piecewise continuous in the hypercube $[t, T]^k$. At this situation it is well known that the generalized multiple Fourier series of $K(t_1, \ldots, t_k) \in L_2([t, T]^k)$ is converging to $K(t_1, \ldots, t_k)$ in the hypercube $[t, T]^k$ in the mean-square sense, i.e.

$$\lim_{p_1, \ldots, p_k \to \infty} \left\| K(t_1, \ldots, t_k) - \sum_{j_1=0}^{p_1} \ldots \sum_{j_k=0}^{p_k} C_{j_k \ldots j_1} \prod_{l=1}^{k} \phi_{j_l}(t_l) \right\|_{L_2([t,T]^k)} = 0,$$

where

$$C_{j_k \ldots j_1} = \int\limits_{[t,T]^k} K(t_1, \ldots, t_k) \prod_{l=1}^{k} \phi_{j_l}(t_l) dt_1 \ldots dt_k \tag{31}$$

is the Fourier coefficient and

$$\|f\|_{L_2([t,T]^k)} = \left(\int\limits_{[t,T]^k} f^2(t_1, \ldots, t_k) dt_1 \ldots dt_k \right)^{1/2}.$$

Consider the partition $\{\tau_j\}_{j=0}^N$ of the interval $[t,T]$ such that

$$t=\tau_0<\ldots<\tau_N=T,\quad \Delta_N=\max_{0\le j\le N-1}\Delta\tau_j\to 0\ \text{ if }\ N\to\infty,\quad \Delta\tau_j=\tau_{j+1}-\tau_j. \tag{32}$$

Theorem 1 [42] (2006) (also see [26],[44]-[46], [58], [61], [67], [68]). *Suppose that every $\psi_l(\tau)$ $(l=1,\ldots,k)$ is a continuous nonrandom function on $[t,T]$ and $\{\phi_j(x)\}_{j=0}^{\infty}$ is a complete orthonormal system of continuous functions in the space $L_2([t,T])$. Then*

$$J[\psi^{(k)}]_{T,t}\ =\ \operatorname*{l.i.m.}_{p_1,\ldots,p_k\to\infty}\ \sum_{j_1=0}^{p_1}\ldots\sum_{j_k=0}^{p_k}C_{j_k\ldots j_1}\left(\prod_{l=1}^{k}\zeta_{j_l}^{(i_l)}\ -\right.$$

$$\left.-\ \operatorname*{l.i.m.}_{N\to\infty}\ \sum_{(l_1,\ldots,l_k)\in \mathrm{G}_k}\phi_{j_1}(\tau_{l_1})\Delta\mathbf{w}_{\tau_{l_1}}^{(i_1)}\ldots\phi_{j_k}(\tau_{l_k})\Delta\mathbf{w}_{\tau_{l_k}}^{(i_k)}\right), \tag{33}$$

where $J[\psi^{(k)}]_{T,t}$ is defined by (2),

$$\mathrm{G}_k=\mathrm{H}_k\backslash\mathrm{L}_k,\quad \mathrm{H}_k=\{(l_1,\ldots,l_k):\ l_1,\ldots,l_k=0,1,\ldots,N-1\},$$

$$\mathrm{L}_k=\{(l_1,\ldots,l_k):\ l_1,\ldots,l_k=0,1,\ldots,N-1;\ l_g\ne l_r\ (g\ne r);\ g,r=1,\ldots,k\},$$

l.i.m. *is a limit in the mean-square sense,* $i_1,\ldots,i_k=0,1,\ldots,m$,

$$\zeta_j^{(i)}=\int_t^T\phi_j(s)d\mathbf{w}_s^{(i)} \tag{34}$$

are independent standard Gaussian random variables for various i or j (in the case when $i\ne 0$), $C_{j_k\ldots j_1}$ is the Fourier coefficient (31), $\Delta\mathbf{w}_{\tau_j}^{(i)}=\mathbf{w}_{\tau_{j+1}}^{(i)}-\mathbf{w}_{\tau_j}^{(i)}$ $(i=0,1,\ldots,m)$, $\{\tau_j\}_{j=0}^N$ *is a partition of the interval $[t,T]$, which satisfies the condition* (32).

Note that a number of modifications and generalizations of Theorem 1 can be found in [26], [67].

Consider transformed particular cases of (33) for $k=1,\ldots,6$ [26], [58], [61], [67], [68]

$$J[\psi^{(1)}]_{T,t}=\operatorname*{l.i.m.}_{p_1\to\infty}\ \sum_{j_1=0}^{p_1}C_{j_1}\zeta_{j_1}^{(i_1)}, \tag{35}$$

$$J[\psi^{(2)}]_{T,t} = \operatorname*{l.i.m.}_{p_1,p_2\to\infty} \sum_{j_1=0}^{p_1}\sum_{j_2=0}^{p_2} C_{j_2j_1}\left(\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)} - \mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\right), \tag{36}$$

$$J[\psi^{(3)}]_{T,t} = \operatorname*{l.i.m.}_{p_1,\ldots,p_3\to\infty} \sum_{j_1=0}^{p_1}\sum_{j_2=0}^{p_2}\sum_{j_3=0}^{p_3} C_{j_3j_2j_1}\left(\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)} - \right.$$

$$\left. -\mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)} - \mathbf{1}_{\{i_2=i_3\ne 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)} - \mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\right), \tag{37}$$

$$J[\psi^{(4)}]_{T,t} = \operatorname*{l.i.m.}_{p_1,\ldots,p_4\to\infty} \sum_{j_1=0}^{p_1}\ldots\sum_{j_4=0}^{p_4} C_{j_4\ldots j_1}\left(\prod_{l=1}^{4}\zeta_{j_l}^{(i_l)} - \right.$$

$$-\mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)} - \mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)} -$$

$$-\mathbf{1}_{\{i_1=i_4\ne 0\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)} - \mathbf{1}_{\{i_2=i_3\ne 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)} -$$

$$-\mathbf{1}_{\{i_2=i_4\ne 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)} - \mathbf{1}_{\{i_3=i_4\ne 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)} +$$

$$+\mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\ne 0\}}\mathbf{1}_{\{j_3=j_4\}} + \mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\ne 0\}}\mathbf{1}_{\{j_2=j_4\}} +$$

$$\left. + \mathbf{1}_{\{i_1=i_4\ne 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\ne 0\}}\mathbf{1}_{\{j_2=j_3\}}\right), \tag{38}$$

$$J[\psi^{(5)}]_{T,t} = \operatorname*{l.i.m.}_{p_1,\ldots,p_5\to\infty} \sum_{j_1=0}^{p_1}\ldots\sum_{j_5=0}^{p_5} C_{j_5\ldots j_1}\left(\prod_{l=1}^{5}\zeta_{j_l}^{(i_l)} - \right.$$

$$-\mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)} - \mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)} -$$

$$-\mathbf{1}_{\{i_1=i_4\ne 0\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)} - \mathbf{1}_{\{i_1=i_5\ne 0\}}\mathbf{1}_{\{j_1=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)} -$$

$$-\mathbf{1}_{\{i_2=i_3\ne 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)} - \mathbf{1}_{\{i_2=i_4\ne 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)} -$$

$$-\mathbf{1}_{\{i_2=i_5\ne 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)} - \mathbf{1}_{\{i_3=i_4\ne 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)} -$$

$$-\mathbf{1}_{\{i_3=i_5\ne 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)} - \mathbf{1}_{\{i_4=i_5\ne 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)} +$$

$$+\mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\ne 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_5}^{(i_5)} + \mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_5\ne 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_4}^{(i_4)} +$$

$$+\mathbf{1}_{\{i_1=i_2\ne 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_4=i_5\ne 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_3}^{(i_3)} + \mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\ne 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_5}^{(i_5)} +$$

$$+\mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_5\ne 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_4}^{(i_4)} + \mathbf{1}_{\{i_1=i_3\ne 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_4=i_5\ne 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_2}^{(i_2)} +$$

$$+\mathbf{1}_{\{i_1=i_4\ne 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\ne 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_5}^{(i_5)} + \mathbf{1}_{\{i_1=i_4\ne 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_5\ne 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_3}^{(i_3)} +$$

$$+\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_2}^{(i_2)}+\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_2}^{(i_2)}+$$
$$+\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}+\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}+$$
$$+\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\Bigg), \tag{39}$$

$$J[\psi^{(6)}]_{T,t}=\operatorname*{l.i.m.}_{p_1,\ldots,p_6\to\infty}\sum_{j_1=0}^{p_1}\ldots\sum_{j_6=0}^{p_6}C_{j_6\ldots j_1}\Bigg(\prod_{l=1}^{6}\zeta_{j_l}^{(i_l)}-$$
$$-\mathbf{1}_{\{i_1=i_6\neq 0\}}\mathbf{1}_{\{j_1=j_6\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}-\mathbf{1}_{\{i_2=i_6\neq 0\}}\mathbf{1}_{\{j_2=j_6\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}-$$
$$-\mathbf{1}_{\{i_3=i_6\neq 0\}}\mathbf{1}_{\{j_3=j_6\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}-\mathbf{1}_{\{i_4=i_6\neq 0\}}\mathbf{1}_{\{j_4=j_6\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}-$$
$$-\mathbf{1}_{\{i_5=i_6\neq 0\}}\mathbf{1}_{\{j_5=j_6\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}-\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-$$
$$-\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-$$
$$-\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}-\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-$$
$$-\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}-$$
$$-\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}-$$
$$-\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_6}^{(i_6)}+$$
$$+\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+$$
$$+\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}+$$
$$+\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}+$$
$$+\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}+$$

$$
\begin{aligned}
&+\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}+\\
&+\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_5}^{(i_5)}+\\
&+\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+\\
&+\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}+\\
&+\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}+\\
&+\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}+\\
&+\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}+\\
&+\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}+\\
&+\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+\\
&+\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}+\\
&+\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}-\\
&-\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}-\\
&-\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_1\neq 0\}}\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}-\\
&-\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_2\neq 0\}}\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}-\\
&-\mathbf{1}_{\{i_6=i_3\neq 0\}}\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}-\\
&-\mathbf{1}_{\{i_3=i_6\neq 0\}}\mathbf{1}_{\{j_3=j_6\}}\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_4=i_5\neq 0\}}\mathbf{1}_{\{j_4=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_1=i_5\neq 0\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}-\\
&-\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_5\neq 0\}}\mathbf{1}_{\{j_2=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_4\neq 0\}}\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_5\neq 0\}}\mathbf{1}_{\{j_3=j_5\}}-\\
&-\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_1=i_4\neq 0\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\neq 0\}}\mathbf{1}_{\{j_2=j_3\}}-\\
&-\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_1=i_2\neq 0\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\neq 0\}}\mathbf{1}_{\{j_3=j_4\}}-\\
&-\mathbf{1}_{\{i_6=i_5\neq 0\}}\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_1=i_3\neq 0\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\neq 0\}}\mathbf{1}_{\{j_2=j_4\}}\Bigg),
\end{aligned}
\tag{40}
$$

where $\mathbf{1}_A$ is the indicator of the set A.

A detailed discussion of advantages of the method based on Theorem 1 over the approximation methods from works [2], [3], [7], [8], [27]-[35], [37]-[39] can be found in [26] (Section 1.1.10) or in [67].

As it turned out, Theorem 1 can be adapted for the iterated Stratonovich stochastic integrals (3) of multiplicities 1 to 5 [26], [67]-[69] (also see bibliography therein). Let as collect some of these results in the following theorem.

Theorem 2 [26], [67]-[69]. *Suppose that $\{\phi_j(x)\}_{j=0}^{\infty}$ is a complete orthonormal system of Legendre polynomials or trigonometric functions in the space $L_2([t,T])$. At the same time $\psi_2(s)$ is a continuously differentiable function on $[t,T]$ and $\psi_1(s)$, $\psi_3(s)$ are twice continuously differentiable functions on $[t,T]$. Then*

$$J^*[\psi^{(2)}]_{T,t} = \operatorname*{l.i.m.}_{p_1,p_2\to\infty} \sum_{j_1=0}^{p_1}\sum_{j_2=0}^{p_2} C_{j_2j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)} \quad (i_1,i_2=1,\ldots,m), \tag{41}$$

$$J^*[\psi^{(3)}]_{T,t} = \operatorname*{l.i.m.}_{p_1,p_2,p_3\to\infty} \sum_{j_1=0}^{p_1}\sum_{j_2=0}^{p_2}\sum_{j_3=0}^{p_3} C_{j_3j_2j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)} \quad (i_1,i_2,i_3=0,1,\ldots,m), \tag{42}$$

$$J^*[\psi^{(3)}]_{T,t} = \operatorname*{l.i.m.}_{p\to\infty} \sum_{j_1,j_2,j_3=0}^{p} C_{j_3j_2j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)} \quad (i_1,i_2,i_3=1,\ldots,m), \tag{43}$$

$$J^*[\psi^{(4)}]_{T,t} = \operatorname*{l.i.m.}_{p\to\infty} \sum_{j_1,\ldots,j_4=0}^{p} C_{j_4j_3j_2j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)} \quad (i_1,\ldots,i_4=0,1,\ldots,m), \tag{44}$$

$$J^*[\psi^{(5)}]_{T,t} = \operatorname*{l.i.m.}_{p\to\infty} \sum_{j_1,\ldots,j_5=0}^{p} C_{j_5j_4j_3j_2j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)} \quad (i_1,\ldots,i_5=0,1,\ldots,m), \tag{45}$$

where $J^*[\psi^{(k)}]_{T,t}$ is defined by (3)*, and $\psi_l(s)\equiv 1$ $(l=1,\ldots,5)$ in* (42)*,* (44)*,* (45)*; another notations are the same as in Theorem* 1.

Consider the following Hypothesis on expansion of the iterated Stratonovich stochastic integrals (3) of arbitrary multiplicity k ($k\in\mathbb{N}$).

Hypothesis 1 [26], [67], [68]. *Assume that $\{\phi_j(x)\}_{j=0}^{\infty}$ is a complete orthonormal system of Legendre polynomials or trigonometric functions in the*

space $L_2([t,T])$. *Moreover, every* $\psi_l(\tau)$ $(l=1,\ldots,k)$ *is an enough smooth non-random function on* $[t,T]$. *Then, for the iterated Stratonovich stochastic integral* (3) *of multiplicity* k *the following expansion*

$$J^*[\psi^{(k)}]_{T,t} = \operatorname*{l.i.m.}_{p\to\infty} \sum_{j_1,\ldots j_k=0}^{p} C_{j_k\ldots j_1} \prod_{l=1}^{k} \zeta_{j_l}^{(i_l)} \tag{46}$$

that converges in the mean-square sense is valid, where notations are the same as in Theorem 1.

Hypothesis 1 allows to approximate the iterated Stratonovich stochastic integral $J^*[\psi^{(k)}]_{T,t}$ by the sum

$$J^*[\psi^{(k)}]_{T,t}^p = \sum_{j_1,\ldots j_k=0}^{p} C_{j_k\ldots j_1} \prod_{l=1}^{k} \zeta_{j_l}^{(i_l)}, \tag{47}$$

where

$$\lim_{p\to\infty} \mathsf{M}\left\{\left(J^*[\psi^{(k)}]_{T,t} - J^*[\psi^{(k)}]_{T,t}^p\right)^2\right\} = 0.$$

Assume that $J[\psi^{(k)}]_{T,t}^p$ is the approximation of (2), which is the expression in (33) before passing to the limit for the case $p_1 = \ldots = p_k = p$, i.e.

$$J[\psi^{(k)}]_{T,t}^p = \sum_{j_1,\ldots,j_k=0}^{p} C_{j_k\ldots j_1}\left(\prod_{l=1}^{k}\zeta_{j_l}^{(i_l)} - \right.$$

$$\left. - \operatorname*{l.i.m.}_{N\to\infty} \sum_{(l_1,\ldots,l_k)\in \mathrm{G}_k} \phi_{j_1}(\tau_{l_1})\Delta\mathbf{w}_{\tau_{l_1}}^{(i_1)}\ldots\phi_{j_k}(\tau_{l_k})\Delta\mathbf{w}_{\tau_{l_k}}^{(i_k)}\right). \tag{48}$$

Let us denote

$$\mathsf{M}\left\{\left(J[\psi^{(k)}]_{T,t} - J[\psi^{(k)}]_{T,t}^p\right)^2\right\} \stackrel{\text{def}}{=} E_k^p,$$

$$\|K\|^2_{L_2([t,T]^k)} = \int\limits_{[t,T]^k} K^2(t_1,\ldots,t_k)dt_1\ldots dt_k \stackrel{\text{def}}{=} I_k.$$

For the futher consideration, we need the following useful estimate [26], [67]

$$E_k^p \le k!\left(I_k - \sum_{j_1,\ldots,j_k=0}^{p} C_{j_k\ldots j_1}^2\right), \tag{49}$$

where $i_1,\ldots,i_k=1,\ldots,m$ for $T-t\in(0,\infty)$ and $i_1,\ldots,i_k=0,1,\ldots,m$ for $T-t\in(0,1)$; another notations are the same as in Theorem 1.

The value E_k^p can be calculated exactly.

Theorem 3 [26], [67], [70]. *Suppose that the conditions of Theorem* 1 *are satisfied. Then*

$$E_k^p=I_k-\sum_{j_1,\ldots,j_k=0}^{p}C_{j_k\ldots j_1}\times$$

$$\times\,\mathsf{M}\left\{J[\psi^{(k)}]_{T,t}\sum_{(j_1,\ldots,j_k)}\int\limits_t^T\phi_{j_k}(t_k)\ldots\int\limits_t^{t_2}\phi_{j_1}(t_1)d\mathbf{w}_{t_1}^{(i_1)}\ldots d\mathbf{w}_{t_k}^{(i_k)}\right\},\qquad(50)$$

where $i_1,\ldots,i_k=1,\ldots,m$; *expression*

$$\sum_{(j_1,\ldots,j_k)}$$

means the sum with respect to all possible permutations $(j_1,\ldots,j_k)$. *At the same time if* j_r *swapped with* j_q *in the permutation* $(j_1,\ldots,j_k)$, *then* i_r *swapped with* i_q *in the permutation* $(i_1,\ldots,i_k)$; *another notations are the same as in Theorem* 1.

Note that

$$\mathsf{M}\left\{J[\psi^{(k)}]_{T,t}\int\limits_t^T\phi_{j_k}(t_k)\ldots\int\limits_t^{t_2}\phi_{j_1}(t_1)d\mathbf{w}_{t_1}^{(i_1)}\ldots d\mathbf{w}_{t_k}^{(i_k)}\right\}=C_{j_k\ldots j_1}.$$

Then from Theorem 3 we obtain

$$E_k^p=I_k-\sum_{j_1,\ldots,j_k=0}^{p}C_{j_k\ldots j_1}^2\quad(i_1,\ldots,i_k\text{ are pairwise different}),\qquad(51)$$

$$E_k^p=I_k-\sum_{j_1,\ldots,j_k=0}^{p}C_{j_k\ldots j_1}\left(\sum_{(j_1,\ldots,j_k)}C_{j_k\ldots j_1}\right)\quad(i_1=\ldots=i_k).\qquad(52)$$

Consider some examples of the application of Theorem 3 ($i_1,\ldots,i_5=1,\ldots,m$):

$$E_2^p=I_2-\sum_{j_1,j_2=0}^{p}C_{j_2j_1}^2-\sum_{j_1,j_2=0}^{p}C_{j_2j_1}C_{j_1j_2}\quad(i_1=i_2),\qquad(53)$$

$$E_3^p = I_3 - \sum_{j_3,j_2,j_1=0}^{p} C_{j_3j_2j_1}^2 - \sum_{j_3,j_2,j_1=0}^{p} C_{j_3j_1j_2}C_{j_3j_2j_1} \quad (i_1 = i_2 \neq i_3), \tag{54}$$

$$E_3^p = I_3 - \sum_{j_3,j_2,j_1=0}^{p} C_{j_3j_2j_1}^2 - \sum_{j_3,j_2,j_1=0}^{p} C_{j_2j_3j_1}C_{j_3j_2j_1} \quad (i_1 \neq i_2 = i_3), \tag{55}$$

$$E_3^p = I_3 - \sum_{j_3,j_2,j_1=0}^{p} C_{j_3j_2j_1}^2 - \sum_{j_3,j_2,j_1=0}^{p} C_{j_3j_2j_1}C_{j_1j_2j_3} \quad (i_1 = i_3 \neq i_2), \tag{56}$$

$$E_4^p = I_4 - \sum_{j_1,\dots,j_4=0}^{p} C_{j_4\dots j_1} \left(\sum_{(j_1,j_2)} C_{j_4\dots j_1} \right) \quad (i_1 = i_2 \neq i_3, i_4;\ i_3 \neq i_4), \tag{57}$$

$$E_4^p = I_4 - \sum_{j_1,\dots,j_4=0}^{p} C_{j_4\dots j_1} \left(\sum_{(j_1,j_3)} C_{j_4\dots j_1} \right) \quad (i_1 = i_3 \neq i_2, i_4;\ i_2 \neq i_4), \tag{58}$$

$$E_4^p = I_4 - \sum_{j_1,\dots,j_4=0}^{p} C_{j_4\dots j_1} \left(\sum_{(j_2,j_3)} C_{j_4\dots j_1} \right) \quad (i_2 = i_3 \neq i_1, i_4;\ i_1 \neq i_4), \tag{59}$$

$$E_4^p = I_4 - \sum_{j_1,\dots,j_4=0}^{p} C_{j_4\dots j_1} \left(\sum_{(j_1,j_4)} C_{j_4\dots j_1} \right) \quad (i_1 = i_4 \neq i_2, i_3;\ i_2 \neq i_3), \tag{60}$$

$$E_4^p = I_4 - \sum_{j_1,\dots,j_4=0}^{p} C_{j_4\dots j_1} \left(\sum_{(j_1,j_4)} \left(\sum_{(j_2,j_3)} C_{j_4\dots j_1} \right) \right) \quad (i_1 = i_4 \neq i_2 = i_3), \tag{61}$$

$$E_4^p = I_4 - \sum_{j_1,\dots,j_4=0}^{p} C_{j_4\dots j_1} \left(\sum_{(j_1,j_2,j_3)} C_{j_4\dots j_1} \right) \quad (i_1 = i_2 = i_3 \neq i_4), \tag{62}$$

$$E_5^p = I_5 - \sum_{j_1,\dots,j_5=0}^{p} C_{j_5\dots j_1} \left(\sum_{(j_1,j_2)} C_{j_5\dots j_1} \right), \tag{63}$$

where $i_1 = i_2 \neq i_3, i_4, i_5$ and i_3, i_4, i_5 are pairwise different,

$$E_5^p = I_5 - \sum_{j_1,\dots,j_5=0}^{p} C_{j_5\dots j_1} \left(\sum_{(j_2,j_3)} C_{j_5\dots j_1} \right), \tag{64}$$

where $i_2 = i_3 \neq i_1, i_4, i_5$ and i_1, i_4, i_5 are pairwise different,

$$E_5^p = I_5 - \sum_{j_1,\ldots,j_5=0}^{p} C_{j_5\ldots j_1} \left(\sum_{(j_4,j_5)} C_{j_5\ldots j_1} \right), \tag{65}$$

where $i_4 = i_5 \neq i_1, i_2, i_3$ and i_1, i_2, i_3 are pairwise different,

$$E_5^p = I_5 - \sum_{j_1,\ldots,j_5=0}^{p} C_{j_5\ldots j_1} \left(\sum_{(j_2,j_4)} \left(\sum_{(j_3,j_5)} C_{j_5\ldots j_1} \right) \right) \quad (i_1 \neq i_2 = i_4 \neq i_3 = i_5 \neq i_1). \tag{66}$$

2.4 Approximations of Iterated Itô Stochastic Integrals from the Numerical Schemes (11)–(16) Using Legendre Polynomials

This section is devoted to approximation of the iterated Itô stochastic integrals (6) of multiplicities 1 to 6 based on Theorem 1. At that we will use multiple Fourier–Legendre series for approximation of the mentioned stochastic integrals.

The numerical schemes (11)–(16) contain the following set (see (6)) of iterated Itô stochastic integrals

$$I_{(0)T,t}^{(i_1)}, \quad I_{(1)T,t}^{(i_1)}, \quad I_{(2)T,t}^{(i_1)}, \quad I_{(00)T,t}^{(i_1i_2)}, \quad I_{(10)T,t}^{(i_1i_2)}, \quad I_{(01)T,t}^{(i_1i_2)}, \quad I_{(000)T,t}^{(i_1i_2i_3)}, \quad I_{(0000)T,t}^{(i_1i_2i_3i_4)}, \tag{67}$$

$$I_{(00000)T,t}^{(i_1i_2i_3i_4i_5)}, \quad I_{(02)T,t}^{(i_1i_2)}, \quad I_{(20)T,t}^{(i_1i_2)}, \quad I_{(11)T,t}^{(i_1i_2)}, \quad I_{(100)T,t}^{(i_1i_2i_3)}, \quad I_{(010)T,t}^{(i_1i_2i_3)}, \quad I_{(001)T,t}^{(i_1i_2i_3)}, \tag{68}$$

$$I_{(0001)T,t}^{(i_1i_2i_3i_4)}, \quad I_{(0010)T,t}^{(i_1i_2i_3i_4)}, \quad I_{(0100)T,t}^{(i_1i_2i_3i_4)}, \quad I_{(1000)T,t}^{(i_1i_2i_3i_4)}, \quad I_{(000000)T,t}^{(i_1i_2i_3i_4i_5i_6)}. \tag{69}$$

Let us consider the complete orthonormal system of Legendre polynomials in the space $L_2([t,T])$

$$\phi_j(x) = \sqrt{\frac{2j+1}{T-t}} P_j\left(\left(x - \frac{T+t}{2} \right) \frac{2}{T-t} \right), \quad j = 0, 1, 2, \ldots, \tag{70}$$

where $P_j(x)$ is the Legendre polynomial

$$P_j(x) = \frac{1}{2^j j!} \frac{d^j}{dx^j} \left(x^2 - 1 \right)^j.$$

Using Theorem 1 and well known properties of the Legendre polynomials, we obtain the following formulas for numerical modeling of the stochastic integrals (67)–(69) [26], [42]-[46], [58], [61], [67], [68], [70]-[72]

$$I_{(0)T,t}^{(i_1)} = \sqrt{T-t}\,\zeta_0^{(i_1)}, \tag{71}$$

$$I^{(i_1)}_{(1)T,t} = -\frac{(T-t)^{3/2}}{2}\left(\zeta^{(i_1)}_0 + \frac{1}{\sqrt{3}}\zeta^{(i_1)}_1\right), \tag{72}$$

$$I^{(i_1)}_{(2)T,t} = \frac{(T-t)^{5/2}}{3}\left(\zeta^{(i_1)}_0 + \frac{\sqrt{3}}{2}\zeta^{(i_1)}_1 + \frac{1}{2\sqrt{5}}\zeta^{(i_1)}_2\right), \tag{73}$$

$$I^{(i_1i_2)q}_{(00)T,t} = \frac{T-t}{2}\left(\zeta^{(i_1)}_0\zeta^{(i_2)}_0 + \sum_{i=1}^{q}\frac{1}{\sqrt{4i^2-1}}\left(\zeta^{(i_1)}_{i-1}\zeta^{(i_2)}_i - \zeta^{(i_1)}_i\zeta^{(i_2)}_{i-1}\right) - \mathbf{1}_{\{i_1=i_2\}}\right), \tag{74}$$

$$I^{(i_1i_2i_3)q_1}_{(000)T,t} = \sum_{j_1,j_2,j_3=0}^{q_1} C^{000}_{j_3j_2j_1}\left(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3} - \right.$$

$$\left. -\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\right), \tag{75}$$

$$I^{(i_1i_2)q_2}_{(10)T,t} = \sum_{j_1,j_2=0}^{q_2} C^{10}_{j_2j_1}\left(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} - \mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\right), \tag{76}$$

$$I^{(i_1i_2)q_2}_{(01)T,t} = \sum_{j_1,j_2=0}^{q_2} C^{01}_{j_2j_1}\left(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} - \mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\right), \tag{77}$$

$$I^{(i_1i_2i_3i_4)q_3}_{(0000)T,t} = \sum_{j_1,j_2,j_3,j_4=0}^{q_3} C^{0000}_{j_4j_3j_2j_1}\left(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \right.$$

$$-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4} -$$

$$-\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_4)}_{j_4} -$$

$$-\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} +$$

$$+\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}} + \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}} +$$

$$\left. + \mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\right), \tag{78}$$

$$I^{(i_1i_2i_3i_4i_5)q_4}_{(00000)T,t} = \sum_{j_1,j_2,j_3,j_4,j_5=0}^{q_4} C^{00000}_{j_5j_4j_3j_2j_1}\left(\prod_{l=1}^{5}\zeta^{(i_l)}_{j_l} - \right.$$

$$-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5} -$$

$$-\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}-\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_1=j_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}-$$
$$-\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}-\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}-$$
$$-\mathbf{1}_{\{i_2=i_5\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}-\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}-$$
$$-\mathbf{1}_{\{i_3=i_5\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}-\mathbf{1}_{\{i_4=i_5\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+$$
$$+\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_5\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_4=i_5\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_5}^{(i_5)}+$$
$$+\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_5\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_4}^{(i_4)}+\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_4=i_5\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_2}^{(i_2)}+$$
$$+\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_5\}}\mathbf{1}_{\{j_2=j_5\}}\zeta_{j_3}^{(i_3)}+$$
$$+\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_3=i_5\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_2}^{(i_2)}+\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_4}^{(i_4)}+$$
$$+\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_2}^{(i_2)}+$$
$$+\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_4=i_5\}}\mathbf{1}_{\{j_4=j_5\}}\zeta_{j_1}^{(i_1)}+\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_3=i_5\}}\mathbf{1}_{\{j_3=j_5\}}\zeta_{j_1}^{(i_1)}+$$
$$+\mathbf{1}_{\{i_2=i_5\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta_{j_1}^{(i_1)}\Bigg), \tag{79}$$

$$I_{(20)T,t}^{(i_1i_2)q_5}=\sum_{j_1,j_2=0}^{q_5}C_{j_2j_1}^{20}\left(\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\right), \tag{80}$$

$$I_{(11)T,t}^{(i_1i_2)q_6}=\sum_{j_1,j_2=0}^{q_6}C_{j_2j_1}^{11}\left(\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\right), \tag{81}$$

$$I_{(02)T,t}^{(i_1i_2)q_7}=\sum_{j_1,j_2=0}^{q_7}C_{j_2j_1}^{02}\left(\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\right), \tag{82}$$

$$I_{(001)T,t}^{(i_1i_2i_3)q_8}=\sum_{j_1,j_2,j_3=0}^{q_8}C_{j_3j_2j_1}^{001}\Bigg(\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta_{j_3}^{(i_3)}-$$
$$-\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta_{j_1}^{(i_1)}-\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta_{j_2}^{(i_2)}\Bigg), \tag{83}$$

$$I^{(i_1i_2i_3)q_9}_{(010)T,t} = \sum_{j_1,j_2,j_3=0}^{q_9} C^{010}_{j_3j_2j_1}\Bigg(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3} - \\ -\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\Bigg), \tag{84}$$

$$I^{(i_1i_2i_3)q_{10}}_{(100)T,t} = \sum_{j_1,j_2,j_3=0}^{q_{10}} C^{100}_{j_3j_2j_1}\Bigg(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3} - \\ -\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\Bigg), \tag{85}$$

$$I^{(i_1i_2i_3i_4)q_{11}}_{(0001)T,t} = \sum_{j_1,j_2,j_3,j_4=0}^{q_{11}} C^{0001}_{j_4j_3j_2j_1}\Bigg(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \\ -\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4} - \\ -\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_4)}_{j_4} - \\ -\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} + \\ +\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}} + \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}} + \\ + \mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\Bigg), \tag{86}$$

$$I^{(i_1i_2i_3i_4)q_{12}}_{(0010)T,t} \sum_{j_1,j_2,j_3,j_4=0}^{q_{12}} C^{0010}_{j_4j_3j_2j_1}\Bigg(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \\ -\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4} - \\ -\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_4)}_{j_4} - \\ -\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} + \\ +\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}} + \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}} + \\ + \mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\Bigg), \tag{87}$$

$$I^{(i_1i_2i_3i_4)q_{13}}_{(0100)T,t} = \sum_{j_1,j_2,j_3,j_4=0}^{q_{13}} C^{0100}_{j_4j_3j_2j_1}\Bigg(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} -$$
$$-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4} -$$
$$-\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_4)}_{j_4} -$$
$$-\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} +$$
$$+\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}} + \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}} +$$
$$+\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\Bigg), \tag{88}$$

$$I^{(i_1i_2i_3i_4)q_{14}}_{(1000)T,t} = \sum_{j_1,j_2,j_3,j_4=0}^{q_{14}} C^{1000}_{j_4j_3j_2j_1}\Bigg(\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} -$$
$$-\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4} -$$
$$-\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_4)}_{j_4} -$$
$$-\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3} - \mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2} +$$
$$+\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_3=i_4\}}\mathbf{1}_{\{j_3=j_4\}} + \mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_2=j_4\}} +$$
$$+\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_2=j_3\}}\Bigg), \tag{89}$$

$$I^{(i_1i_2i_3i_4i_5i_6)q_{15}}_{(000000)T,t} = \sum_{j_1,j_2,j_3,j_4,j_5,j_6=0}^{q_{15}} C^{000000}_{j_6j_5j_4j_3j_2j_1}\Bigg(\prod_{l=1}^{6}\zeta^{(i_l)}_{j_l} -$$
$$-\mathbf{1}_{\{j_1=j_6\}}\mathbf{1}_{\{i_1=i_6\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5} - \mathbf{1}_{\{j_2=j_6\}}\mathbf{1}_{\{i_2=i_6\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5} -$$
$$-\mathbf{1}_{\{j_3=j_6\}}\mathbf{1}_{\{i_3=i_6\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5} - \mathbf{1}_{\{j_4=j_6\}}\mathbf{1}_{\{i_4=i_6\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_5)}_{j_5} -$$
$$-\mathbf{1}_{\{j_5=j_6\}}\mathbf{1}_{\{i_5=i_6\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4} - \mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5}\zeta^{(i_6)}_{j_6} -$$
$$-\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5}\zeta^{(i_6)}_{j_6} - \mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_5)}_{j_5}\zeta^{(i_6)}_{j_6} -$$
$$-\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\zeta^{(i_2)}_{j_2}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4}\zeta^{(i_6)}_{j_6} - \mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_4)}_{j_4}\zeta^{(i_5)}_{j_5}\zeta^{(i_6)}_{j_6} -$$
$$-\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3}\zeta^{(i_5)}_{j_5}\zeta^{(i_6)}_{j_6} - \mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\zeta^{(i_1)}_{j_1}\zeta^{(i_3)}_{j_3}\zeta^{(i_4)}_{j_4}\zeta^{(i_6)}_{j_6} -$$

$$-\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}-\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}-$$

$$-\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\zeta_{j_5}^{(i_5)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\zeta_{j_4}^{(i_4)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\zeta_{j_3}^{(i_3)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_6}^{(i_6)}+$$

$$+\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_6}^{(i_6)}+\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}+\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}+$$

$$+\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+$$

$$+\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}+$$

$$+\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}+\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}+$$

$$+\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}+\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}+\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}+$$

$$+\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_5}^{(i_5)}+$$

$$+\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_5}^{(i_5)}+\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}+$$

$$+\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\zeta_{j_1}^{(i_1)}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\zeta_{j_1}^{(i_1)}\zeta_{j_4}^{(i_4)}+$$

$$+\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}+\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\zeta_{j_2}^{(i_2)}\zeta_{j_4}^{(i_4)}+$$

$$+\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}-$$

$$
\begin{aligned}
&-\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}-\\
&-\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_1\}}\mathbf{1}_{\{i_6=i_1\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}-\\
&-\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_2\}}\mathbf{1}_{\{i_6=i_2\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}-\\
&-\mathbf{1}_{\{j_6=j_3\}}\mathbf{1}_{\{i_6=i_3\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}-\\
&-\mathbf{1}_{\{j_3=j_6\}}\mathbf{1}_{\{i_3=i_6\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_4=j_5\}}\mathbf{1}_{\{i_4=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_1=j_5\}}\mathbf{1}_{\{i_1=i_5\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}-\\
&-\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_2=j_5\}}\mathbf{1}_{\{i_2=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_4\}}\mathbf{1}_{\{i_6=i_4\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_3=j_5\}}\mathbf{1}_{\{i_3=i_5\}}-\\
&-\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_1=j_4\}}\mathbf{1}_{\{i_1=i_4\}}\mathbf{1}_{\{j_2=j_3\}}\mathbf{1}_{\{i_2=i_3\}}-\\
&-\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_1=j_2\}}\mathbf{1}_{\{i_1=i_2\}}\mathbf{1}_{\{j_3=j_4\}}\mathbf{1}_{\{i_3=i_4\}}-\\
&-\mathbf{1}_{\{j_6=j_5\}}\mathbf{1}_{\{i_6=i_5\}}\mathbf{1}_{\{j_1=j_3\}}\mathbf{1}_{\{i_1=i_3\}}\mathbf{1}_{\{j_2=j_4\}}\mathbf{1}_{\{i_2=i_4\}}\Bigg),
\end{aligned}
\tag{90}
$$

where $\mathbf{1}_A$ is the indicator of the set A and

$$
C^{000}_{j_3j_2j_1}=\frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)}}{8}(T-t)^{3/2}\bar{C}^{000}_{j_3j_2j_1},
\tag{91}
$$

$$
C^{01}_{j_2j_1}=\frac{\sqrt{(2j_1+1)(2j_2+1)}}{8}(T-t)^{2}\bar{C}^{01}_{j_2j_1},
\tag{92}
$$

$$
C^{10}_{j_2j_1}=\frac{\sqrt{(2j_1+1)(2j_2+1)}}{8}(T-t)^{2}\bar{C}^{10}_{j_2j_1},
\tag{93}
$$

$$
C^{0000}_{j_4j_3j_2j_1}=\frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)}}{16}(T-t)^{2}\bar{C}^{0000}_{j_4j_3j_2j_1},
\tag{94}
$$

$$
C^{02}_{j_2j_1}=\frac{\sqrt{(2j_1+1)(2j_2+1)}}{16}(T-t)^{3}\bar{C}^{02}_{j_2j_1},
\tag{95}
$$

$$C^{20}_{j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)}}{16}(T-t)^3\bar{C}^{20}_{j_2j_1}, \tag{96}$$

$$C^{11}_{j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)}}{16}(T-t)^3\bar{C}^{11}_{j_2j_1}, \tag{97}$$

$$C^{001}_{j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)}}{16}(T-t)^{5/2}\bar{C}^{001}_{j_3j_2j_1}, \tag{98}$$

$$C^{010}_{j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)}}{16}(T-t)^{5/2}\bar{C}^{010}_{j_3j_2j_1}, \tag{99}$$

$$C^{100}_{j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)}}{16}(T-t)^{5/2}\bar{C}^{100}_{j_3j_2j_1}, \tag{100}$$

$$C^{00000}_{j_5j_4j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)}}{32}(T-t)^{5/2}\bar{C}^{00000}_{j_5j_4j_3j_2j_1}, \tag{101}$$

$$C^{0001}_{j_4j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)}}{32}(T-t)^3\bar{C}^{0001}_{j_4j_3j_2j_1}, \tag{102}$$

$$C^{0010}_{j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)}}{32}(T-t)^3\bar{C}^{0010}_{j_4j_3j_2j_1}, \tag{103}$$

$$C^{0100}_{j_4j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)}}{32}(T-t)^3\bar{C}^{0100}_{j_3j_2j_1}, \tag{104}$$

$$C^{1000}_{j_4j_3j_2j_1} = \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)}}{32}(T-t)^3\bar{C}^{1000}_{j_4j_3j_2j_1}, \tag{105}$$

$$C^{000000}_{j_6j_5j_4j_3j_2j_1} =$$

$$= \frac{\sqrt{(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)(2j_6+1)}}{64}(T-t)^3\bar{C}^{000000}_{j_6j_5j_4j_3j_2j_1}, \tag{106}$$

where

$$\bar{C}^{000}_{j_3j_2j_1} = \int\limits_{-1}^{1} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydz, \tag{107}$$

$$\bar{C}^{01}_{j_2j_1} = -\int\limits_{-1}^{1} (1+y)P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdy, \tag{108}$$

$$\bar{C}^{10}_{j_2j_1} = -\int\limits_{-1}^{1} P_{j_2}(y) \int\limits_{-1}^{y} (1+x)P_{j_1}(x)dxdy, \tag{109}$$

$$\bar{C}^{0000}_{j_4j_3j_2j_1} = \int\limits_{-1}^{1} P_{j_4}(u) \int\limits_{-1}^{u} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydzdu, \tag{110}$$

$$\bar{C}^{02}_{j_2j_1} = \int\limits_{-1}^{1} P_{j_2}(y)(y+1)^2 \int\limits_{-1}^{y} P_{j_1}(x)dxdy, \tag{111}$$

$$\bar{C}^{20}_{j_2j_1} = \int\limits_{-1}^{1} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)(x+1)^2dxdy, \tag{112}$$

$$\bar{C}^{11}_{j_2j_1} = \int\limits_{-1}^{1} P_{j_2}(y)(y+1) \int\limits_{-1}^{y} P_{j_1}(x)(x+1)dxdy, \tag{113}$$

$$\bar{C}^{001}_{j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_3}(z)(z+1) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydz, \tag{114}$$

$$\bar{C}^{010}_{j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y)(y+1) \int\limits_{-1}^{y} P_{j_1}(x)dxdydz, \tag{115}$$

$$\bar{C}^{100}_{j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)(x+1)dxdydz, \tag{116}$$

$$\bar{C}^{00000}_{j_5j_4j_3j_2j_1} = \int\limits_{-1}^{1} P_{j_5}(v) \int\limits_{-1}^{v} P_{j_4}(u) \int\limits_{-1}^{u} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydzdudv, \tag{117}$$

$$\bar{C}^{1000}_{j_4j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_4}(u) \int\limits_{-1}^{u} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)(x+1)dxdydz, \tag{118}$$

$$\bar{C}^{0100}_{j_4j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_4}(u) \int\limits_{-1}^{u} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y)(y+1) \int\limits_{-1}^{y} P_{j_1}(x)dxdydz, \tag{119}$$

$$\bar{C}^{0010}_{j_4j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_4}(u) \int\limits_{-1}^{u} P_{j_3}(z)(z+1) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydz, \tag{120}$$

$$\bar{C}^{0001}_{j_4j_3j_2j_1} = -\int\limits_{-1}^{1} P_{j_4}(u)(u+1) \int\limits_{-1}^{u} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydz, \tag{121}$$

$$\bar{C}^{000000}_{j_6j_5j_4j_3j_2j_1} =$$

$$= \int\limits_{-1}^{1} P_{j_6}(w) \int\limits_{-1}^{w} P_{j_5}(v) \int\limits_{-1}^{v} P_{j_4}(u) \int\limits_{-1}^{u} P_{j_3}(z) \int\limits_{-1}^{z} P_{j_2}(y) \int\limits_{-1}^{y} P_{j_1}(x)dxdydzdudvdw; \tag{122}$$

another notations are the same as in Theorem 1.

2.5 Optimization of Approximations of Iterated Itô Stochastic Integrals from the Numerical Schemes (12)–(16)

This section is devoted to the optimization of approximations of iterated Itô stochastic integrals from the numerical schemes (12)–(16). More precisely, we discuss how to minimize the numbers $q, q_1, q_2, \ldots, q_{15}$ from Section 2.4.

Suppose that $\varepsilon > 0$ is the mean-square accuracy of approximation of the iterated Itô stochastic integrals (6), i.e.

$$E^{(i_1\ldots i_k)p}_{(l_1\ldots l_k)T,t} \stackrel{\mathsf{def}}{=} \mathsf{M}\left\{\left(I^{(i_1\ldots i_k)}_{(l_1\ldots l_k)T,t} - I^{(i_1\ldots i_k)p}_{(l_1\ldots l_k)T,t}\right)^2\right\} \le \varepsilon,$$

where $I^{(i_1\ldots i_k)p}_{(l_1\ldots l_k)T,t}$, $p \in \mathbb{N}$ is the approximation of the iterated Itô stochastic integral $I^{(i_1\ldots i_k)}_{(l_1\ldots l_k)T,t}$. Then from (74) and (49) we obtain the following conditions

for choosing the numbers $q, q_1, q_2, \ldots, q_{15}$ for approximations of the iterated Itô stochastic integrals (67)–(69) [26], [67]

$$E_{(00)T,t}^{(i_1 i_2)q} = \frac{(T-t)^2}{2}\left(\frac{1}{2} - \sum_{i=1}^{q}\frac{1}{4i^2-1}\right) \le \varepsilon, \tag{123}$$

$$E_{(000)T,t}^{(i_1 i_2 i_3)q_1} \le 6\left(\frac{(T-t)^3}{6} - \sum_{j_1,j_2,j_3=0}^{q_1}\left(C_{j_3 j_2 j_1}^{000}\right)^2\right) \le \varepsilon, \tag{124}$$

$$E_{(01)T,t}^{(i_1 i_2)q_2} \le 2\left(\frac{(T-t)^4}{4} - \sum_{j_1,j_2=0}^{q_2}\left(C_{j_2 j_1}^{01}\right)^2\right) \le \varepsilon, \tag{125}$$

$$E_{(10)T,t}^{(i_1 i_2)q_2} \le 2\left(\frac{(T-t)^4}{12} - \sum_{j_1,j_2=0}^{q_2}\left(C_{j_2 j_1}^{10}\right)^2\right) \le \varepsilon, \tag{126}$$

$$E_{(0000)T,t}^{(i_1 \ldots i_4)q_3} \le 24\left(\frac{(T-t)^4}{24} - \sum_{j_1,j_2,j_3,j_4=0}^{q_3}\left(C_{j_4 j_3 j_2 j_1}^{0000}\right)^2\right) \le \varepsilon, \tag{127}$$

$$E_{(00000)T,t}^{(i_1 \ldots i_5)q_4} \le 120\left(\frac{(T-t)^5}{120} - \sum_{j_1,j_2,j_3,j_4,j_5=0}^{q_4}\left(C_{j_5 j_4 j_3 j_2 j_1}^{00000}\right)^2\right) \le \varepsilon, \tag{128}$$

$$E_{(20)T,t}^{(i_1 i_2)q_5} \le 2\left(\frac{(T-t)^6}{30} - \sum_{j_2,j_1=0}^{q_5}\left(C_{j_2 j_1}^{20}\right)^2\right) \le \varepsilon, \tag{129}$$

$$E_{(11)T,t}^{(i_1 i_2)q_6} \le 2\left(\frac{(T-t)^6}{18} - \sum_{j_2,j_1=0}^{q_6}\left(C_{j_2 j_1}^{11}\right)^2\right) \le \varepsilon, \tag{130}$$

$$E_{(02)T,t}^{(i_1 i_2)q_7} \le 2\left(\frac{(T-t)^6}{6} - \sum_{j_2,j_1=0}^{q_7}\left(C_{j_2 j_1}^{02}\right)^2\right) \le \varepsilon, \tag{131}$$

$$E_{(001)T,t}^{(i_1 i_2 i_3)q_8} \le 6\left(\frac{(T-t)^5}{10} - \sum_{j_1,j_2,j_3=0}^{q_8}\left(C_{j_3 j_2 j_1}^{001}\right)^2\right) \le \varepsilon, \tag{132}$$

$$E_{(010)T,t}^{(i_1 i_2 i_3)q_9} \le 6\left(\frac{(T-t)^5}{20} - \sum_{j_1,j_2,j_3=0}^{q_9}\left(C_{j_3 j_2 j_1}^{010}\right)^2\right) \le \varepsilon, \tag{133}$$

$$E_{(100)T,t}^{(i_1 i_2 i_3)q_{10}} \le 6\left(\frac{(T-t)^5}{60} - \sum_{j_1,j_2,j_3=0}^{q_{10}}\left(C_{j_3 j_2 j_1}^{100}\right)^2\right) \le \varepsilon, \tag{134}$$

$$E_{(0001)T,t}^{(i_1\ldots i_4)q_{11}} \le 24\left(\frac{(T-t)^6}{36} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{11}} \left(C_{j_4j_3j_2j_1}^{0001}\right)^2\right) \le \varepsilon, \tag{135}$$

$$E_{(0010)T,t}^{(i_1\ldots i_4)q_{12}} \le 24\left(\frac{(T-t)^6}{60} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{12}} \left(C_{j_4j_3j_2j_1}^{0010}\right)^2\right) \le \varepsilon, \tag{136}$$

$$E_{(0100)T,t}^{(i_1\ldots i_4)q_{13}} \le 24\left(\frac{(T-t)^6}{120} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{13}} \left(C_{j_4j_3j_2j_1}^{0100}\right)^2\right) \le \varepsilon, \tag{137}$$

$$E_{(1000)T,t}^{(i_1\ldots i_4)q_{14}} \le 24\left(\frac{(T-t)^6}{360} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{14}} \left(C_{j_4j_3j_2j_1}^{1000}\right)^2\right) \le \varepsilon, \tag{138}$$

$$E_{(000000)T,t}^{(i_1\ldots i_6)q_{15}} \le 720\left(\frac{(T-t)^6}{720} - \sum_{j_1,j_2,j_3,j_4,j_5,j_6=0}^{q_{15}} \left(C_{j_6j_5j_4j_3j_2j_1}\right)^2\right) \le \varepsilon. \tag{139}$$

Taking into account (17) and (91)–(122), (123)–(139), we obtain the following conditions for choosing the numbers $q, q_1, q_2, \ldots, q_{15}$ for the numerical schemes (12)–(16) (constant C is independent of $T-t$ (see below)).

Milstein scheme (12)

$$\frac{1}{2}\left(\frac{1}{2} - \sum_{i=1}^{q} \frac{1}{4i^2-1}\right) \le C(T-t).$$

Strong Taylor–Itô scheme (13) with convergence order 1.5

$$\frac{1}{2}\left(\frac{1}{2} - \sum_{i=1}^{q} \frac{1}{4i^2-1}\right) \le C(T-t)^2,$$

$$6\left(\frac{1}{6} - \frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1} (2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{000}\right)^2\right) \le C(T-t). \tag{140}$$

Strong Taylor–Itô scheme (14) with convergence order 2.0

$$\frac{1}{2}\left(\frac{1}{2} - \sum_{i=1}^{q} \frac{1}{4i^2-1}\right) \le C(T-t)^3,$$

$$6\left(\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{000}_{j_3j_2j_1}\right)^2\right)\le C(T-t)^2, \quad (141)$$

$$2\left(\frac{1}{4}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}^{01}_{j_2j_1}\right)^2\right)\le C(T-t), \quad (142)$$

$$2\left(\frac{1}{12}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}^{10}_{j_2j_1}\right)^2\right)\le C(T-t), \quad (143)$$

$$24\left(\frac{1}{24}-\frac{1}{256}\sum_{j_1,\ldots,j_4=0}^{q_3}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0000}_{j_4\ldots j_1}\right)^2\right)\le$$

$$\le C(T-t). \quad (144)$$

Strong Taylor–Itô scheme (15) with convergence order 2.5

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\le C(T-t)^4,$$

$$6\left(\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{000}_{j_3j_2j_1}\right)^2\right)\le C(T-t)^3, \quad (145)$$

$$2\left(\frac{1}{4}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}^{01}_{j_2j_1}\right)^2\right)\le C(T-t)^2, \quad (146)$$

$$2\left(\frac{1}{12}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}^{10}_{j_2j_1}\right)^2\right)\le C(T-t)^2, \quad (147)$$

$$24\left(\frac{1}{24}-\frac{1}{256}\sum_{j_1,\ldots,j_4=0}^{q_3}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0000}_{j_4\ldots j_1}\right)^2\right)\le$$

$$\le C(T-t)^2, \quad (148)$$

$$6\left(\frac{1}{10}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_8}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{001}_{j_3j_2j_1}\right)^2\right)\le$$

$$\le C(T-t), \quad (149)$$

$$6\left(\frac{1}{20}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_9}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{010}\right)^2\right)\leq$$
$$\leq C(T-t), \tag{150}$$
$$6\left(\frac{1}{60}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_{10}}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{100}\right)^2\right)\leq$$
$$\leq C(T-t), \tag{151}$$

$$120\left(\frac{1}{120}-\frac{1}{32^2}\sum_{j_1,\ldots,j_5=0}^{q_4}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)\times\right.$$
$$\left.\times\left(\bar{C}_{j_5\ldots j_1}^{00000}\right)^2\right)\leq C(T-t). \tag{152}$$

Strong Taylor–Itô scheme (16) with convergence order 3.0

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\leq C(T-t)^5,$$
$$6\left(\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{000}\right)^2\right)\leq C(T-t)^4, \tag{153}$$
$$2\left(\frac{1}{4}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}_{j_2j_1}^{01}\right)^2\right)\leq C(T-t)^3, \tag{154}$$
$$2\left(\frac{1}{12}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}_{j_2j_1}^{10}\right)^2\right)\leq C(T-t)^3, \tag{155}$$

$$24\left(\frac{1}{24}-\frac{1}{256}\sum_{j_1,\ldots,j_4=0}^{q_3}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}_{j_4\ldots j_1}^{0000}\right)^2\right)\leq$$
$$\leq C(T-t)^3, \tag{156}$$
$$6\left(\frac{1}{10}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_8}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{001}\right)^2\right)\leq$$
$$\leq C(T-t)^2, \tag{157}$$

$$6\left(\frac{1}{20}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_9}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{010}_{j_3j_2j_1}\right)^2\right)\le$$

$$\le C(T-t)^2, \tag{158}$$

$$6\left(\frac{1}{60}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_{10}}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{100}_{j_3j_2j_1}\right)^2\right)\le$$

$$\le C(T-t)^2, \tag{159}$$

$$120\left(\frac{1}{120}-\frac{1}{32^2}\sum_{j_1,\dots,j_5=0}^{q_4}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)\times\right.$$

$$\left.\times\left(\bar{C}^{00000}_{j_5\dots j_1}\right)^2\right)\le C(T-t)^2, \tag{160}$$

$$2\left(\frac{1}{30}-\frac{1}{256}\sum_{j_1,j_2=0}^{q_5}(2j_1+1)(2j_2+1)\left(\bar{C}^{20}_{j_2j_1}\right)^2\right)\le C(T-t), \tag{161}$$

$$2\left(\frac{1}{18}-\frac{1}{256}\sum_{j_1,j_2=0}^{q_6}(2j_1+1)(2j_2+1)\left(\bar{C}^{11}_{j_2j_1}\right)^2\right)\le C(T-t), \tag{162}$$

$$2\left(\frac{1}{6}-\frac{1}{256}\sum_{j_1,j_2=0}^{q_7}(2j_1+1)(2j_2+1)\left(\bar{C}^{02}_{j_2j_1}\right)^2\right)\le C(T-t), \tag{163}$$

$$24\left(\frac{1}{36}-\frac{1}{32^2}\sum_{j_1,\dots,j_4=0}^{q_{11}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0001}_{j_4\dots j_1}\right)^2\right)\le$$

$$\le C(T-t), \tag{164}$$

$$24\left(\frac{1}{60}-\frac{1}{32^2}\sum_{j_1,\dots,j_4=0}^{q_{12}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0010}_{j_4\dots j_1}\right)^2\right)\le$$

$$\le C(T-t), \tag{165}$$

$$24\left(\frac{1}{120}-\frac{1}{32^2}\sum_{j_1,\dots,j_4=0}^{q_{13}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0100}_{j_4\dots j_1}\right)^2\right)\le$$

$$\le C(T-t), \tag{166}$$

$$24\left(\frac{1}{360}-\frac{1}{32^2}\sum_{j_1,\ldots,j_4=0}^{q_{14}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}_{j_4\ldots j_1}^{1000}\right)^2\right)\leq$$

$$\leq C(T-t), \tag{167}$$

$$720\left(\frac{1}{720}-\frac{1}{64^2}\sum_{j_1,\ldots,j_6=0}^{q_{15}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)(2j_6+1)\times\right.$$

$$\left.\times\left(\bar{C}_{j_6\ldots j_1}^{000000}\right)^2\right)\leq C(T-t). \tag{168}$$

Taking into account Theorem 3 and the results of Listings 5 and 6 (see Section 5) we decided to exclude the multiplier factors $k!$ from the left-hand sides of (140), (141)–(144), (145)–(152), (153)–(168). The detailed numerical confirmation of the mentioned possibility can be found in [73]. This means that we will use the following conditions for choosing the numbers $q, q_1, q_2, \ldots, q_{15}$ for the numerical schemes (12)–(16) (constant C is independent of $T-t$ (see below)).

Milstein scheme (12)

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\leq C(T-t). \tag{169}$$

Strong Taylor–Itô scheme (13) with convergence order 1.5

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\leq C(T-t)^2, \tag{170}$$

$$\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{000}\right)^2\leq C(T-t). \tag{171}$$

Strong Taylor–Itô scheme (14) with convergence order 2.0

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\leq C(T-t)^3, \tag{172}$$

$$\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{000}\right)^2\leq C(T-t)^2, \tag{173}$$

$$\frac{1}{4}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}_{j_2j_1}^{01}\right)^2\le C(T-t), \tag{174}$$

$$\frac{1}{12}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}_{j_2j_1}^{10}\right)^2\le C(T-t), \tag{175}$$

$$\frac{1}{24}-\frac{1}{256}\sum_{j_1,\ldots,j_4=0}^{q_3}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}_{j_4\ldots j_1}^{0000}\right)^2\le C(T-t). \tag{176}$$

Strong Taylor–Itô scheme (15) with convergence order 2.5

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\le C(T-t)^4, \tag{177}$$

$$\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{000}\right)^2\le C(T-t)^3, \tag{178}$$

$$\frac{1}{4}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}_{j_2j_1}^{01}\right)^2\le C(T-t)^2, \tag{179}$$

$$\frac{1}{12}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}_{j_2j_1}^{10}\right)^2\le C(T-t)^2, \tag{180}$$

$$\frac{1}{24}-\frac{1}{256}\sum_{j_1,\ldots,j_4=0}^{q_3}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}_{j_4\ldots j_1}^{0000}\right)^2\le C(T-t)^2, \tag{181}$$

$$\frac{1}{10}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_8}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{001}\right)^2\le C(T-t), \tag{182}$$

$$\frac{1}{20}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_9}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{010}\right)^2\le C(T-t), \tag{183}$$

$$\frac{1}{60}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_{10}}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}_{j_3j_2j_1}^{100}\right)^2\le C(T-t), \tag{184}$$

$$\frac{1}{120}-\frac{1}{32^2}\sum_{j_1,\ldots,j_5=0}^{q_4}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)\left(\bar{C}_{j_5\ldots j_1}^{00000}\right)^2\le$$

$$\le C(T-t), \tag{185}$$

Strong Taylor–Itô scheme (16) with convergence order 3.0

$$\frac{1}{2}\left(\frac{1}{2}-\sum_{i=1}^{q}\frac{1}{4i^2-1}\right)\le C(T-t)^5, \tag{186}$$

$$\frac{1}{6}-\frac{1}{64}\sum_{j_1,j_2,j_3=0}^{q_1}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{000}_{j_3j_2j_1}\right)^2\le C(T-t)^4, \tag{187}$$

$$\frac{1}{4}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}^{01}_{j_2j_1}\right)^2\le C(T-t)^3, \tag{188}$$

$$\frac{1}{12}-\frac{1}{64}\sum_{j_1,j_2=0}^{q_2}(2j_1+1)(2j_2+1)\left(\bar{C}^{10}_{j_2j_1}\right)^2\le C(T-t)^3, \tag{189}$$

$$\frac{1}{24}-\frac{1}{256}\sum_{j_1,\ldots,j_4=0}^{q_3}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0000}_{j_4\ldots j_1}\right)^2\le C(T-t)^3, \tag{190}$$

$$\frac{1}{10}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_8}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{001}_{j_3j_2j_1}\right)^2\le C(T-t)^2, \tag{191}$$

$$\frac{1}{20}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_9}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{010}_{j_3j_2j_1}\right)^2\le C(T-t)^2, \tag{192}$$

$$\frac{1}{60}-\frac{1}{256}\sum_{j_1,j_2,j_3=0}^{q_{10}}(2j_1+1)(2j_2+1)(2j_3+1)\left(\bar{C}^{100}_{j_3j_2j_1}\right)^2\le C(T-t)^2, \tag{193}$$

$$\frac{1}{120}-\frac{1}{32^2}\sum_{j_1,\ldots,j_5=0}^{q_4}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)\left(\bar{C}^{00000}_{j_5\ldots j_1}\right)^2\le$$

$$\le C(T-t)^2, \tag{194}$$

$$\frac{1}{30}-\frac{1}{256}\sum_{j_1,j_2=0}^{q_5}(2j_1+1)(2j_2+1)\left(\bar{C}^{20}_{j_2j_1}\right)^2\le C(T-t), \tag{195}$$

$$\frac{1}{18}-\frac{1}{256}\sum_{j_1,j_2=0}^{q_6}(2j_1+1)(2j_2+1)\left(\bar{C}^{11}_{j_2j_1}\right)^2\le C(T-t), \tag{196}$$

$$\frac{1}{6}-\frac{1}{256}\sum_{j_1,j_2=0}^{q_7}(2j_1+1)(2j_2+1)\left(\bar{C}^{02}_{j_2j_1}\right)^2\le C(T-t), \tag{197}$$

$$\frac{1}{36}-\frac{1}{32^2}\sum_{j_1,\ldots,j_4=0}^{q_{11}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0001}_{j_4\ldots j_1}\right)^2\le C(T-t), \quad (198)$$

$$\frac{1}{60}-\frac{1}{32^2}\sum_{j_1,\ldots,j_4=0}^{q_{12}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0010}_{j_4\ldots j_1}\right)^2\le C(T-t), \quad (199)$$

$$\frac{1}{120}-\frac{1}{32^2}\sum_{j_1,\ldots,j_4=0}^{q_{13}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{0100}_{j_4\ldots j_1}\right)^2\le C(T-t), \quad (200)$$

$$\frac{1}{360}-\frac{1}{32^2}\sum_{j_1,\ldots,j_4=0}^{q_{14}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)\left(\bar{C}^{1000}_{j_4\ldots j_1}\right)^2\le C(T-t), \quad (201)$$

$$\frac{1}{720}-\frac{1}{64^2}\sum_{j_1,\ldots,j_6=0}^{q_{15}}(2j_1+1)(2j_2+1)(2j_3+1)(2j_4+1)(2j_5+1)(2j_6+1)\times$$
$$\times\left(\bar{C}^{000000}_{j_6\ldots j_1}\right)^2\le C(T-t). \quad (202)$$

2.6 Approximations of Iterated Stratonovich Stochastic Integrals from the Numerical Schemes (24)–(28) Using Legendre Polynomials

This section is devoted to approximation of the Stratonovich stochastic integrals (20) of multiplicities 1 to 6 based on Theorem 2 and Hypothesis 1. At that we will use multiple Fourier–Legendre series for approximation of the mentioned stochastic integrals.

The numerical schemes (24)–(28) contain the following set (see (20)) of iterated Stratonovich stochastic integrals

$$I^{*(i_1)}_{(0)T,t},\quad I^{*(i_1)}_{(1)T,t},\quad I^{*(i_1)}_{(2)T,t},\quad I^{*(i_1i_2)}_{(00)T,t},\quad I^{*(i_1i_2)}_{(10)T,t},\quad I^{*(i_1i_2)}_{(01)T,t},\quad I^{*(i_1i_2i_3)}_{(000)T,t},\quad I^{*(i_1i_2i_3i_4)}_{(0000)T,t}, \quad (203)$$

$$I^{*(i_1i_2i_3i_4i_5)}_{(00000)T,t},\quad I^{*(i_1i_2)}_{(02)T,t},\quad I^{*(i_1i_2)}_{(20)T,t},\quad I^{*(i_1i_2)}_{(11)T,t},\quad I^{*(i_1i_2i_3)}_{(100)T,t},\quad I^{*(i_1i_2i_3)}_{(010)T,t},\quad I^{*(i_1i_2i_3)}_{(001)T,t}, \quad (204)$$

$$I^{*(i_1i_2i_3i_4)}_{(0001)T,t},\quad I^{*(i_1i_2i_3i_4)}_{(0010)T,t},\quad I^{*(i_1i_2i_3i_4)}_{(0100)T,t},\quad I^{*(i_1i_2i_3i_4)}_{(1000)T,t},\quad I^{*(i_1i_2i_3i_4i_5i_6)}_{(000000)T,t}. \quad (205)$$

Using Theorem 2, Hypothesis 1, and well known properties of the Legendre polynomials, we obtain the following formulas for numerical modeling of the stochastic integrals (203)–(205) [26], [42]-[46], [58], [61], [67], [68], [70]-[72]

$$I^{*(i_1)}_{(0)T,t}=\sqrt{T-t}\,\zeta^{(i_1)}_0, \quad (206)$$

$$I^{*(i_1)}_{(1)T,t} = -\frac{(T-t)^{3/2}}{2}\left(\zeta_0^{(i_1)} + \frac{1}{\sqrt{3}}\zeta_1^{(i_1)}\right), \tag{207}$$

$$I^{*(i_1)}_{(2)T,t} = \frac{(T-t)^{5/2}}{3}\left(\zeta_0^{(i_1)} + \frac{\sqrt{3}}{2}\zeta_1^{(i_1)} + \frac{1}{2\sqrt{5}}\zeta_2^{(i_1)}\right), \tag{208}$$

$$I^{*(i_1 i_2)q}_{(00)T,t} = \frac{T-t}{2}\left(\zeta_0^{(i_1)}\zeta_0^{(i_2)} + \sum_{i=1}^{q}\frac{1}{\sqrt{4i^2-1}}\left(\zeta_{i-1}^{(i_1)}\zeta_i^{(i_2)} - \zeta_i^{(i_1)}\zeta_{i-1}^{(i_2)}\right)\right), \tag{209}$$

$$I^{*(i_1 i_2 i_3)q_1}_{(000)T,t} = \sum_{j_1,j_2,j_3=0}^{q_1} C^{000}_{j_3 j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}, \tag{210}$$

$$I^{*(i_1 i_2)q_2}_{(10)T,t} = \sum_{j_1,j_2=0}^{q_2} C^{10}_{j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}, \tag{211}$$

$$I^{*(i_1 i_2)q_2}_{(01)T,t} = \sum_{j_1,j_2=0}^{q_2} C^{01}_{j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}, \tag{212}$$

$$I^{(i_1 i_2 i_3 i_4)q_3}_{(0000)T,t} = \sum_{j_1,j_2,j_3,j_4=0}^{q_3} C^{0000}_{j_4 j_3 j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}, \tag{213}$$

$$I^{(i_1 i_2 i_3 i_4 i_5)q_4}_{(00000)T,t} = \sum_{j_1,j_2,j_3,j_4,j_5=0}^{q_4} C^{00000}_{j_5 j_4 j_3 j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}\zeta_{j_4}^{(i_4)}\zeta_{j_5}^{(i_5)}, \tag{214}$$

$$I^{*(i_1 i_2)q_5}_{(20)T,t} = \sum_{j_1,j_2=0}^{q_5} C^{20}_{j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}, \tag{215}$$

$$I^{(i_1 i_2)q_6}_{(11)T,t} = \sum_{j_1,j_2=0}^{q_6} C^{11}_{j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}, \tag{216}$$

$$I^{(i_1 i_2)q_7}_{(02)T,t} = \sum_{j_1,j_2=0}^{q_7} C^{02}_{j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}, \tag{217}$$

$$I^{*(i_1 i_2 i_3)q_8}_{(001)T,t} = \sum_{j_1,j_2,j_3=0}^{q_8} C^{001}_{j_3 j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}, \tag{218}$$

$$I^{(i_1 i_2 i_3)q_9}_{(010)T,t} = \sum_{j_1,j_2,j_3=0}^{q_9} C^{010}_{j_3 j_2 j_1}\zeta_{j_1}^{(i_1)}\zeta_{j_2}^{(i_2)}\zeta_{j_3}^{(i_3)}, \tag{219}$$

$$I_{(100)T,t}^{(i_1i_2i_3)q_{10}} = \sum_{j_1,j_2,j_3=0}^{q_{10}} C_{j_3j_2j_1}^{100} \zeta_{j_1}^{(i_1)} \zeta_{j_2}^{(i_2)} \zeta_{j_3}^{(i_3)}, \tag{220}$$

$$I_{(0001)T,t}^{(i_1i_2i_3i_4)q_{11}} = \sum_{j_1,j_2,j_3,j_4=0}^{q_{11}} C_{j_4j_3j_2j_1}^{0001} \zeta_{j_1}^{(i_1)} \zeta_{j_2}^{(i_2)} \zeta_{j_3}^{(i_3)} \zeta_{j_4}^{(i_4)}, \tag{221}$$

$$I_{(0010)T,t}^{(i_1i_2i_3i_4)q_{12}} \sum_{j_1,j_2,j_3,j_4=0}^{q_{12}} C_{j_4j_3j_2j_1}^{0010} \zeta_{j_1}^{(i_1)} \zeta_{j_2}^{(i_2)} \zeta_{j_3}^{(i_3)} \zeta_{j_4}^{(i_4)}, \tag{222}$$

$$I_{(0100)T,t}^{(i_1i_2i_3i_4)q_{13}} = \sum_{j_1,j_2,j_3,j_4=0}^{q_{13}} C_{j_4j_3j_2j_1}^{0100} \zeta_{j_1}^{(i_1)} \zeta_{j_2}^{(i_2)} \zeta_{j_3}^{(i_3)} \zeta_{j_4}^{(i_4)}, \tag{223}$$

$$I_{(1000)T,t}^{(i_1i_2i_3i_4)q_{14}} = \sum_{j_1,j_2,j_3,j_4=0}^{q_{14}} C_{j_4j_3j_2j_1}^{1000} \zeta_{j_1}^{(i_1)} \zeta_{j_2}^{(i_2)} \zeta_{j_3}^{(i_3)} \zeta_{j_4}^{(i_4)}, \tag{224}$$

$$I_{(000000)T,t}^{(i_1i_2i_3i_4i_5i_6)q_{15}} = \sum_{j_1,j_2,j_3,j_4,j_5,j_6=0}^{q_{15}} C_{j_6j_5j_4j_3j_2j_1}^{000000} \zeta_{j_1}^{(i_1)} \zeta_{j_2}^{(i_2)} \zeta_{j_3}^{(i_3)} \zeta_{j_4}^{(i_4)} \zeta_{j_5}^{(i_5)} \zeta_{j_6}^{(i_6)}, \tag{225}$$

where $\mathbf{1}_A$ is the indicator of the set A; another notations are the same as in Section 2.4.

The question on choosing the numbers $q_1, q_2, \ldots, q_{15}$ in (210)–(225) turned out to be nontrivial [26] (Chapter 5). The expansions (210)–(225) for iterated Stratonovich stochastic integrals are simpler than their analogues (75)–(90) for iterated Itô stochastic integrals. However, the calculation of the mean-square approximation error for iterated Stratonovich stochastic integrals turns out to be much more difficult than for iterated Itô stochastic integrals [26] (Chapter 5). Below we give some reasoning regarding this problem.

Denote

$$E_{(l_1\ldots l_k)T,t}^{*(i_1\ldots i_k)p} \stackrel{\mathsf{def}}{=} \mathsf{M}\left\{\left(I_{(l_1\ldots l_k)T,t}^{*(i_1\ldots i_k)} - I_{(l_1\ldots l_k)T,t}^{*(i_1\ldots i_k)p}\right)^2\right\},$$

where $I_{(l_1\ldots l_k)T,t}^{*(i_1\ldots i_k)p}$, $p \in \mathbb{N}$ is the approximation of the iterated Stratonovich stochastic integral $I_{(l_1\ldots l_k)T,t}^{*(i_1\ldots i_k)}$.

From (209) for $i_1 \neq i_2$ we obtain [26], [67]

$$E_{(00)T,t}^{*(i_1i_2)q} = \frac{(T-t)^2}{2} \sum_{i=q+1}^{\infty} \frac{1}{4i^2-1} \leq \frac{(T-t)^2}{2} \int_q^{\infty} \frac{1}{4x^2-1} dx =$$

$$= -\frac{(T-t)^2}{8}\ln\left|1-\frac{2}{2q+1}\right| \le C_1\frac{(T-t)^2}{q}, \tag{226}$$

where constant C_1 is independent of q.

It is easy to notice that for a sufficiently small $T-t$ (recall that $T-t \ll 1$ since it is a step of integration for numerical schemes for Itô SDEs) there exists a constant C_2 such that

$$E^{*(i_1\ldots i_k)q}_{(l_1\ldots l_k)T,t} \le C_2 E^{*(i_1 i_2)q}_{(00)T,t}. \tag{227}$$

From (226) and (227) we finally obtain

$$E^{*(i_1\ldots i_k)q}_{(l_1\ldots l_k)T,t} \le C\frac{(T-t)^2}{q}, \tag{228}$$

where constant C does not depend on $T-t$. The same idea can be found in [2] in the framework of the method of approximation of iterated Stratonovich stochastic integrals based on the trigonometric expansion of the Brownian bridge process.

Obviously, we can get more information about the numbers $q_1, q_2, \ldots, q_{15}$ (these numbers are different for different iterated Stratonovich stochastic integrals) using the another approach. Since

$$J^*[\psi^{(k)}]_{T,t} = J[\psi^{(k)}]_{T,t} \quad \text{w. p. 1}$$

for pairwise different $i_1, \ldots, i_k = 1, \ldots, m$, where $J[\psi^{(k)}]_{T,t}$, $J^*[\psi^{(k)}]_{T,t}$ are defined by (2) and (3) correspondingly, then for pairwise different $i_1, \ldots, i_6 = 1, \ldots, m$ from (51) we obtain [26], [67]

$$E^{*(i_1 i_2)q}_{(00)T,t} = \frac{(T-t)^2}{2}\left(\frac{1}{2} - \sum_{i=1}^{q}\frac{1}{4i^2-1}\right), \tag{229}$$

$$E^{*(i_1 i_2 i_3)q_1}_{(000)T,t} = \frac{(T-t)^3}{6} - \sum_{j_3,j_2,j_1=0}^{q_1}\left(C^{000}_{j_3 j_2 j_1}\right)^2, \tag{230}$$

$$E^{*(i_1 i_2)q_2}_{(01)T,t} = \frac{(T-t)^4}{4} - \sum_{j_1,j_2=0}^{q_2}\left(C^{01}_{j_2 j_1}\right)^2, \tag{231}$$

$$E^{*(i_1 i_2)q_2}_{(10)T,t} = \frac{(T-t)^4}{12} - \sum_{j_1,j_2=0}^{q_2}\left(C^{10}_{j_2 j_1}\right)^2, \tag{232}$$

$$E^{*(i_1\ldots i_4)q_3}_{(0000)T,t} = \frac{(T-t)^4}{24} - \sum_{j_1,j_2,j_3,j_4=0}^{q_3} \left(C^{0000}_{j_4j_3j_2j_1}\right)^2, \tag{233}$$

$$E^{*(i_1\ldots i_5)q_4}_{(00000)T,t} = \frac{(T-t)^5}{120} - \sum_{j_1,j_2,j_3,j_4,j_5=0}^{q_4} \left(C^{00000}_{j_5i_4i_3i_2j_1}\right)^2, \tag{234}$$

$$E^{*(i_1i_2)q_5}_{(20)T,t} = \frac{(T-t)^6}{30} - \sum_{j_2,j_1=0}^{q_5} \left(C^{20}_{j_2j_1}\right)^2, \tag{235}$$

$$E^{*(i_1i_2)q_6}_{(11)T,t} = \frac{(T-t)^6}{18} - \sum_{j_2,j_1=0}^{q_6} \left(C^{11}_{j_2j_1}\right)^2, \tag{236}$$

$$E^{*(i_1i_2)q_7}_{(02)T,t} = \frac{(T-t)^6}{6} - \sum_{j_2,j_1=0}^{q_7} \left(C^{02}_{j_2j_1}\right)^2, \tag{237}$$

$$E^{*(i_1i_2i_3)q_8}_{(001)T,t} = \frac{(T-t)^5}{10} - \sum_{j_1,j_2,j_3=0}^{q_8} \left(C^{001}_{j_3j_2j_1}\right)^2, \tag{238}$$

$$E^{*(i_1i_2i_3)q_9}_{(010)T,t} = \frac{(T-t)^5}{20} - \sum_{j_1,j_2,j_3=0}^{q_9} \left(C^{010}_{j_3j_2j_1}\right)^2, \tag{239}$$

$$E^{*(i_1i_2i_3)q_{10}}_{(100)T,t} = \frac{(T-t)^5}{60} - \sum_{j_1,j_2,j_3=0}^{q_{10}} \left(C^{100}_{j_3j_2j_1}\right)^2, \tag{240}$$

$$E^{*(i_1\ldots i_4)q_{11}}_{(0001)T,t} = \frac{(T-t)^6}{36} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{11}} \left(C^{0001}_{j_4j_3j_2j_1}\right)^2, \tag{241}$$

$$E^{*(i_1\ldots i_4)q_{12}}_{(0010)T,t} = \frac{(T-t)^6}{60} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{12}} \left(C^{0010}_{j_4j_3j_2j_1}\right)^2, \tag{242}$$

$$E^{*(i_1\ldots i_4)q_{13}}_{(0100)T,t} = \frac{(T-t)^6}{120} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{13}} \left(C^{0100}_{j_4j_3j_2j_1}\right)^2, \tag{243}$$

$$E^{*(i_1\ldots i_4)q_{14}}_{(1000)T,t} = \frac{(T-t)^6}{360} - \sum_{j_1,j_2,j_3,j_4=0}^{q_{14}} \left(C^{1000}_{j_4j_3j_2j_1}\right)^2, \tag{244}$$

$$E^{*(i_1\ldots i_6)q_{15}}_{(000000)T,t} = \frac{(T-t)^6}{720} - \sum_{j_1,j_2,j_3,j_4,j_5,j_6=0}^{q_{15}} \left(C^{000000}_{j_6j_5j_4j_3j_2j_1}\right)^2. \tag{245}$$

Table 1. High-order strong Taylor–Stratonovich schemes.

Order of convergence	Scheme	Conditions for choosing the numbers $q, q_1, \ldots, q_{15}$
1.0	(24)	(169)
1.5	(25)	(170), (171)
2.0	(26)	(172)–(176)
2.5	(27)	(177)–(185)
3.0	(28)	(186)–(202)

Taking into account (229)–(245) and the results of paper [73], we use in the SDE-MATH software package the conditions from Table 1 for choosing the numbers $q, q_1, q_2, \ldots, q_{15}$ for the numerical schemes (24)–(28).

Note that in the SDE-MATH software package, which is presented in the following sections, we use the following upper bounds b on the numbers $q_1, \ldots, q_{15}$

$$b = 56 \quad \text{for} \quad q_1, \qquad b = 15 \quad \text{for} \quad q_2,\ q_3, \qquad b = 6 \quad \text{for} \quad q_4,\ q_8,\ q_9,\ q_{10},$$

$$b = 2 \quad \text{for} \quad q_5,\ q_6,\ q_7,\ q_{11},\ q_{12},\ q_{13},\ q_{14},\ q_{15}.$$

This means that for the implementing of the numerical methods (13)–(16) and (25)–(28) we use in the SDE-MATH software package the following quantities of the exactly calculated Fourier–Legendre coefficients

$$57^3 = 185{,}193 \quad \text{for} \quad C^{000}_{j_3 j_2 j_1},$$

$$16^3 = 4{,}096 \quad \text{for each of} \quad C^{10}_{j_2 j_1},\ C^{01}_{j_2 j_1},$$

$$16^4 = 65{,}536 \quad \text{for} \quad C^{0000}_{j_4 j_3 j_2 j_1},$$

$$7^3 = 343 \quad \text{for each of} \quad C^{100}_{j_3 j_2 j_1},\ C^{010}_{j_3 j_2 j_1},\ C^{001}_{j_3 j_2 j_1},$$

$$7^5 = 16{,}807 \quad \text{for} \quad C^{00000}_{j_5 j_4 j_3 j_2 j_1},$$

$$3^2 = 9 \quad \text{for each of} \quad C^{20}_{j_2 j_1},\ C^{02}_{j_2 j_1},\ C^{11}_{j_2 j_1},$$

$$3^4 = 81 \quad \text{for each of} \quad C^{1000}_{j_4 j_3 j_2 j_1},\ C^{0100}_{j_4 j_3 j_2 j_1},\ C^{0010}_{j_4 j_3 j_2 j_1},\ C^{0001}_{j_4 j_3 j_2 j_1},$$

$$3^6 = 729 \quad \text{for} \quad C^{000000}_{j_6 j_5 j_4 j_3 j_2 j_1}.$$

It should be noted that unlike the method based on Theorems 1–3, existing and well-known approaches to the mean-square approximation of iterated stochastic integrals based on the trigonometric basis functions [2], [3], [7], [27],

[28], [34], [37] do not allow choosing theoretically different numbers q for approximations of different iterated stochastic integrals (starting form the multiplicity 2 of stochastic integrals). Moreover, the noted approaches [2], [3], [7], [27], [28], [34], [37] exclude the possibility for obtaining of approximate and exact expressions for the mean-square approximation error similar to the formulas (49), (50).

2.7 Numerical Algorithm for Linear Stationary Systems of Itô SDEs Based on Spectral Decomposition

Consider the following linear stationary system of Itô SDEs

$$d\mathbf{x}_t = (A\mathbf{x}_t + B\mathbf{u}(t))\,dt + Fd\mathbf{w}_t, \quad \mathbf{x}_0 = \mathbf{x}(0), \quad t \in [0,T], \tag{246}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is a solution of the system (246), $\mathbf{u}(t) : [0,T] \to \mathbb{R}^k$ is a non-random function, $A \in \mathbb{R}^{n\times n}$, $F \in \mathbb{R}^{n\times m}$, $B \in \mathbb{R}^{n\times k}$, and $\mathbf{w}_t$ is a standard m-dimensional Wiener process with independent components $\mathbf{w}_t^{(i)}$, $i = 1,\ldots,m$. Also we suppose that n, m, $k \geq 1$. The process $\mathbf{y}_t = H\mathbf{x}_t \in \mathbb{R}^1$ is interpreted as an output process of the system (246), where $H \in \mathbb{R}^{1\times n}$.

It is well-known that the solution of (246) has the form [4]

$$\mathbf{x}_t = e^{A(t-t_0)}\mathbf{x}_{t_0} + \int\limits_{t_0}^{t} e^{A(t-s)}B\mathbf{u}(s)ds + \int\limits_{t_0}^{t} e^{A(t-s)}Fd\mathbf{w}_s, \quad 0 \leq t_0 \leq t \leq T, \tag{247}$$

where e^C is a matrix exponent

$$\sum_{j=0}^{\infty}\frac{C^j}{j!} \stackrel{\mathsf{def}}{=} e^C,$$

C is a square matrix, and $C^0 \stackrel{\mathsf{def}}{=} I$ is a unity matrix.

Consider the partition $\{\tau_p\}_{p=0}^N$ of $[0,T]$ such that $\tau_p = p\Delta$, $\Delta > 0$. For simplicity, we will suppose that $u(s)$, $s \in [0,T]$ can be approximated by the step function, i.e. $\mathbf{u}(s) \approx \hat{\mathbf{u}}(s)$, $s \in [0,T]$, where $\hat{\mathbf{u}}(s) = \mathbf{u}(\tau_p)$ for $s \in [\tau_p, \tau_{p+1})$, $p = 0, 1, \ldots, N-1$ (more accurate approximations of $u(s)$ are discussed in [62] (also see [58], [61])). Substituting $t = \tau_{p+1}$, $t_0 = \tau_p$, and $\hat{\mathbf{u}}(s)$ instead of $\mathbf{u}(s)$ into (247), we obtain

$$\hat{\mathbf{x}}_{p+1} = e^{A\Delta}\hat{\mathbf{x}}_p + A^{-1}\big(e^{A\Delta} - I\big)B\mathbf{u}(p\Delta) + \tilde{\mathbf{w}}_{p+1}(\Delta), \quad \mathbf{x}_0 = \mathbf{x}(0), \tag{248}$$

where $\hat{\mathbf{x}}_p$ is the approximation of $\mathbf{x}_{\tau_p}$ and

$$\int_0^{\Delta} e^{A(\Delta-s)} F d\mathbf{w}_{s+p\Delta} \stackrel{\mathsf{def}}{=} \tilde{\mathbf{w}}_{p+1}(\Delta).$$

Also we assume that $\hat{\mathbf{y}}_p = H\hat{\mathbf{x}}_p$, where $\hat{\mathbf{y}}_p$ is the approximation of $\mathbf{y}_{\tau_p}$. The random column $\tilde{\mathbf{w}}_{p+1}(\Delta)$ admits the following representation [4]

$$\tilde{\mathbf{w}}_{p+1}(\Delta) = S_D(\Delta)\Lambda_D(\Delta)\bar{\mathbf{w}}_{p+1}, \tag{249}$$

where $\bar{\mathbf{w}}_p \in \mathbb{R}^n$ is a column of independent standard Gaussian random variables such that $\mathsf{M}\left\{\bar{\mathbf{w}}_p \bar{\mathbf{w}}_q^{\mathsf{T}}\right\} = \mathcal{O}$ for $p \neq q$, $\mathcal{O}$ is a zero matrix of size $n \times n$, $S_D(\Delta)$ is a matrix of orthonormal eigenvectors of the matrix $D_f(\Delta)$ and $\Lambda_D^2(\Delta)$ is a diagonal matrix on the main diagonal of which are the eigenvalues of the matrix $D_f(\Delta)$, the matrix $D_f(\Delta)$ is defined by

$$D_f(\Delta) = \mathsf{M}\left\{\tilde{\mathbf{w}}_{p+1}(\Delta)\tilde{\mathbf{w}}_{p+1}^{\mathsf{T}}(\Delta)\right\} = \int_0^{\Delta} \exp(A(\Delta-s))FF^{\mathsf{T}}\exp(A^{\mathsf{T}}(\Delta-s))ds,$$

where C^{T} is a transposed matrix C. Moreover, $D_f(\Delta) = D_f(t)\big|_{t=\Delta}$, where $D_f(t)$ is a solution of the following Cauchy problem [4]

$$\frac{dD_f}{dt}(t) = AD_f(t) + D_f(t)A^{\mathsf{T}} + FF^{\mathsf{T}}, \quad D_f(0) = \mathcal{O}.$$

In the SDE-MATH software package, we implement the numerical modeling of the system (246) by the formulas (248), (249). At that we use Algorithms 2.3–2.6 from [62] (also see [61], Chapter 11) for the implemetation of (249).

3 The Structure of the SDE-MATH Software Package

3.1 Development Tools

The software package was implemented with Python programming language. The main reason to use it is a huge community and significant amount of helpful libraries for calculations and mathematics. The development was performed in free to use Atom text editor[1].

[1]All programs in Python programming language from this paper were written by the first author

3.2 Dependency Libraries

In the development of the SDE-MATH software package such libraries as SymPy, NumPy, PyQt5, and Matplotlib were involved. All these libraries and tools are free and open source.

- SymPy is a Python library able to perform symbolic algebra calculations.
- NumPy is a library which specialization is efficient mathematical calculations. Most part of this library is written in C programming language that guarantees high calculation performance.
- The database is SQLite3. This is a tiny database for a local usage on one machine.
- Matplotlib library is a piece of software used to present obtained results in a best way.
- PyQt5 is a library used to build graphical user interface for the SDE-MATH software package.

3.3 Architecture

Taking into account, that the SDE-MATH software package is oriented on a numerical modeling its architecture is clear. There are two main statements. The first is that mathematical formulas are strongly integrated with SymPy library. By that we mean that they completely rely on SymPy. And the second is usage of database to make some calculations able for caching. The architecture itself is provided on Figure 1. Here all parts of the software package can be seen.

The main package is responsible for startup, so it decides which part of the software package must be started. The software package has several modes of operation. The objectives now are

- Run program to calculate and store the Fourier–Legendre coefficients in few text files with further loading in database.
- Run program with graphical user interface. This is the main program entry for the SDE-MATH software package.

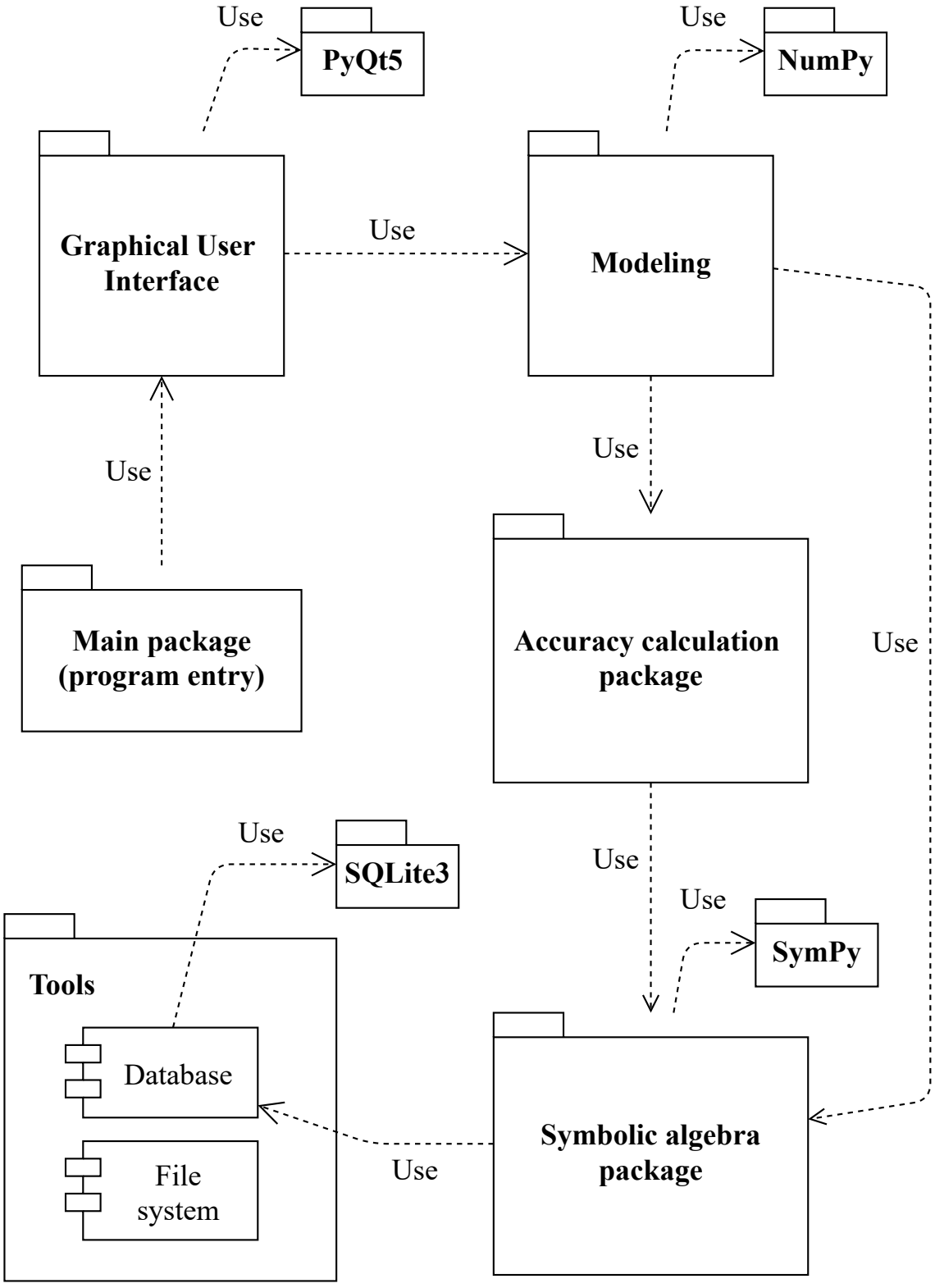


Figure 1: The SDE-MATH software package architecture

On the current state of development the main entry package booting up PyQt5 library with all necessary widgets. More detailed description of this process will be provided later.

Moving further, the modeling package comes up. This package responsible for all work referenced to modeling including initialization of modeling environment, calculations loops and more. Also, it depends on accuracy calculation module deciding which amount of members in each approximation of iterated stochastic integral should be used in modeling of the Itô SDE (1) solution.

Accuracy calculation module accepts the order of strong numerical scheme for the Itô SDE (1) and its integration step and then calculates necessary amount of members in approximations of iterated Itô and Stratonovich stochastic integrals.

Symbolic algebra module is the construction part which combines many supplementary differential operators with strong numerical schemes for the Itô SDE (1). Having these components combined this module performs simplification of resulting formula so the modeling package can do its modeling work.

Tools module provides some functionality related to bootstrap of runtime environment and external instruments such as database and file system.

3.3.1 Integration with SymPy

Class inheritance tree was extended to implement strong numerical schemes for Itô SDEs. While numerical schemes for Itô SDEs were being implemented it was also necessary to implement supplementary subprograms. SymPy is a Python library able to perform symbolic algebra calculations. This is a core part of the project since it is differentiates input functions, builds and simplifies strong numerical schemes for Itô SDEs to model the Itô SDE (1) solution. Without this part the program package cannot be able to provide such flexible input of data.

3.3.2 Purpose of NumPy

NumPy is a library that helps with calculation optimizations in this project. The library specialization is efficient mathematical calculations. The main usage case is to calculate compiled symbolic formulas with it. It has integration with SymPy to replace symbolic functions with high performance numerical functions.

3.3.3 Purpose of SQLite Database

The database was used to store the precalculated Fourier–Legendre coefficients, so getting them from there made numerical modeling much faster, because calculation process for these Fourier–Legendre coefficients involve high-cost symbolic operations. The database contains only one table, and might be thought redundant, but modeling needs hundreds (or even thousands) of precalculated coefficients. Obviously, calculation of them at runtime is terribly inefficient, but text files also not the best choice. Text files provide a sequential access memory and combining different accuracy values $q_1, \ldots, q_{15}$ it causes sequential search which extends time to give the result. That is where database comes up. The random access allows to get any Fourier–Legendre coefficient or any quantity of them which makes solution as flexible as it possible.

The download of precalculated Fourier–Legendre coefficients is built in supplemental subprograms to provide fluent calculation pipeline. Having the precalculated Fourier–Legendre coefficient not found, subprogram initiates calculation for it with following store in the database.

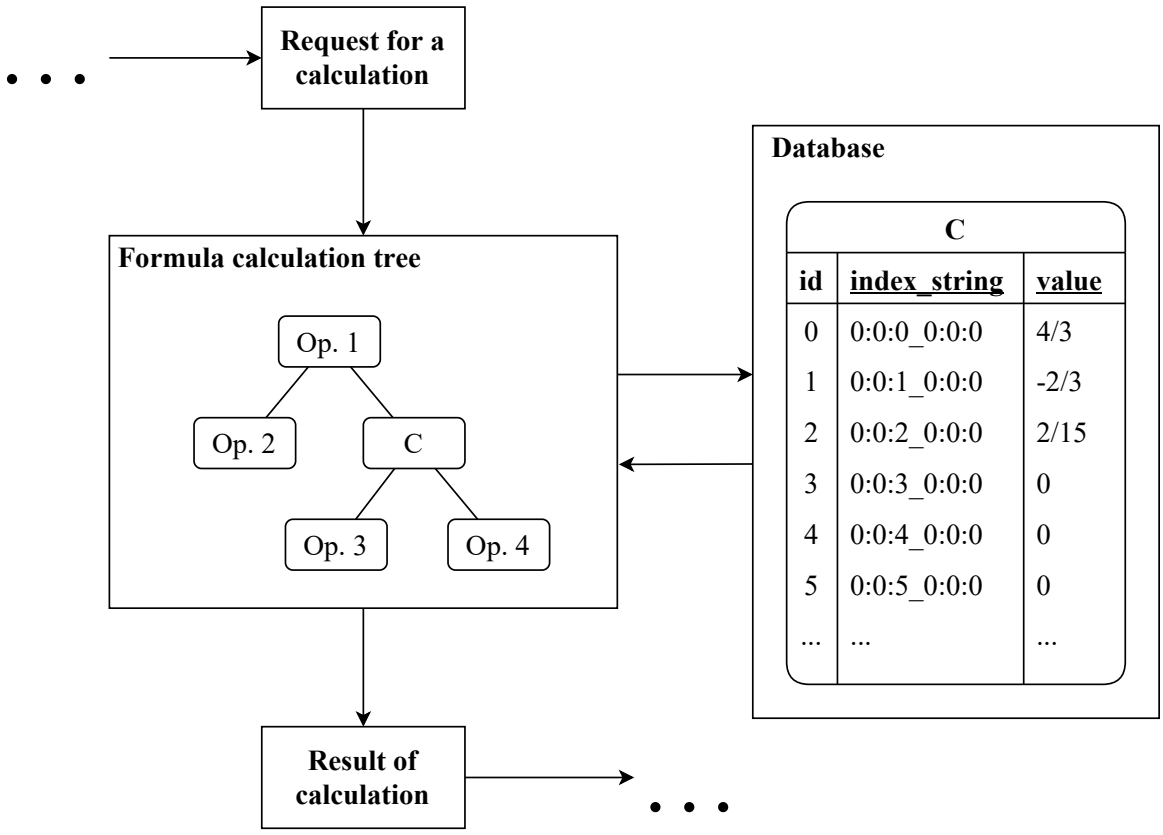


Figure 2: Fourier–Legendre coefficients calculations explanation

It is an interesting note that having mentioned earlier optimization done, the calculations performance were increased in several times. Now the most heavy operation is symbolic simplifications before modeling. The actual modeling takes seconds, for thousands of iterations on m components of stochastic process, so it is not such important how long modeling period of time, as the accuracy that needs to be accomplished.

The scheme of calculation process is presented on Figure 2.

3.3.4 Purpose of Matplotlib

Matplotlib library is a piece of software used to present obtained results in a best way. This library has many features, but feature that needed in this project is to print charts with modeling results in a PyQt5 widget. Thus the data visualization is integrated in graphical user interface.

3.4 Implementation Plan

The implementation of SDE-MATH software package was performed sequentially. The components of SDE-MATH software package were implemented in order of their necessity for calculation pipeline completion.

3.4.1 Calculation of the Fourier–Legendre Coefficients

The Fourier–Legendre coefficients for the approximations of iterated Itô and Stratonovich stochastic integrals were implemented and placed in Listings 43–61. This was the first step since the Fourier–Legendre coefficients involved almost in every strong numerical scheme for the Itô SDE (1).

Also it is important to note that the SDE-MATH software package contains a Python script intended for generating of Fourier–Legendre coefficients using multiprocessing. This script placed in Listing 62 and already contains tasks that were performed to generate about 300,000 Fourier–Legendre coefficients. Similarly, user can run and calculate additional Fourier–Legendre coefficients if they are needed. To determine which Fourier–Legendre coefficients will be calculated user must specify pairs of starting and ending values of components in lower multi-index and specify upper multi-index of the Fourier–Legendre coefficient. For example $(((0, 15), (0, 15), (0, 15)), (0, 1, 0))$. This means that

program calculates the Fourier–Legendre coefficients $C^{010}_{j_3 j_2 j_1}$, where $j_1, j_2, j_3 \in \{0, 1, \ldots, 14\}$.

3.4.2 Differential Operators L, $\bar{L}$, $G_0^{(i)}$, $i = 1, \ldots, m$

Moving further, strong numerical schemes for Itô SDEs rely on the differential operators (4), (5), and (23). They were implemented and placed in Listings 64–67.

3.4.3 Approximations of Iterated Stochastic Integrals

The next step is implementation of approximations of iterated Itô and Stratonovich stochastic integrals for the numerical schemes (12)–(16), (24)–(28). They are implemented and definition of their classes are placed in Listings 69–108.

3.4.4 Strong Numerical Schemes for Itô SDEs

The strong numerical schemes (12)–(16), (24)–(28) for Itô SDEs were implemented. They are placed in Listings 110–131.

3.4.5 Graphical User Interface

Finally, the graphical user interface was implemented. The source codes referenced to graphical user interface are placed in Listings 7–42.

4 Software Package Graphical User Interface

For the SDE-MATH software package mentioned above the graphical user interface was developed. The graphical user interface is important and massive part of SDE-MATH software package because it allows user to perform modeling experiments without programming skills and understanding of program package architecture and principles of work.

4.1 Information Model of The Graphical User Interface

The development of graphical user interface was started from consideration of

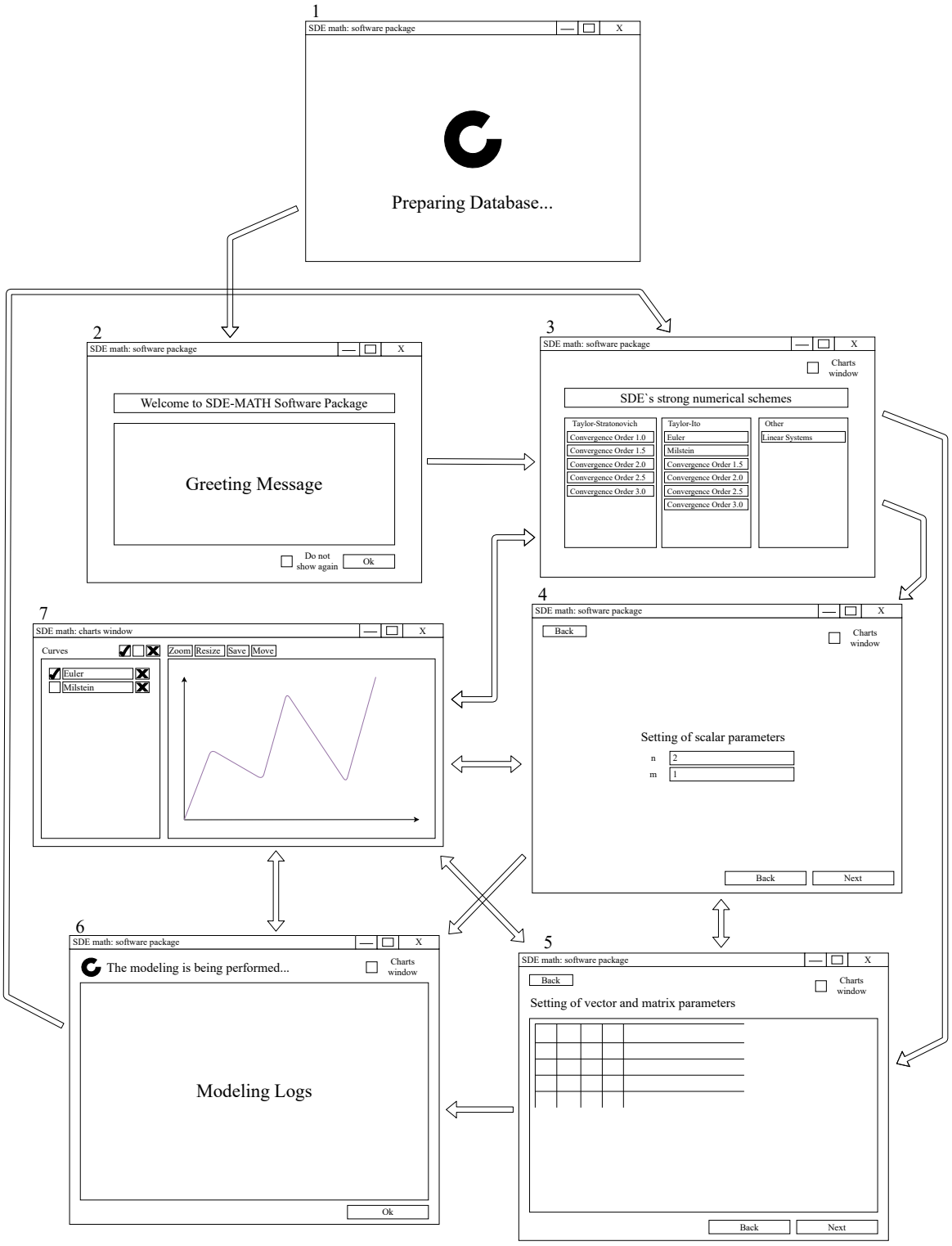


Figure 3: Information model of graphical user interface

experiments and routines which can be performed with the SDE-MATH software package. The graphical user interface is aimed on provision of user capabilities to perform nonlinear and linear systems of Itô SDEs modeling experi-

ments. The information model which schematically describes the graphical user interface structure is presented on Figure 3.

4.1.1 Processing Screens

To represent long duration processes the graphical user interface has two dialogs which can be seen on Figure 3, in Windows 1 and 6. The first one represents database preparing process on application very first run. During this process the Fourier–Legendre coefficients are being loaded into the SQLite database. This screen appears also when user calculates new Fourier–Legendre coefficients. The other screen shows logs during modeling experiment.

4.1.2 Greetings Dialog

After the SDE-MATH software package has completed the database preparation, it shows greeting dialog which represents short information about its purposes. The greeting dialog can be seen on Figure 3 in Window 2.

4.1.3 Main Menu Dialog

In the main menu of the SDE-MATH software package user can choose one of strong numerical schemes for Itô SDEs to perform modeling experiments. The main menu dialog can be seen on Figure 3 in Window 3.

4.1.4 Visualization Tool

It is important to note that the main SDE-MATH software package window has a checkbox in right upper corner which do switching on and off charts window. In any time user can call this window or hide it if it is not needed. The charts window is universal utility for modeling experiments results visualization. This window has few instruments on it. The left side bar contains all curves labels, and control elements for hiding, showing, and deleting curves. On the right side of the window there are plot which draws the curves. The charts window can be seen on Figure 3, it is Window 3.

4.1.5 Data Input Dialogs

Since the software package has options to perform linear Itô SDEs modeling experiments it is necessary to provide user with input fields for numerical data

both scalar and matrix. On the other side, for nonlinear Itô SDEs it is necessary to provide symbolic input. The choice of control elements is conditioned by the above obstacles. On Figure 3, and especially in Windows 4 and 5, these input controls can be seen. There are "LineEditWidget" and "TableWidget" which are sufficient to provide input abilities. The topic of input data validation is also important but to be more accurate referenced to user experience rather than to information model, so it will be described further.

4.2 The User Experience and Implementation Results

The above part represents the structure of software package but not the dynamics and user experience of it. Let us discuss the SDE-MATH software package user experience on few examples provided further on Figures 4–36. This examples represent two scenarios of the SDE-MATH software package use.

The database preparation screen is presented on Figure 4. During the database preparation this screen displays informational message and spinning visualizer of process continuation.

The screen that presented on Figure 5 appears every time when software package runs unless user presses "Ok" button with marked checkbox. In such case this message screen will not be shown again.

On Figure 6 the main menu dialog is presented. In this dialog user can choose any strong numerical scheme for Itô SDEs to perform modeling experiment.

The tooltip example can be seen on Figure 7. Such tooltips displayed with characteristic icon are placed all over software package interface to help user with explanations.

As noted earlier, the dedicated charts window is universal tool for visualization. The specific examples of such visualization are presented on Figure 8,19, 34–36.

The initial state of input dialogs for nonlinear and linear Itô SDEs are displayed on Figures 9 and 20. At that moment user can start to input the data.

The example of wrong scalar data input is presented on Figures 10, 15, 21, and 31. When user input wrong data the error message appears and "Next" or "Perform modeling" button is blocked. The input field is being checked all the user data input process, and as soon as wrong character is entered notification

pops up.

If scalar data is correct the "Next" button is automatically unblocked. On Figures 11, 16, 22, and 32 the examples of scenario are displayed.

On Figures 12, 13, and 28 the example of correct matrix data input is presented. In this particular case the input is symbolic. Symbolic algebra input errors are much harder to determine so this is done on further stages, in modeling runtime.

In the other case when matrix input data are numerical, the validation is performed right after user has finished input. The examples of incorrect matrix numerical input can be found on Figure 24.

When user finishes input with a success the "Next" or "Perform modeling" button is automatically unblocked. On Figures 23, 25–27, 29, and 30 that can be clearly seen.

The Figures 17, 18, and 33 displays sequence of log messages emerged during the modeling process.

After modeling has been done the focus moves to the charts window where obtained modeling results can be seen. The results of modeling is displayed on Figures 19, 34–36. On Figures 35 and 36 the expectations and variances of obtained components of solution are displayed.

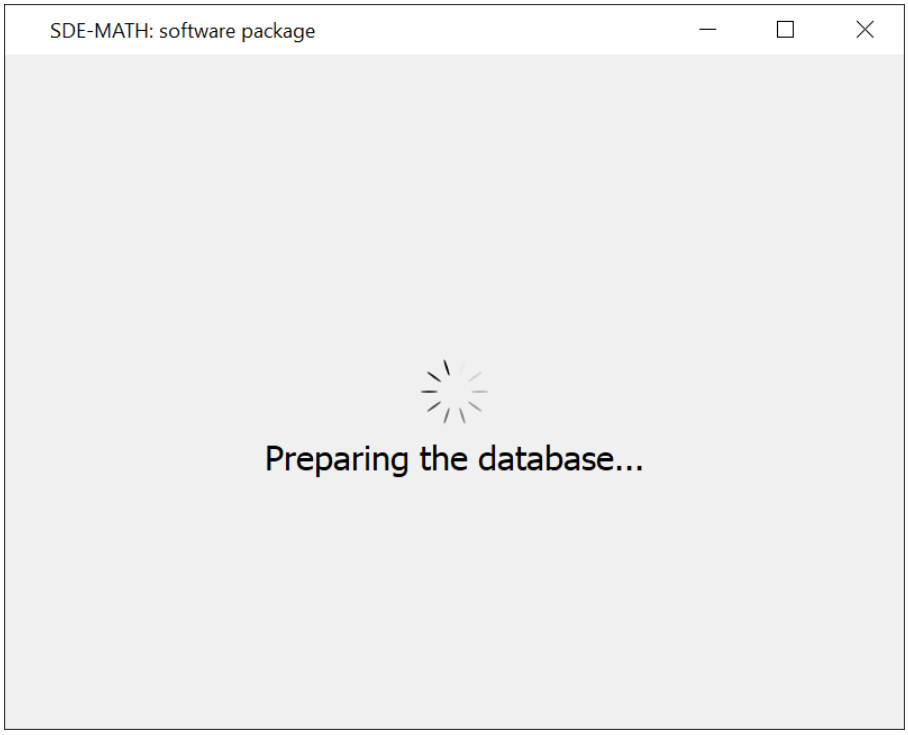


Figure 4: Fourier–Legendre coefficients database preparation screen

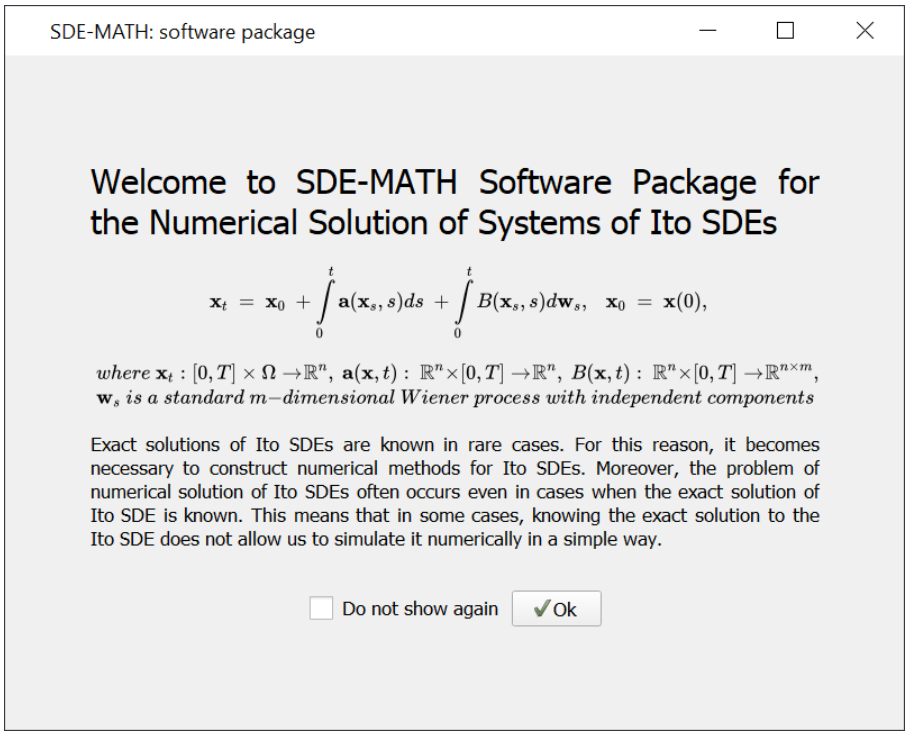


Figure 5: Greetings screen

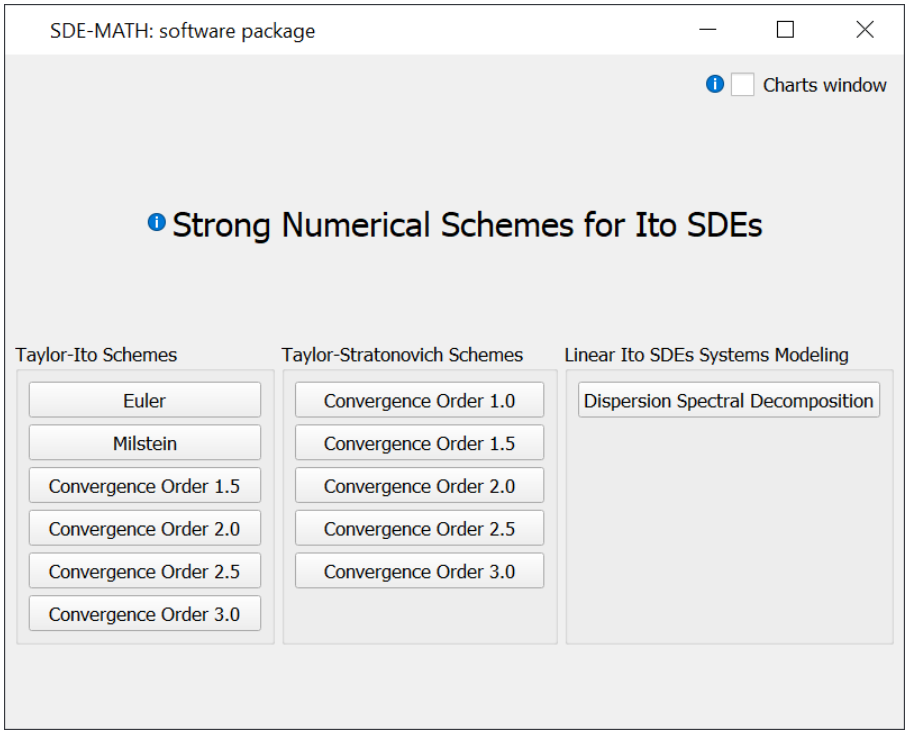


Figure 6: Main menu dialog

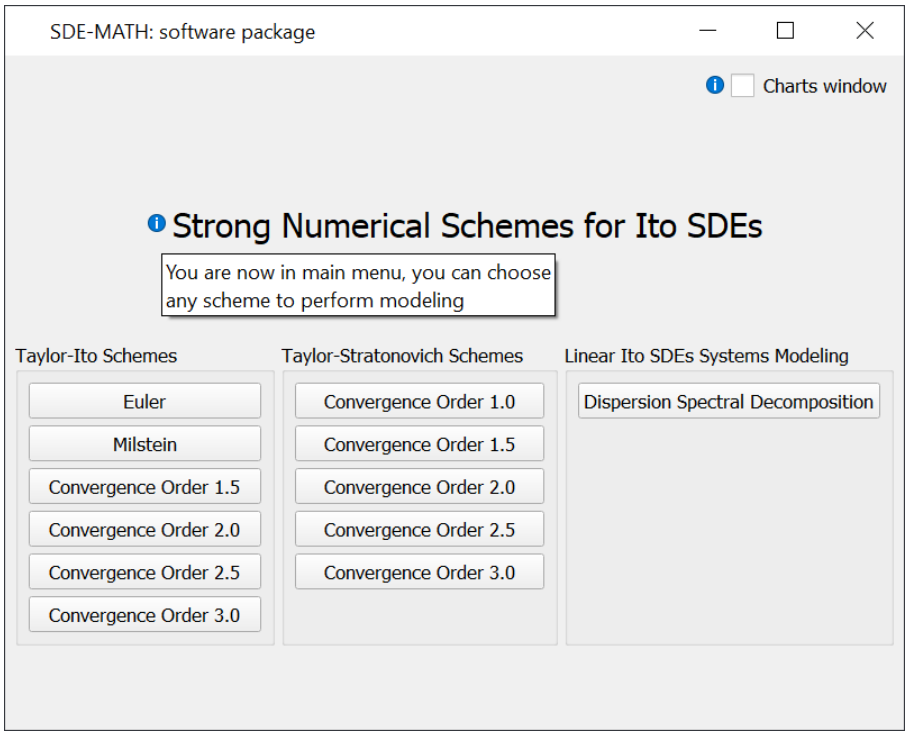


Figure 7: Tooltip

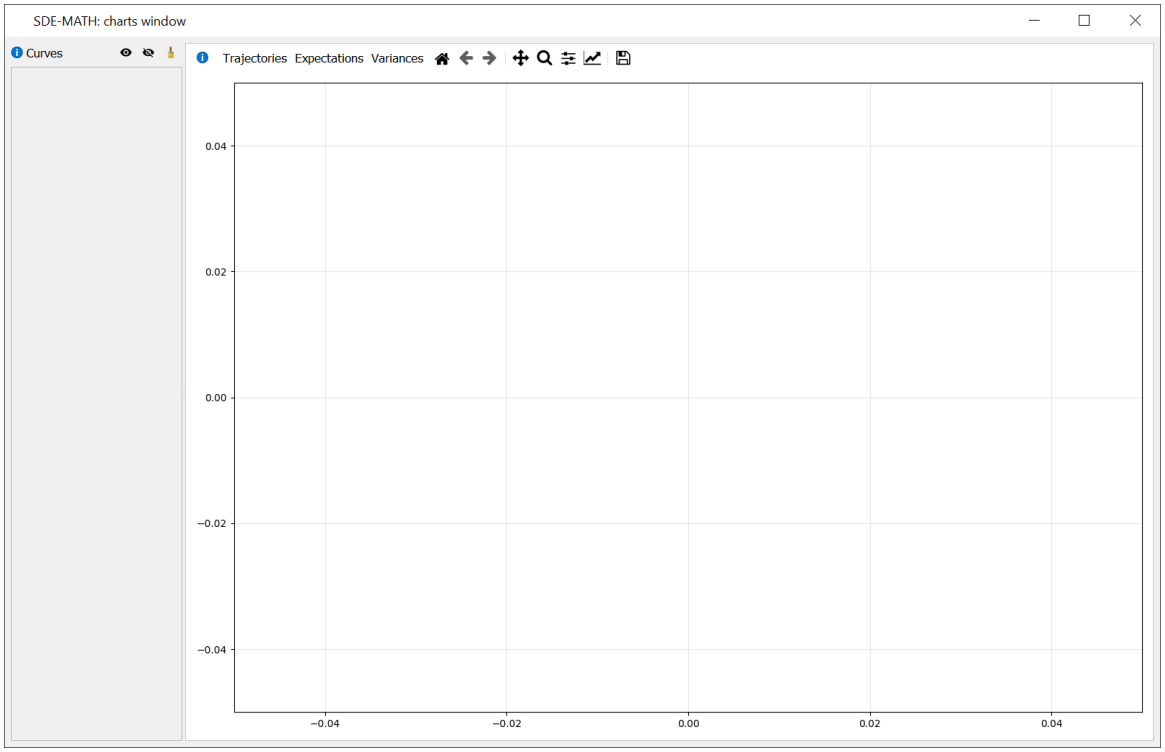


Figure 8: Charts window

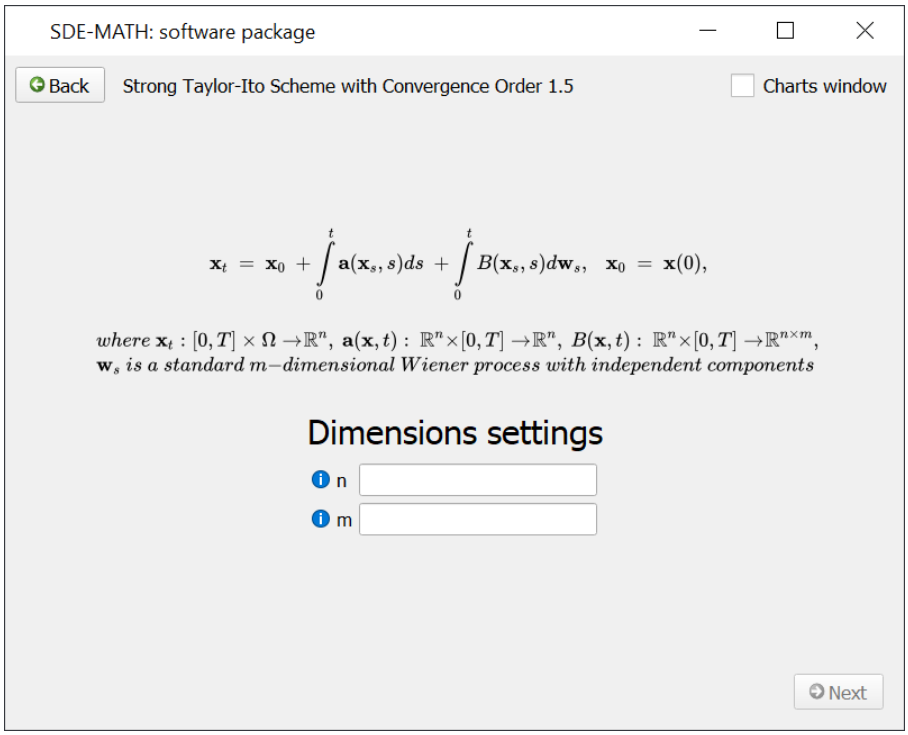


Figure 9: Nonlinear system of Itô SDEs data input

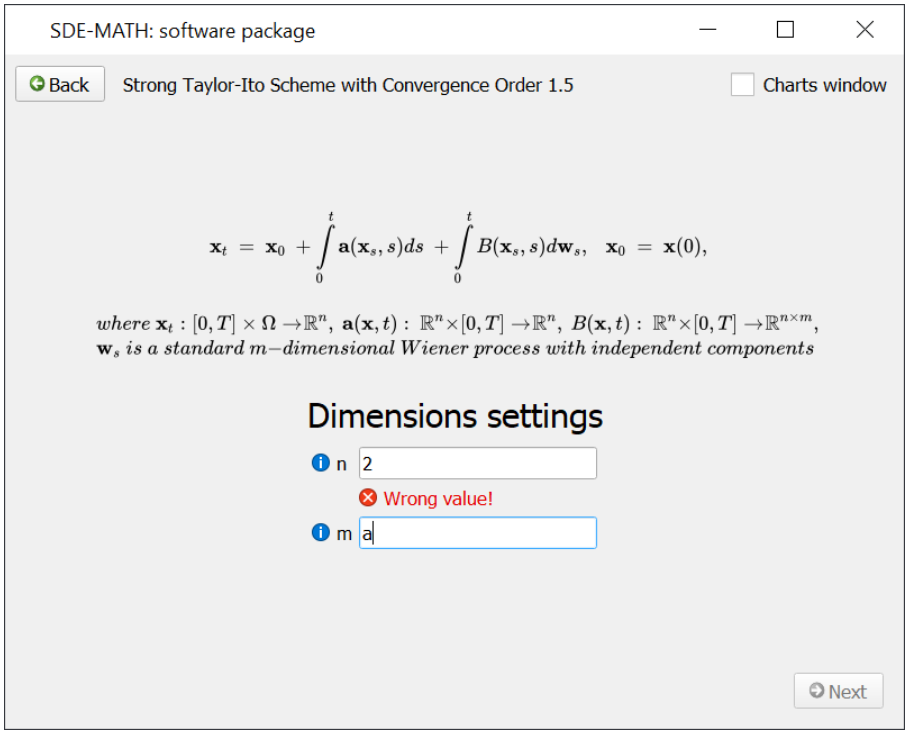


Figure 10: Wrong data input

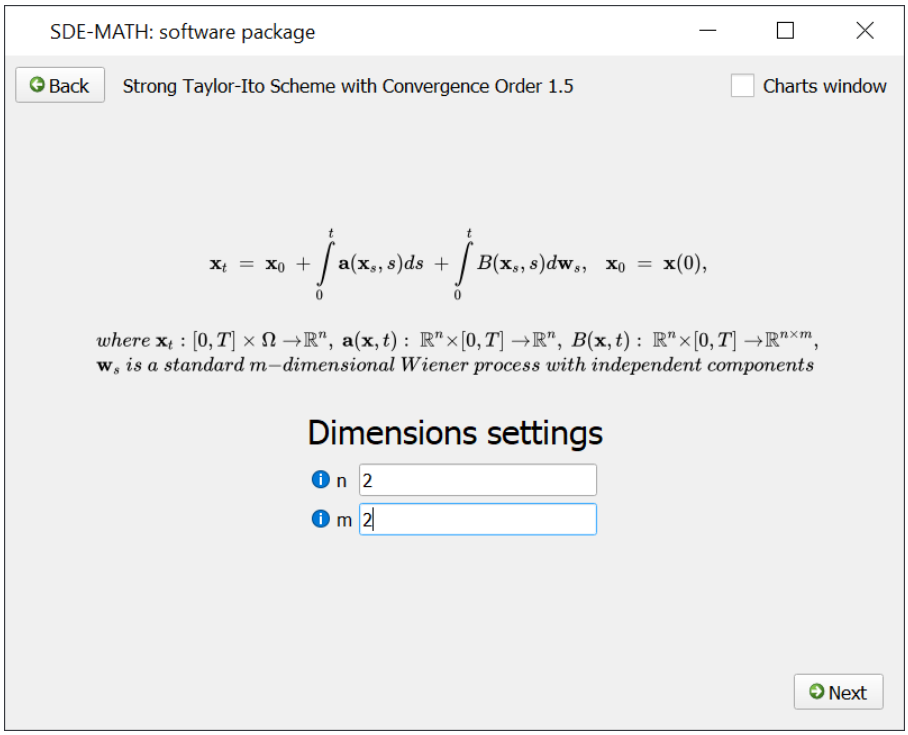


Figure 11: Correct data input

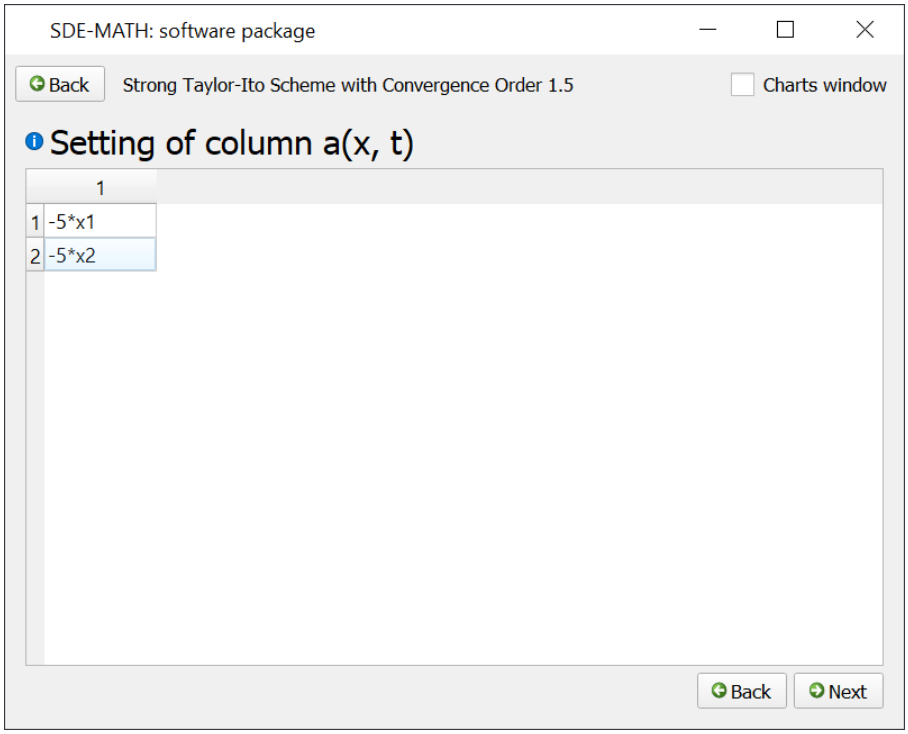


Figure 12: Vector function $\mathbf{a}(\mathbf{x}, t)$ input

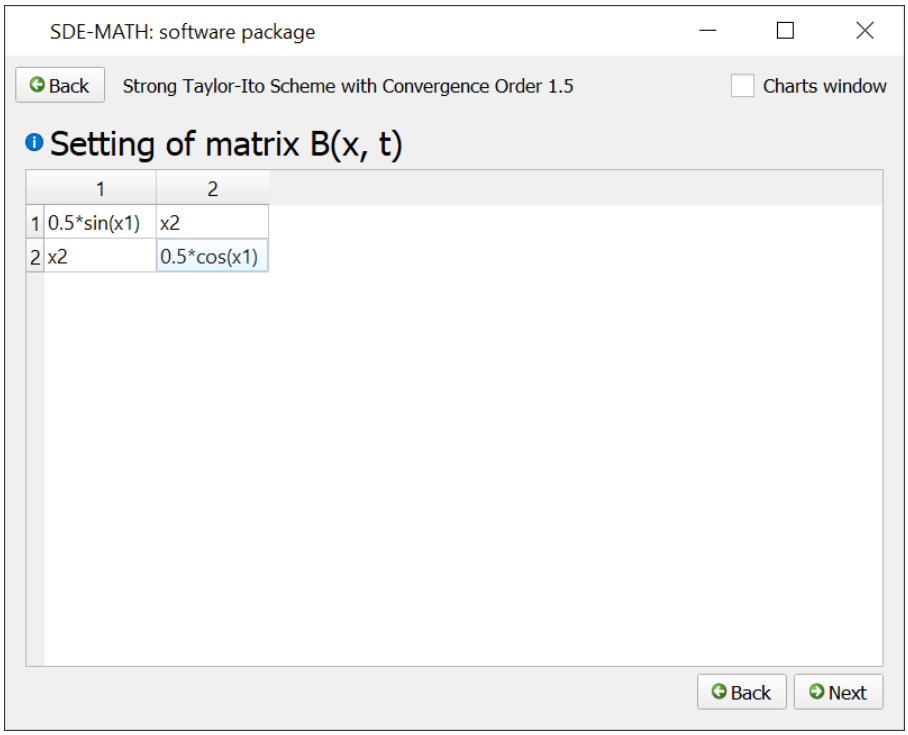


Figure 13: Matrix function $B(\mathbf{x}, t)$ input

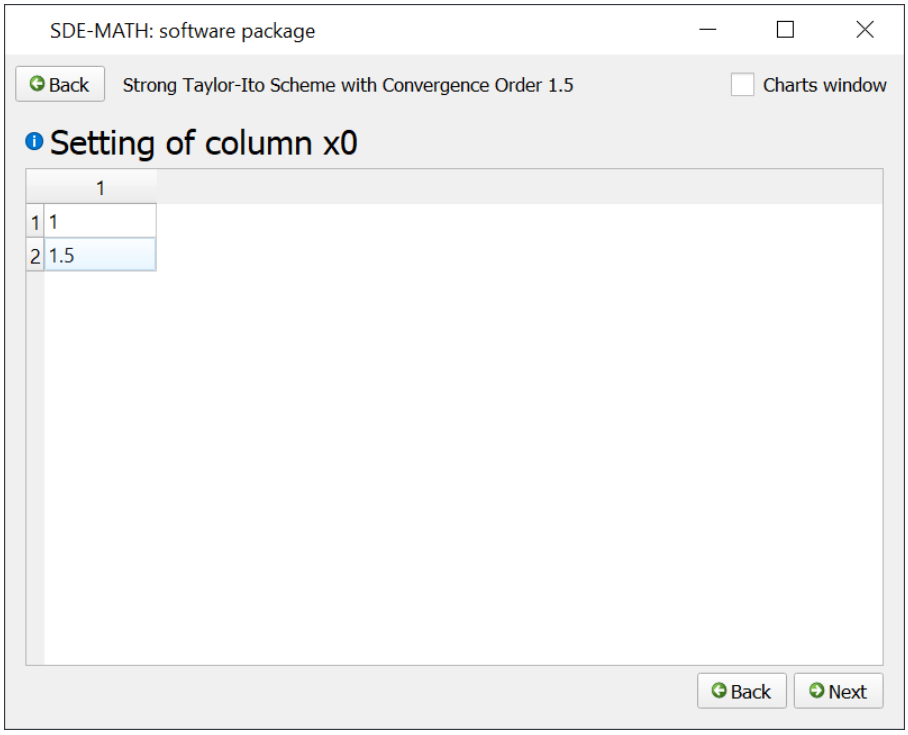


Figure 14: Initial data input

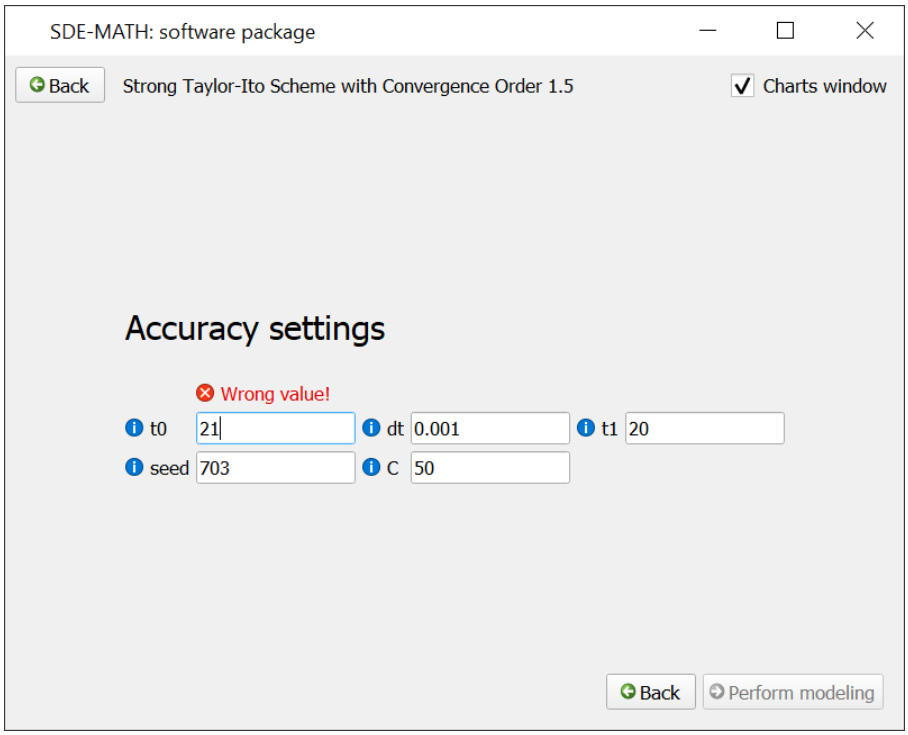


Figure 15: Wrong data input

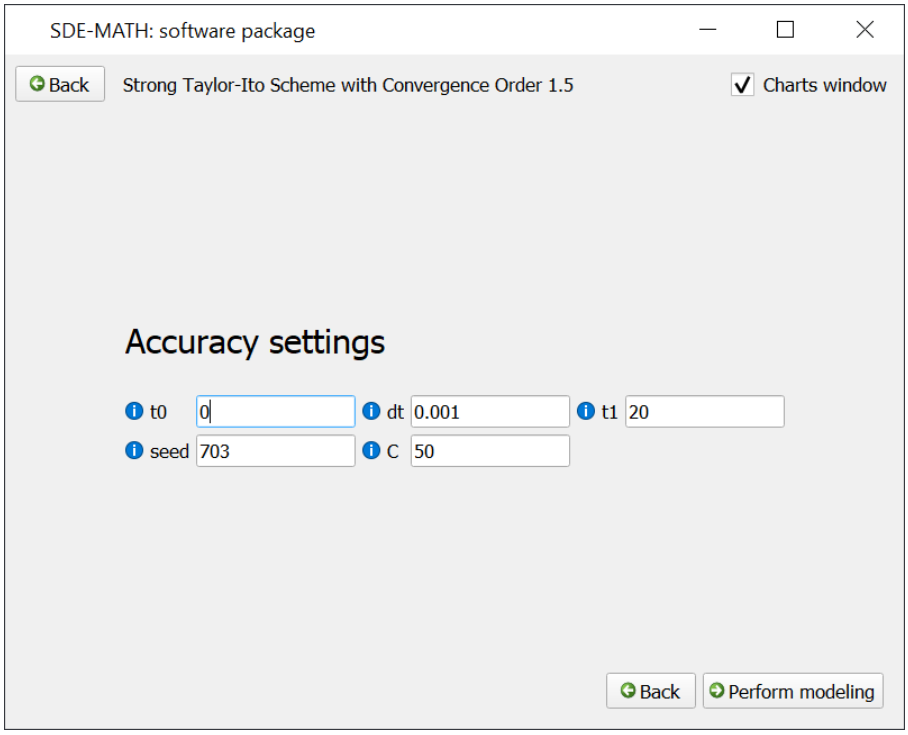


Figure 16: Correct data input

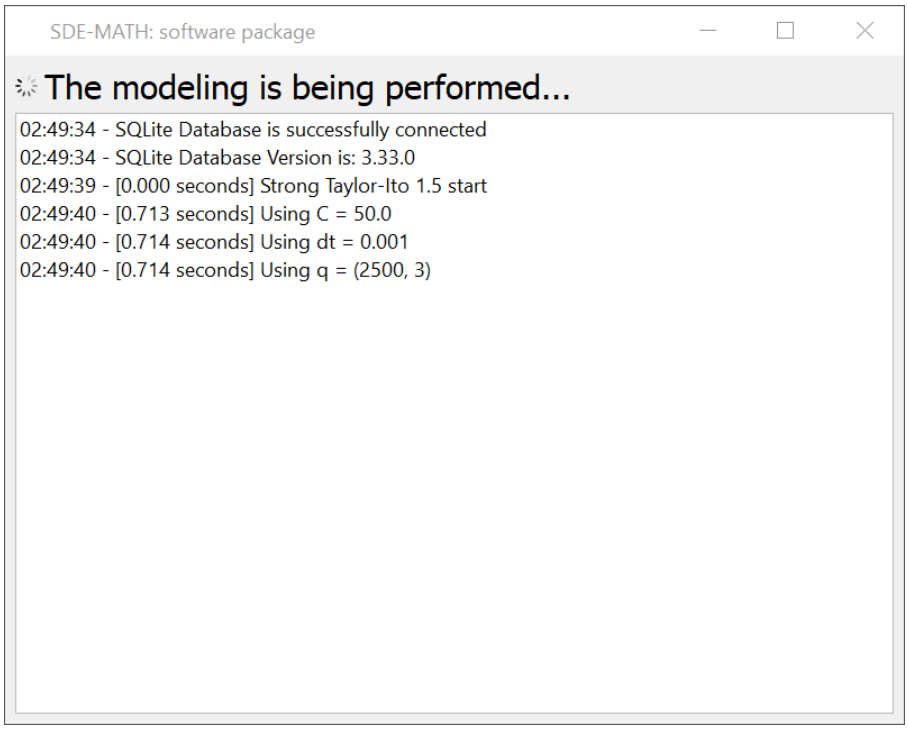


Figure 17: Modeling logs

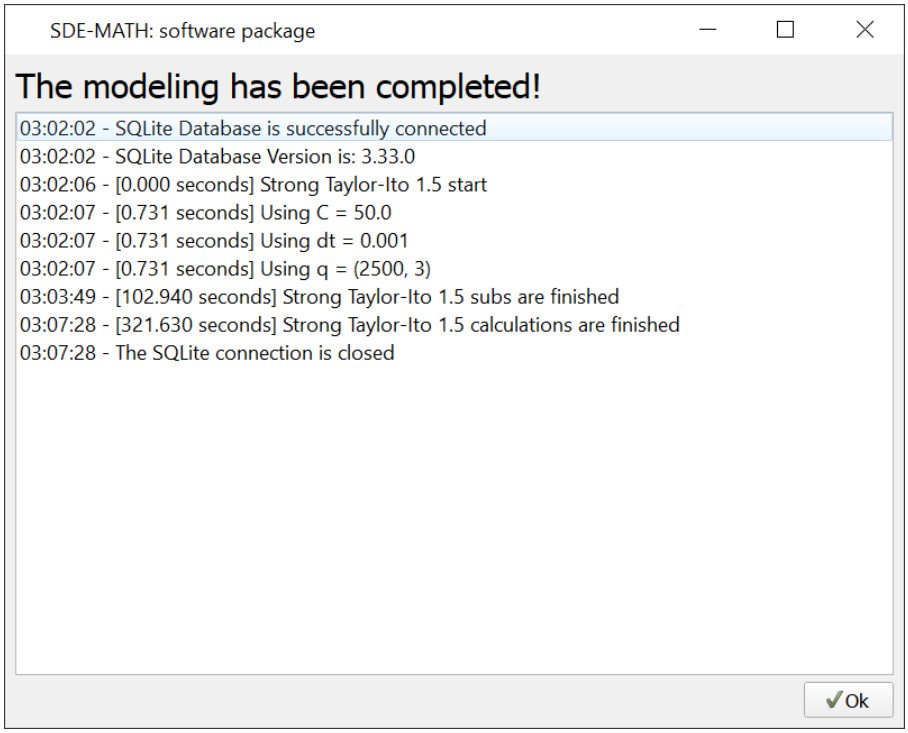


Figure 18: Modeling logs

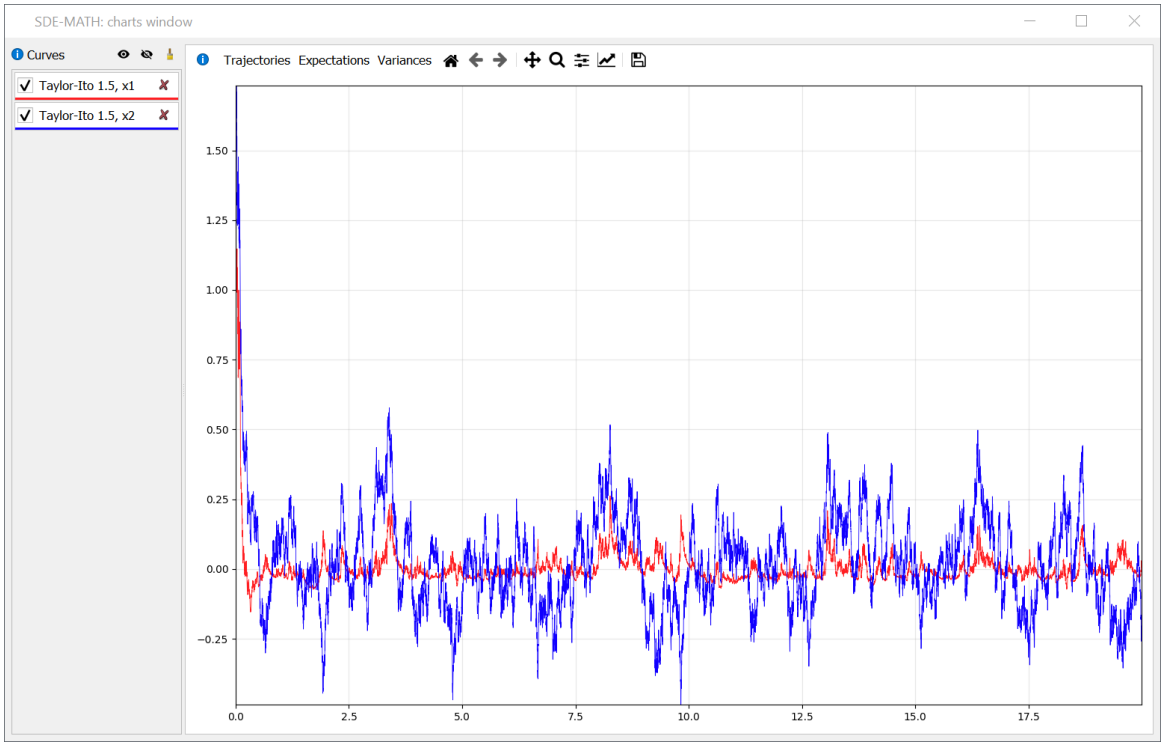


Figure 19: Modeling results

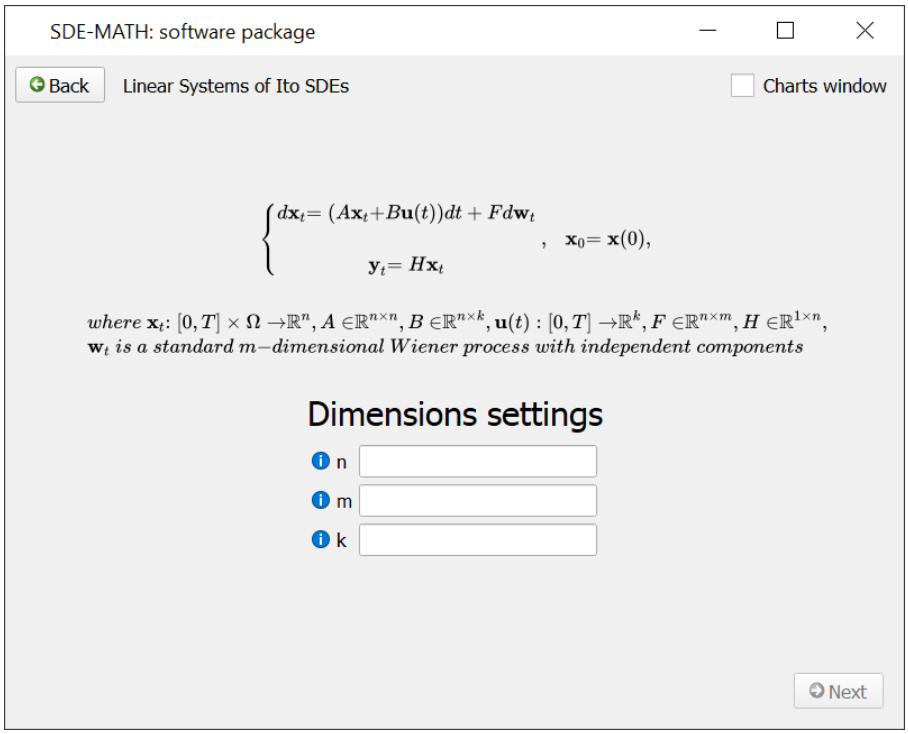


Figure 20: Linear system of Itô SDEs data input

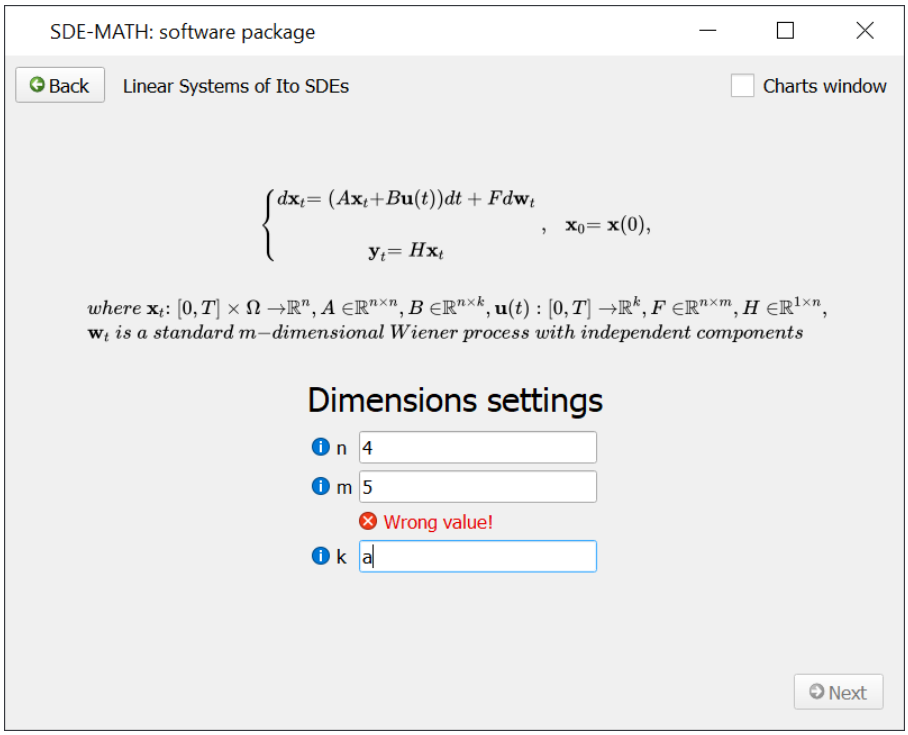


Figure 21: Wrong data input

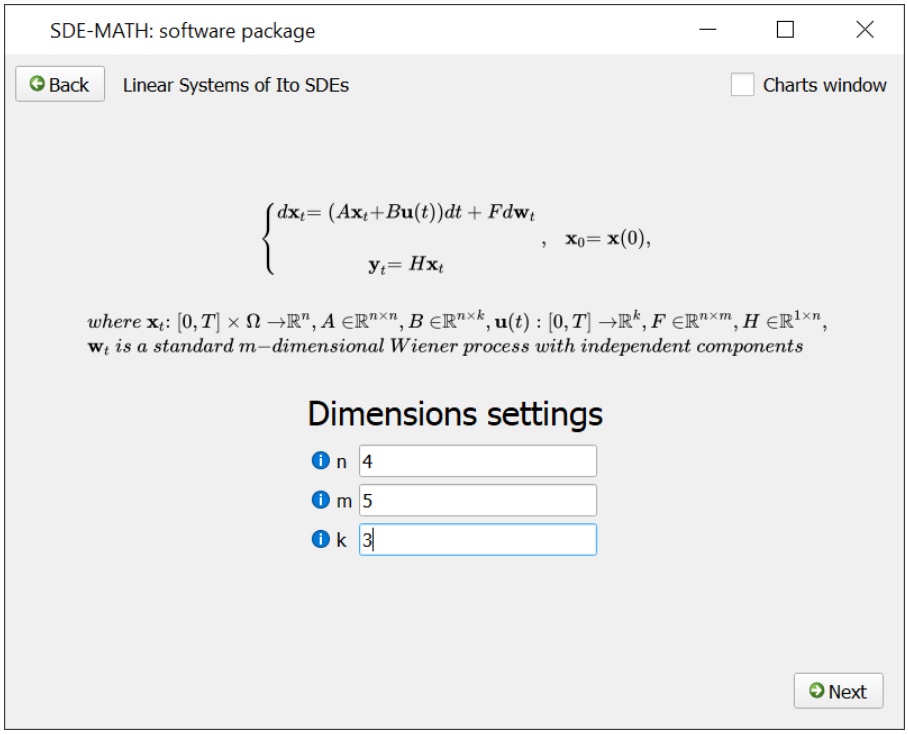


Figure 22: Correct data input

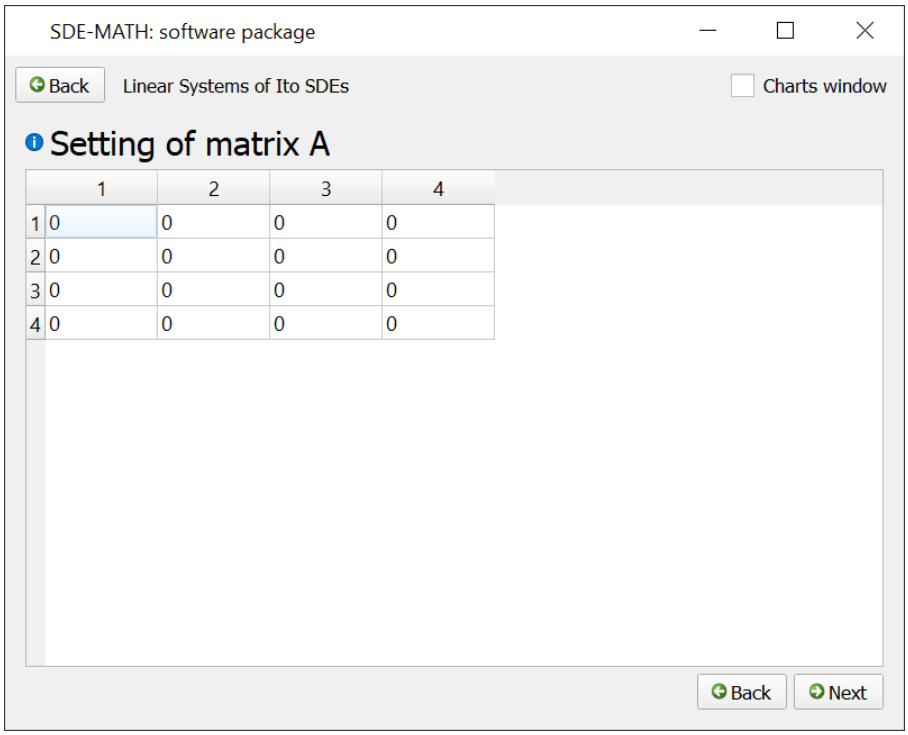


Figure 23: Matrix A input

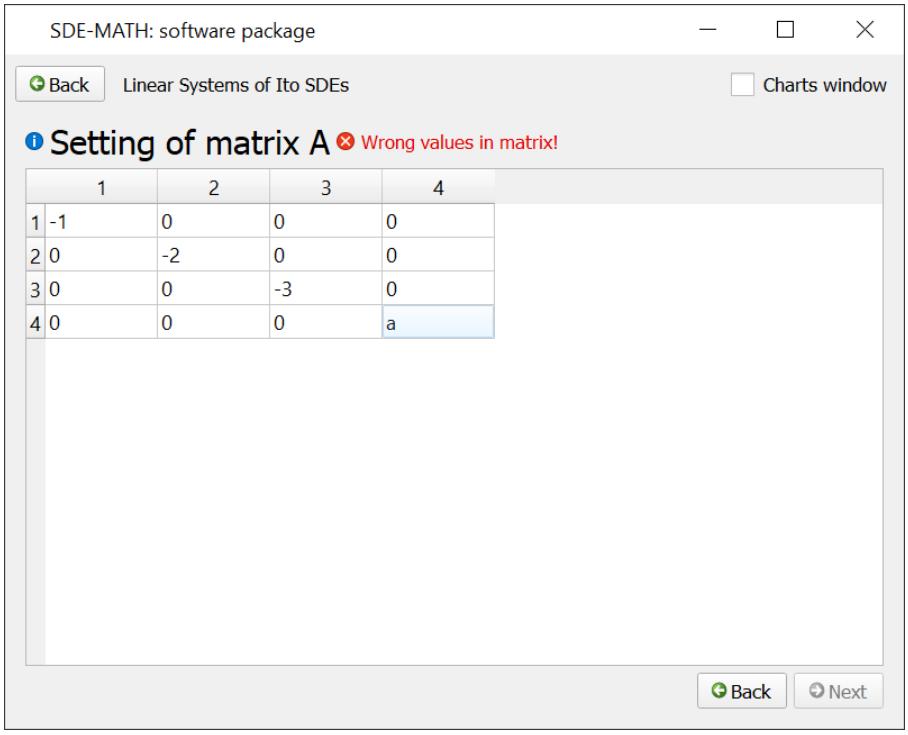


Figure 24: Wrong matrix A input

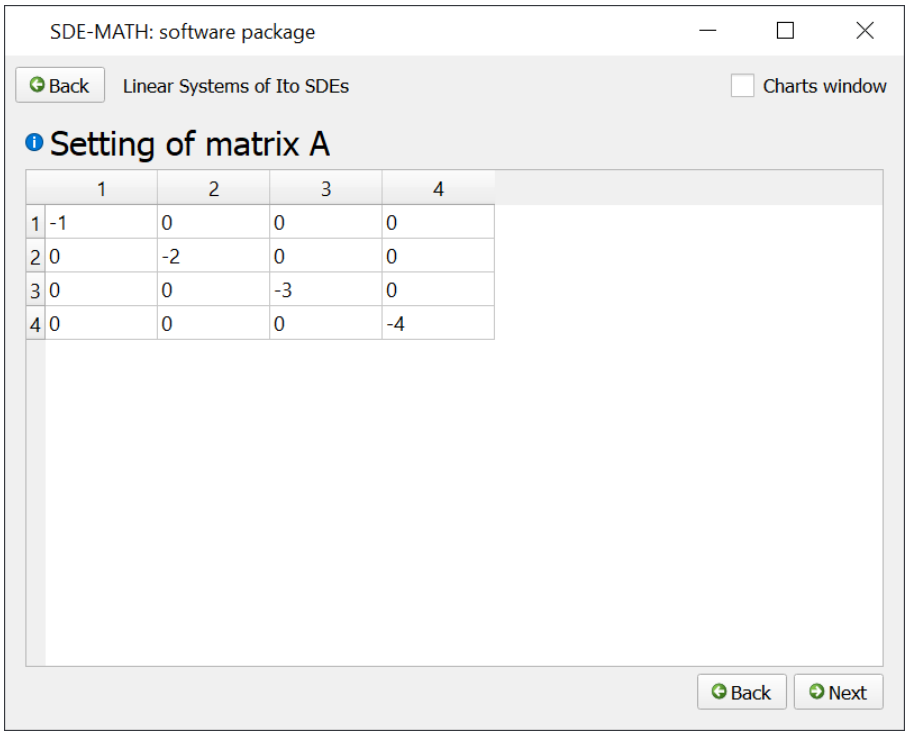


Figure 25: Correct matrix A input

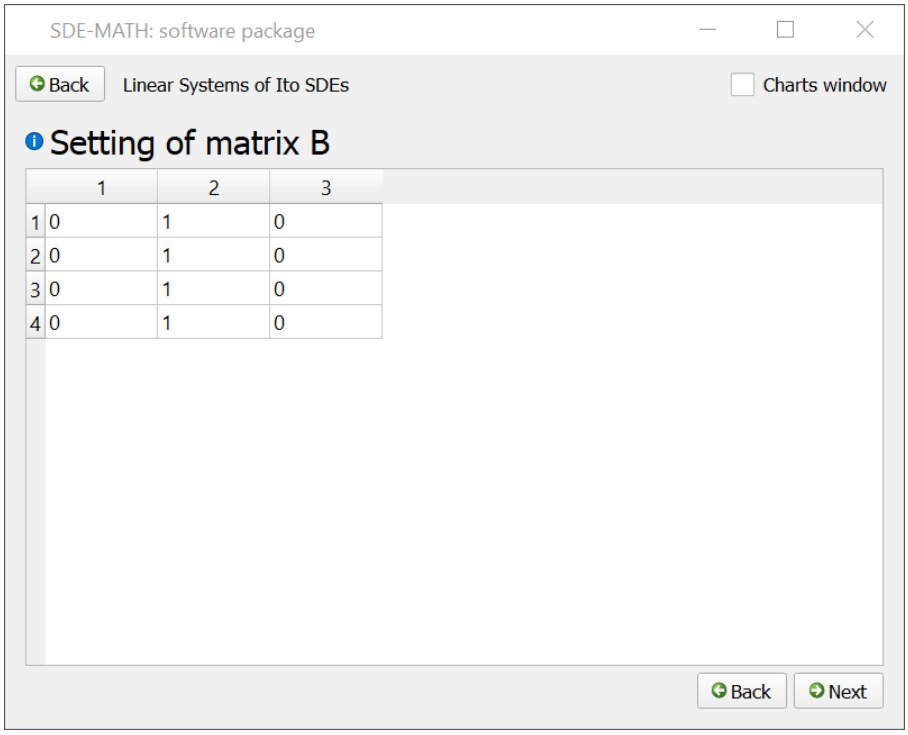


Figure 26: Matrix B input

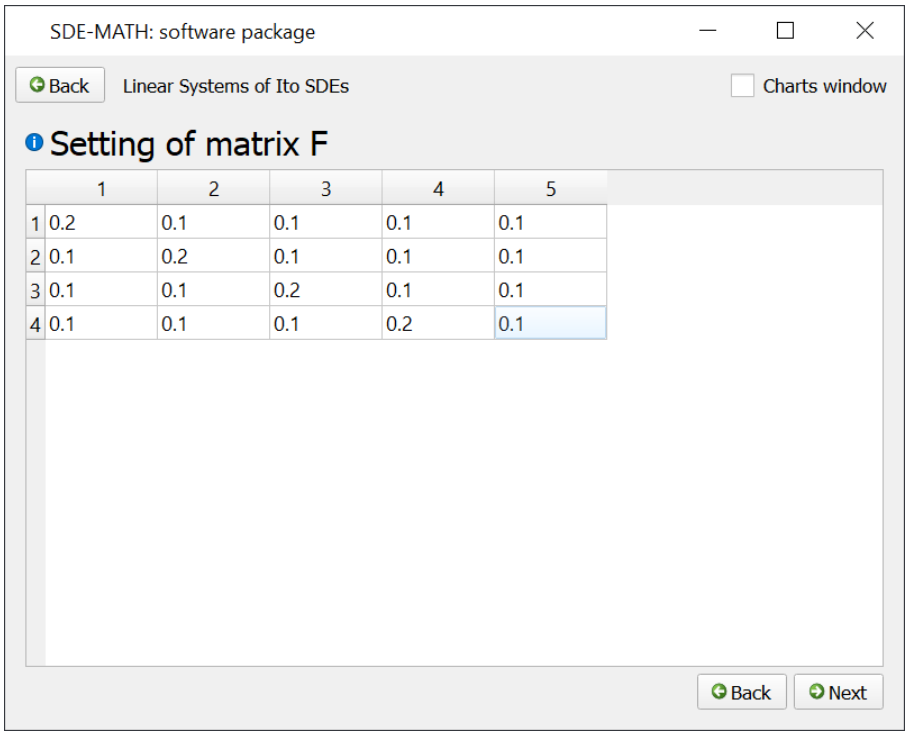


Figure 27: Matrix F input

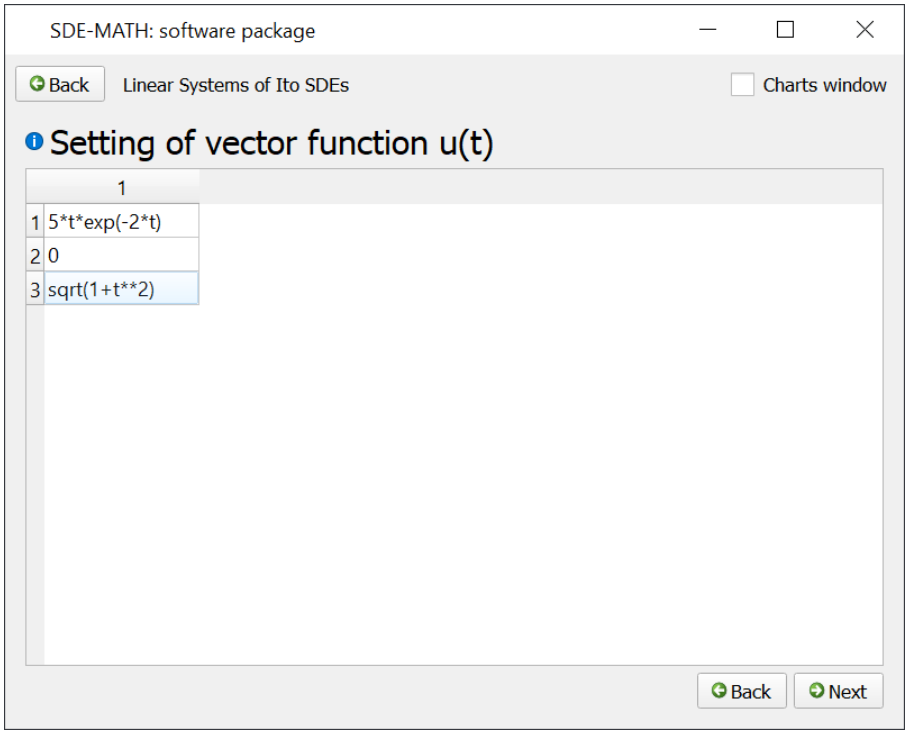


Figure 28: Vector function $\mathbf{u}(t)$ input

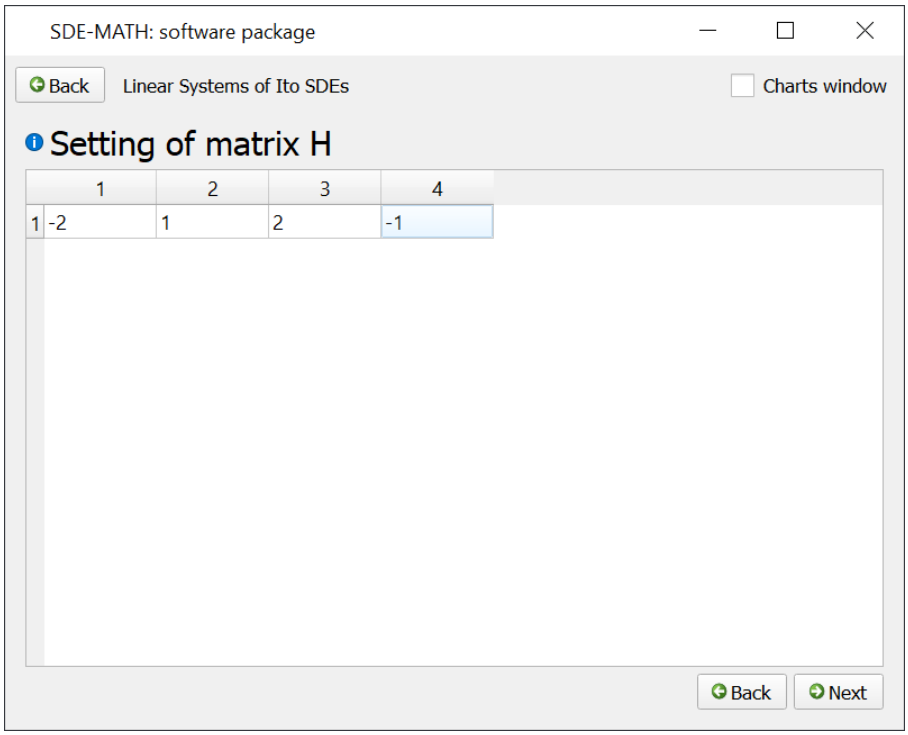


Figure 29: Matrix H input

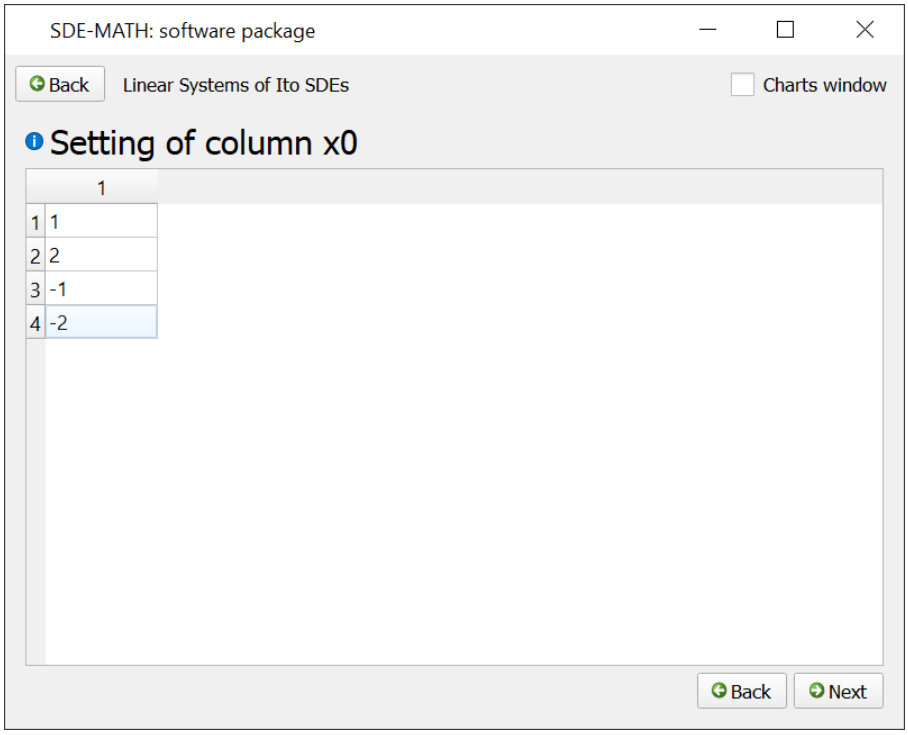


Figure 30: Initial data input

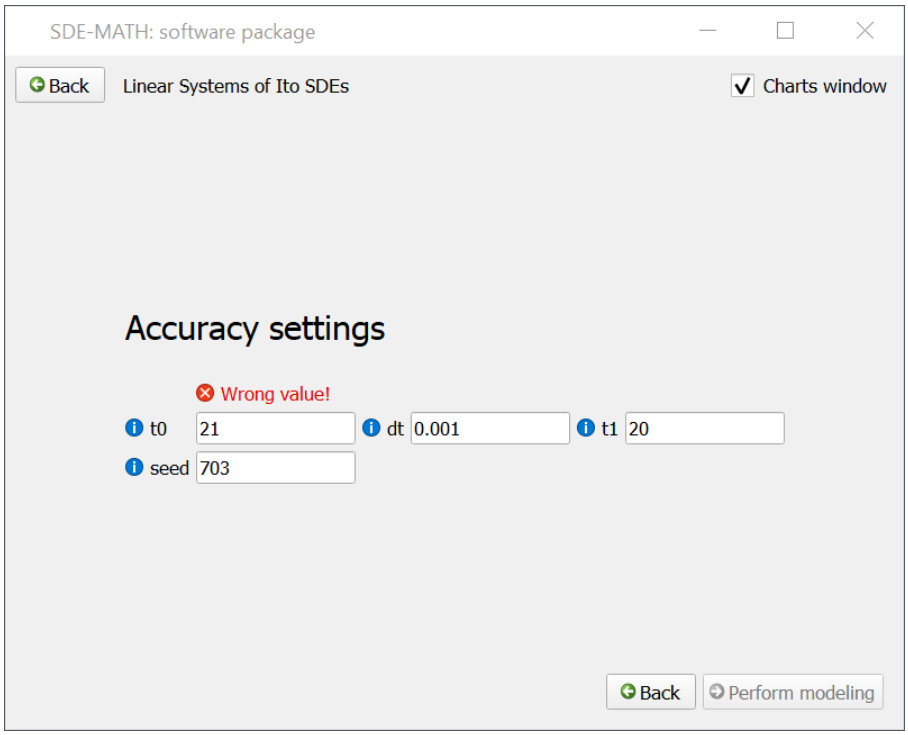


Figure 31: Wrong data input

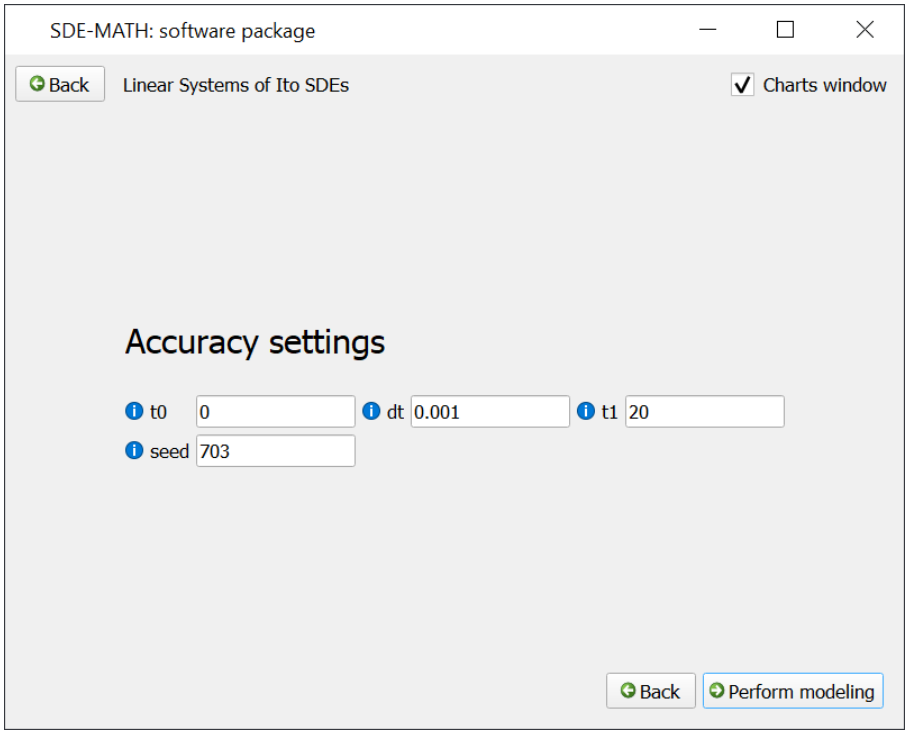


Figure 32: Correct data input

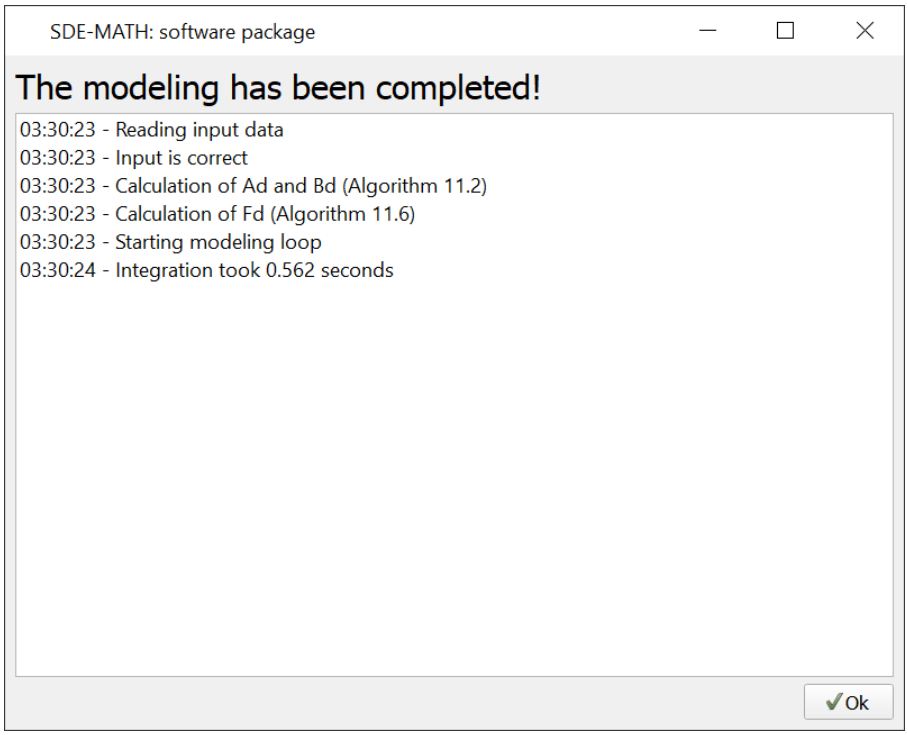


Figure 33: Modeling logs

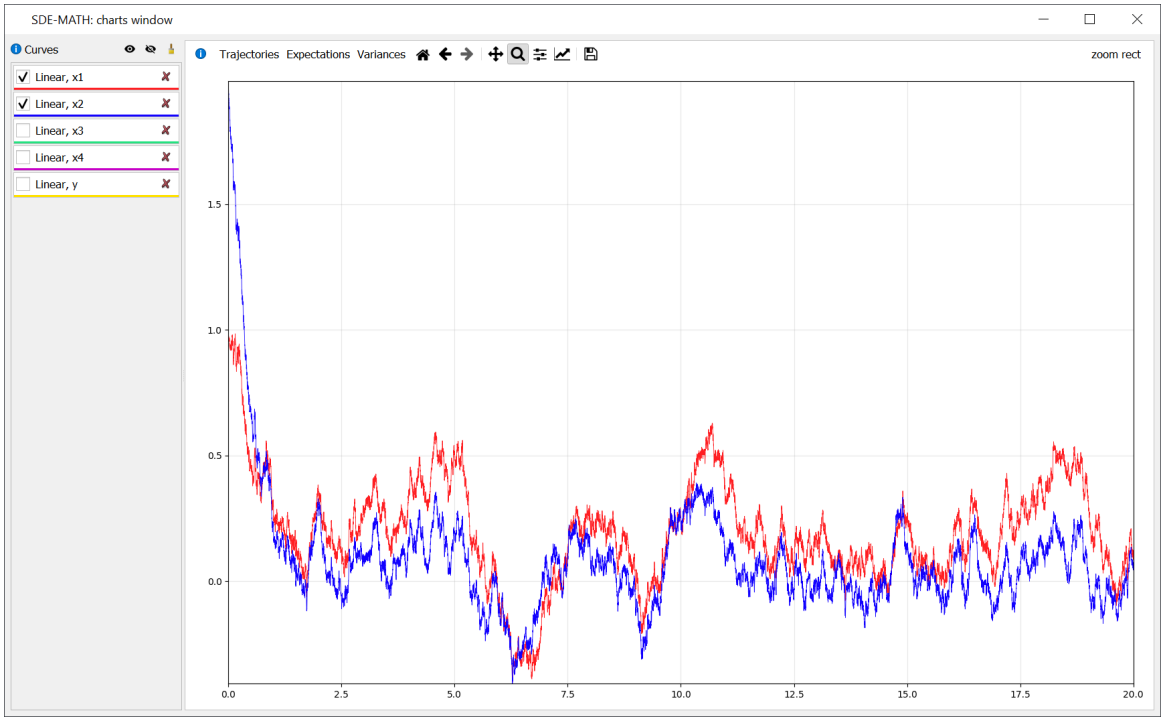


Figure 34: Modeling results (components of solution)

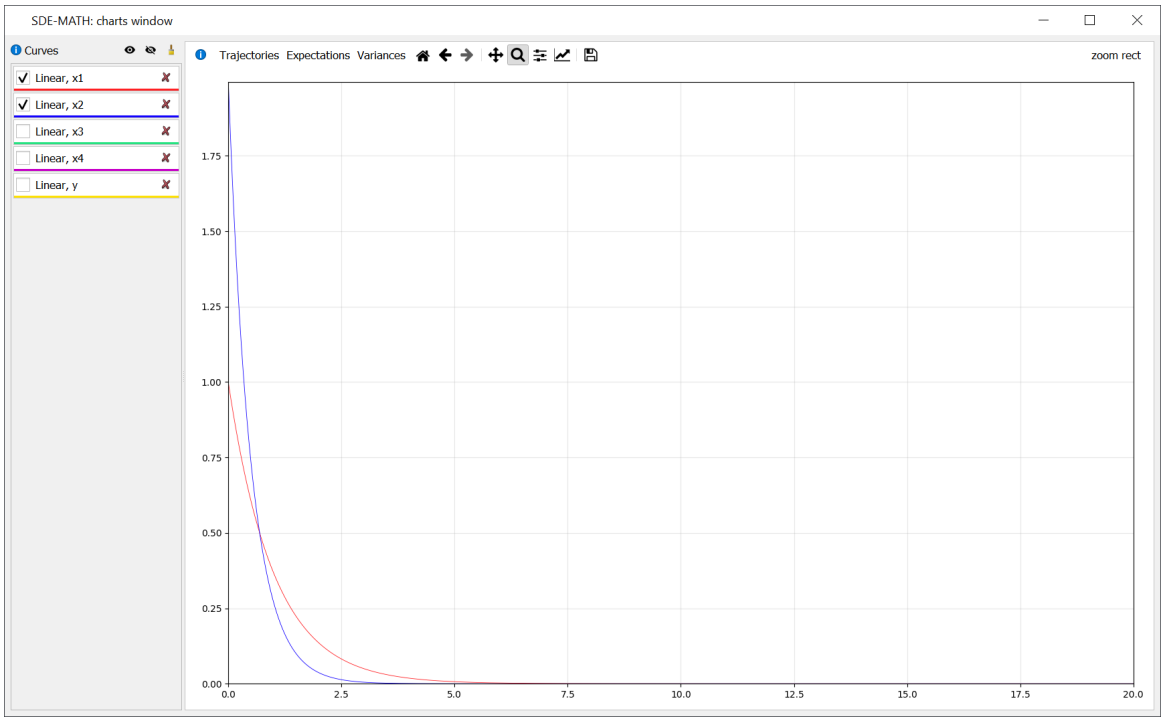


Figure 35: Modeling results (expectations)

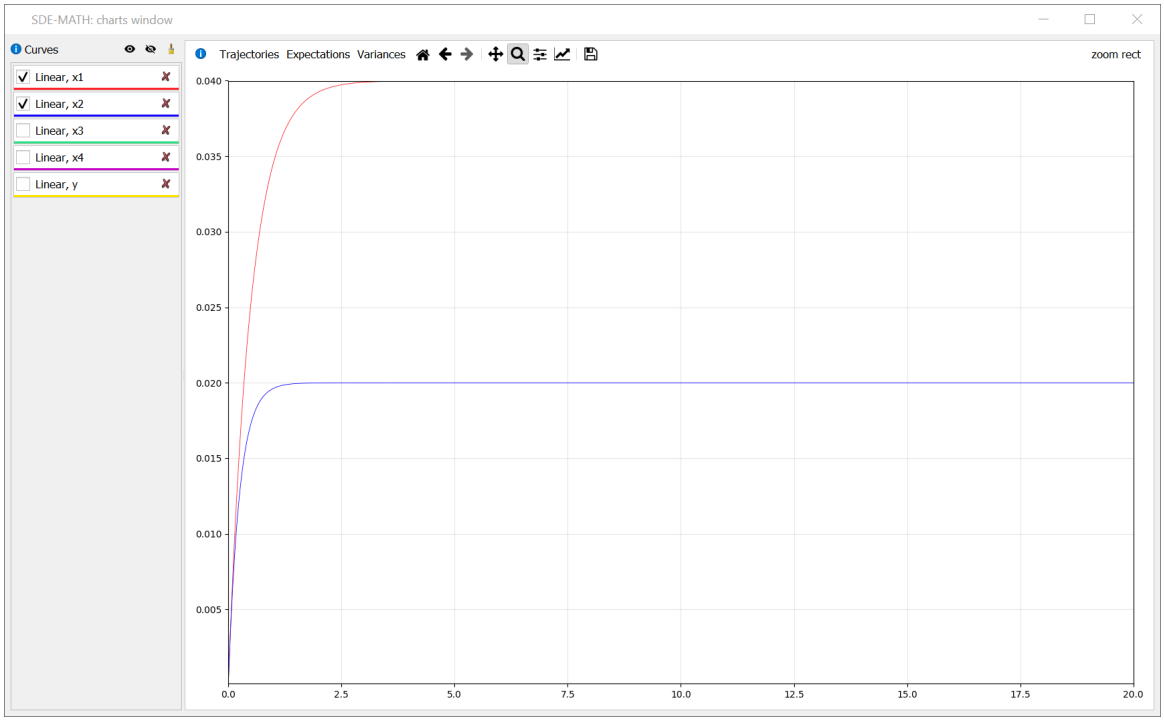


Figure 36: Modeling results (variances)

5 The Results Obtained Using the SDE-MATH Software Package

This section represents the results that were obtained with the SDE-MATH software package at the current stage of the development.

5.1 The Calculated Fourier–Legendre Coefficients

When application runs first time it performs loading of Fourier–Legendre coefficients basic pack in the database from the files. Further, in Listings 1–4 few examples of them can be seen.

Listing 1: **The Fourier–Legendre coefficients $C^{000}_{j_3j_2j_1}$ examples**

```
C_0:0:0 = 4/3
C_0:0:1 = −2/3
C_0:0:2 = 2/15
C_0:0:3 = 0
...
C_0:6:4 = −4/429
C_0:6:5 = 2/143
C_0:6:6 = 2/2145
C_1:0:0 = 2/3
...
C_47:33:44 = 3874457388633368/313349483077359067106604 85
C_47:33:45 = 0
C_47:33:46 = 52892292737827468/22247813298492492376456894435
C_47:34:0 = 0
C_47:34:1 = 0
```

Listing 2: **The Fourier–Legendre coefficients $C^{0000}_{j_4j_3j_2j_1}$ examples**

```
C_0:0:0:0 = 2/3
C_0:0:0:1 = −2/5
C_0:0:0:2 = 2/15
C_0:0:1:0 = −2/15
C_0:0:1:1 = 2/15
...
C_1:1:0:1 = −2/35
C_1:1:0:2 = 0
C_1:1:1:0 = 2/105
C_1:1:1:1 = 0
...
C_20:20:20:1 = −2401828/165607444685315115
C_20:20:20:2 = 0
C_20:20:20:3 = −1241929832/77669891557412788935
C_20:20:20:4 = 0
```

Listing 3: **The Fourier–Legendre coefficients $C^{00000}_{j_5j_4j_3j_2j_1}$ examples**

```
C_0:0:0:0:0 = 4/15
C_0:0:0:0:1 = −8/45
C_0:0:0:1:0 = −4/45
C_0:0:0:1:1 = 8/105
C_0:0:1:0:0 = 0
C_0:0:1:0:1 = 4/315
...
C_1:1:0:1:0 = −4/315
C_1:1:0:1:1 = 4/315
C_1:1:1:0:0 = 2/105
C_1:1:1:0:1 = −8/945
C_1:1:1:1:0 = 2/945
C_1:1:1:1:1 = 0
...
C_20:20:20:20:1 = 0
C_20:20:20:20:2 = −24987720702361001010969028984480026752856704371
C_20:20:20:20:3 = 0
C_20:20:20:20:4 = −307937246954575016/571102494076952592887484313076115
```

Listing 4: **The Fourier–Legendre coefficients $C^{000000}_{j_6j_5j_4j_3j_2j_1}$ examples**

```
C_0:0:0:0:0:0 = 4/45
C_0:0:0:0:0:1 = −4/63
C_0:0:0:0:0:2 = 2/63
C_0:0:0:0:1:0 = −4/105
...
C_2:1:0:1:0:2 = −2/1575
C_2:1:0:1:1:0 = 38/22275
C_2:1:0:1:1:1 = −2/1575
C_2:1:0:1:1:2 = 68/81081
...
C_15:15:15:15:15:15 = 0
C_15:15:15:15:15:16 = −798538765964/24307635224228051171391378347 5
C_15:15:15:15:15:17 = 0
C_15:15:15:15:15:18 = −59075427603328/173026167096096036974544044769175
```

5.2 Accuracy Settings

From Theorem 3 (see formulas (51)–(66)) it follows that the number p in the formula (48) should be chosen individually for various combinations of indices $i_1, \ldots, i_k \in \{1, \ldots, m\}$. As follows from Listing 5 (see below) and the results of work [73], these numbers p in the overwhelming majority of cases do not exceed the number p from the formula (51). Moreover, all the mentioned numbers p are many times less than the number p selected using the formula (49) (due to the presence of the multiplier factor $k!$ on the left-hand side of (49)).

In this work, we have replaced the mentioned numbers p for all possible

combinations of indices $i_1, \ldots, i_k \in \{1, \ldots, m\}$ with the number p according to the formula (51). This is possible due to the results of Listing 6. This listing shows that the above replacement does not lead to noticeable accuracy loss of the mean square approximation of iterated Itô stochastic integrals (for more details see [73]).

Thus, in this paper we decided to exclude the multiplier factor $k!$ in the conditions for choosing the numbers $q_1, \ldots, q_{15}$ (see (169)–(202)). Recall that these numbers are used to construct the approximations of iterated Itô and Stratonovich stochastic integrals from the numerical schemes (13)–(16), (25)–(28). The test script was written. The results of its work are presented in Listings 5 and 6, where

1. dt is the integration step;
2. q1(1,2) means p from (54), q1(2,3) means p from (55), q1(1,3) means p from (56), q1 means p from (51) for $k = 3$;
3. $C = 1$ (see (17) and (29));
4. error 1 means the left-hand side of (171);
5. error 2 means the left-hand side of (54) divided by $(T - t)^3$;
6. error 3 means the left-hand side of (56) divided by $(T - t)^3$;
7. error 4 means the left-hand side of (55) divided by $(T - t)^3$.

The above idea of calculation of the numbers $q_1, \ldots, q_{15}$ is described in Listing 109.

Listing 5: **Accuracy calculation module**

```
1
2  dt = 0.011
3    q1    = 12
4    q1 (1, 2) = 6
5    q1 (1, 3) = 12
6    q1 (2, 3) = 6
7
8  dt = 0.008
9    q1    = 16
10   q1 (1, 2) = 8
11   q1 (1, 3) = 16
12   q1 (2, 3) = 8
```

```
13
14 dt = 0.0045
15   q1    = 28
16   q1 (1, 2) = 14
17   q1 (1, 3) = 28
18   q1 (2, 3) = 14
19
20 dt = 0.0035
21   q1    = 36
22   q1 (1, 2) = 18
23   q1 (1, 3) = 36
24   q1 (2, 3) = 18
25
26 dt = 0.0027
27   q1    = 47
28   q1 (1, 2) = 23
29   q1 (1, 3) = 47
30   q1 (2, 3) = 23
31
32 dt = 0.0025
33   q1    = 50
34   q1 (1, 2) = 25
35   q1 (1, 3) = 51
36   q1 (2, 3) = 25
37
38
39 Process finished with exit code 0
```

Listing 6: **Accuracy calculation module**

```
1
2 dt = 0.011
3   error 1 = 0.010153888451696458
4   q1    = 12
5   error 2 = 0.005076944225848201
6   q1 (1, 2) = 12
7   error 3 = 0.010307776903394072
8   q1 (1, 3) = 12
9   error 4 = 0.005076944225848284
10  q1 (2, 3) = 12
11
12 dt = 0.008
13  error 1 = 0.007681193827577537
14  q1    = 16
15  error 2 = 0.003840596913789046
16  q1 (1, 2) = 16
17  error 3 = 0.0077866300793989485
18  q1 (1, 3) = 16
19  error 4 = 0.003840596913789157
20  q1 (2, 3) = 16
21
22 dt = 0.0045
```

```
  error 1 = 0.004432832059862973
  q1    = 28
  error 2 = 0.0022164160299319446
  q1 (1, 2) = 28
  error 3 = 0.004479699207443705
  q1 (1, 3) = 28
  error 4 = 0.002216416029932139
  q1 (2, 3) = 28

dt = 0.0035
  error 1 = 0.0034564405520411956
  q1    = 36
  error 2 = 0.0017282202760207088
  q1 (1, 2) = 36
  error 3 = 0.003488223569838411
  q1 (1, 3) = 36
  error 4 = 0.0017282202760210141
  q1 (2, 3) = 36

dt = 0.0027
  error 1 = 0.0026523659377455377
  q1    = 47
  error 2 = 0.00132618296887127
  q1 (1, 2) = 47
  error 3 = 0.0026731529281250332
  q1 (1, 3) = 47
  error 4 = 0.0013261829688714366
  q1 (2, 3) = 47

dt = 0.0025
  error 1 = 0.002494053620431952
  q1    = 50
  error 2 = 0.0012470268102122428
  q1 (1, 2) = 50
  error 3 = 0.0025128597161119537
  q1 (1, 3) = 50
  error 4 = 0.0012470268102123538
  q1 (2, 3) = 50


Process finished with exit code 0
```

5.3 Testing Example (Nonlinear System of Itô SDEs)

The input data for testing of the SDE-MATH software package correspond to the autonomous variant of nonlinear system of Itô SDE (1) with multidimensional non-commutative noise. More precisely, we choose $n = 2$, $m = 2$, $\mathbf{x}_0^{(1)} = 1$, $\mathbf{x}_0^{(2)} = 1.5$,

$$\mathbf{a}\left(\mathbf{x}^{(1)},\mathbf{x}^{(2)}\right)=\begin{pmatrix}-5\mathbf{x}^{(1)}\\ \\ -5\mathbf{x}^{(2)}\end{pmatrix},$$

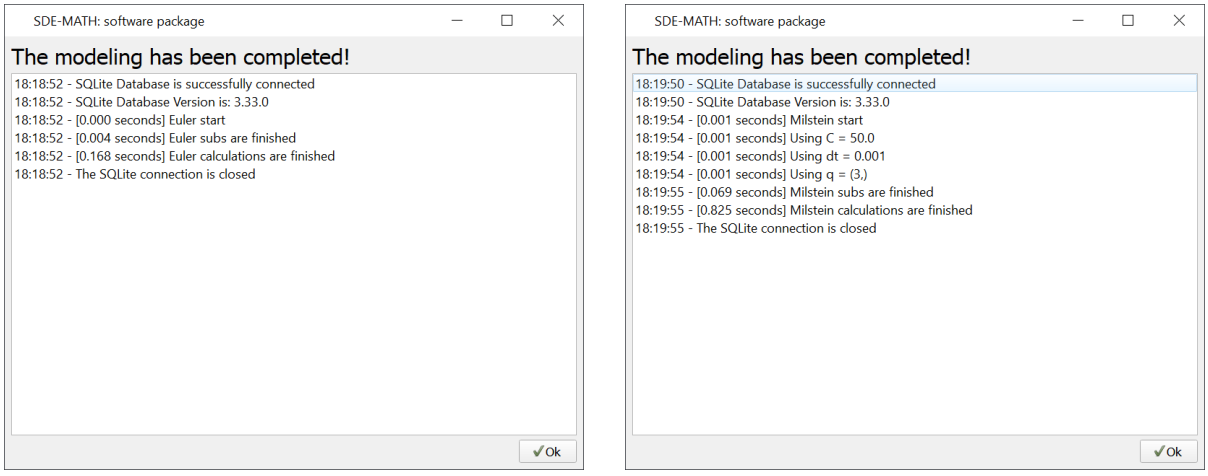
$$B\left(\mathbf{x}^{(1)},\mathbf{x}^{(2)}\right)=\begin{pmatrix}0.5\cdot\sin\left(\mathbf{x}^{(1)}\right) & \mathbf{x}^{(2)}\\ \\ \mathbf{x}^{(2)} & 0.5\cdot\cos\left(\mathbf{x}^{(1)}\right)\end{pmatrix}.$$

Figures 37–92 related to the strong high-order Taylor–Itô and Taylor–Stratonovich schemes (12)–(16), (24)–(28) for the Itô SDE (1) represent modeling results.

Test machine specifications are CPU with maximum core frequency 4.2 GHz and 16GB of RAM.

5.4 Visualization and Numerical Results for Nonlinear System of Itô SDEs Obtained via the SDE-MATH Software Package

This subsection is fully devoted to modeling logs and results visualization. They are presented on Figures 37–91



Euler scheme ($dt = 0.001$) Milstein scheme ($C = 50$, $dt = 0.001$)

Figure 37: Modeling logs

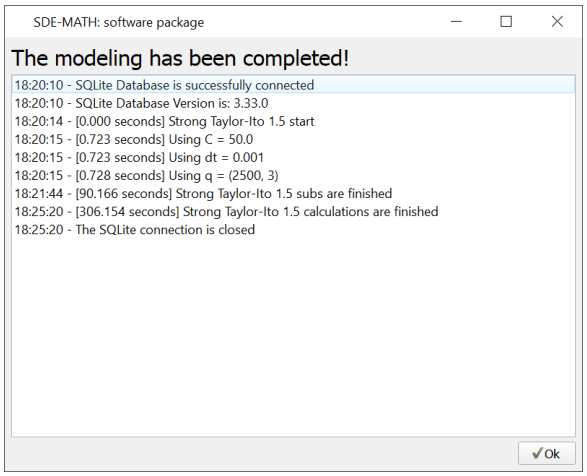


Figure 38: Strong Taylor–Itô scheme of order 1.5 ($C = 50$, $dt = 0.001$)

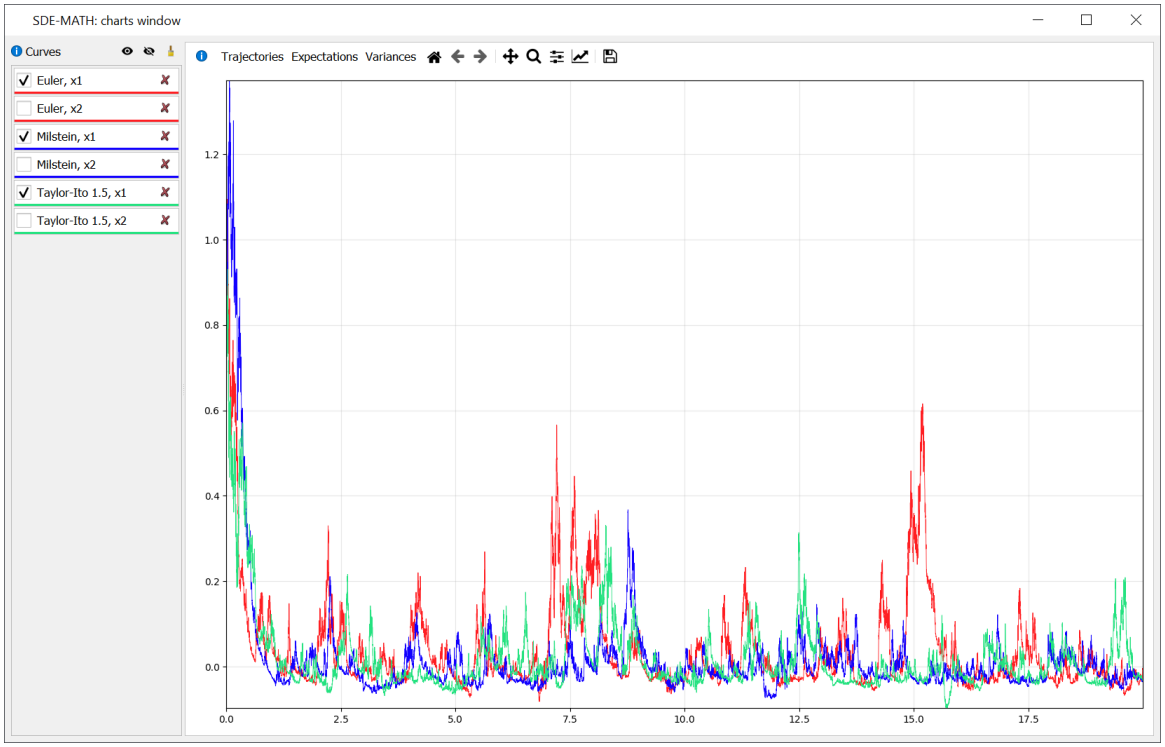


Figure 39: Strong Taylor–Itô schemes of orders 0.5, 1.0, and 1.5 ($\mathbf{x}_t^{(1)}$ component, $C = 50$, $dt = 0.001$)

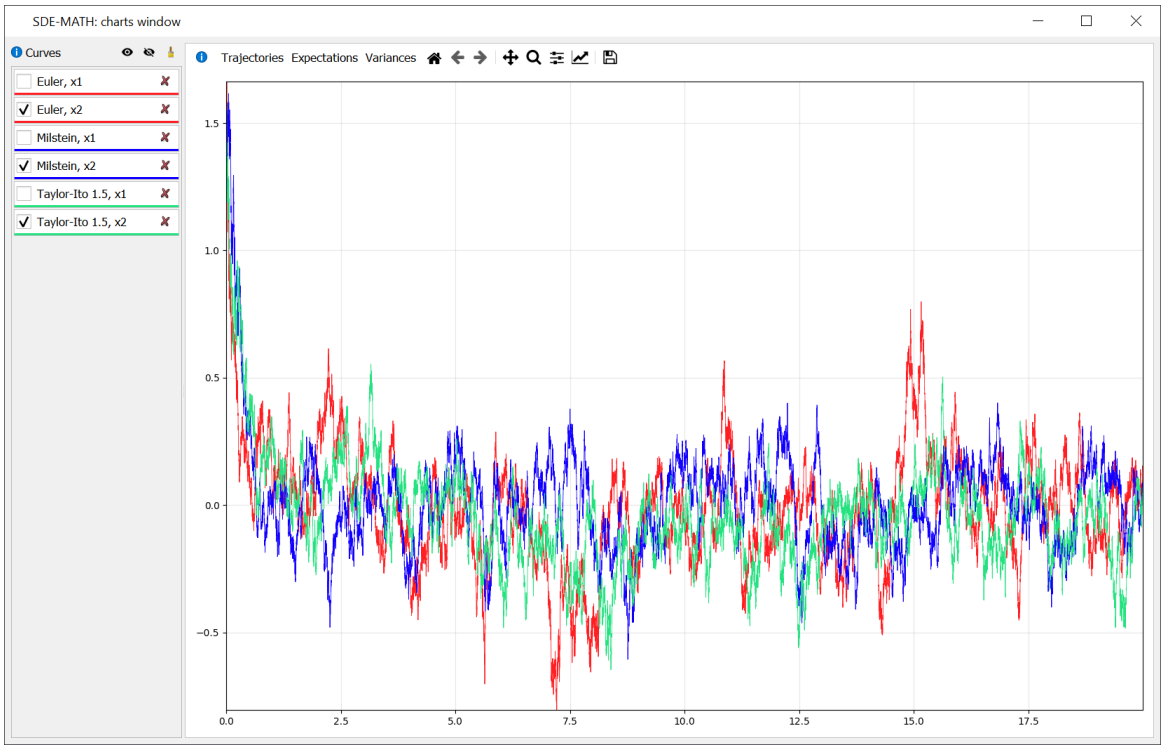
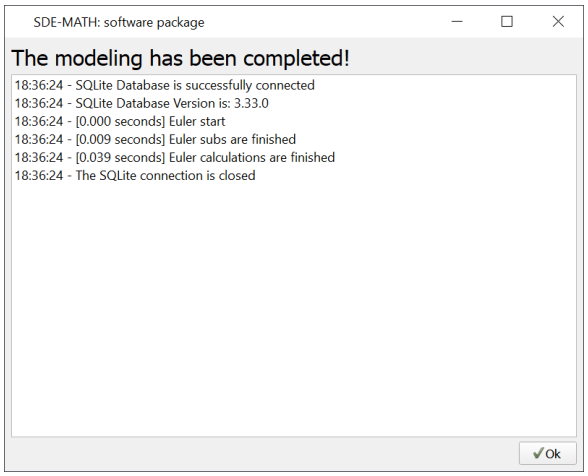
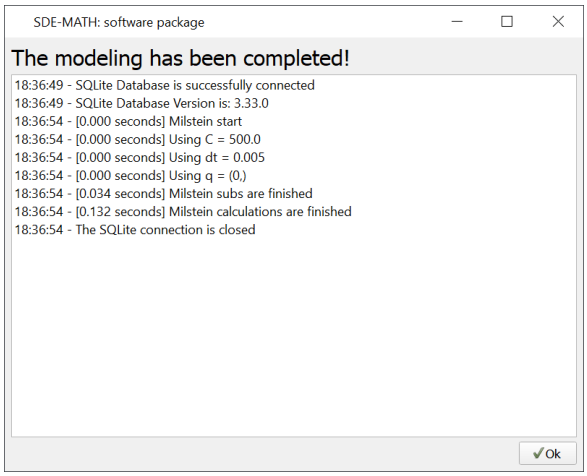


Figure 40: Strong Taylor–Itô schemes of orders 0.5, 1.0, and 1.5 ($\mathbf{x}_t^{(2)}$ component, $C = 50$, $dt = 0.001$)

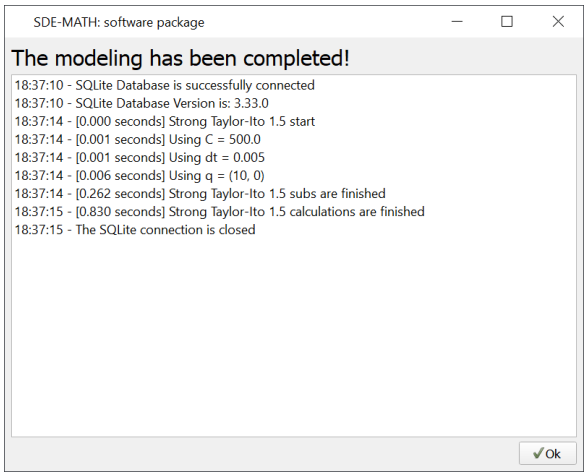


Euler scheme ($dt = 0.005$)

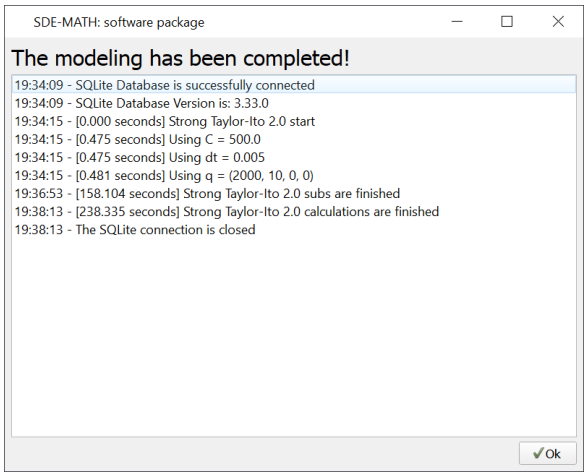


Milstein scheme ($C = 500$, $dt = 0.005$)

Figure 41: Modeling logs



Strong Taylor–Itô scheme of order 1.5 ($C = 500$, $dt = 0.005$)



Strong Taylor–Itô scheme of order 2.0 ($C = 500$, $dt = 0.005$)

Figure 42: Modeling logs

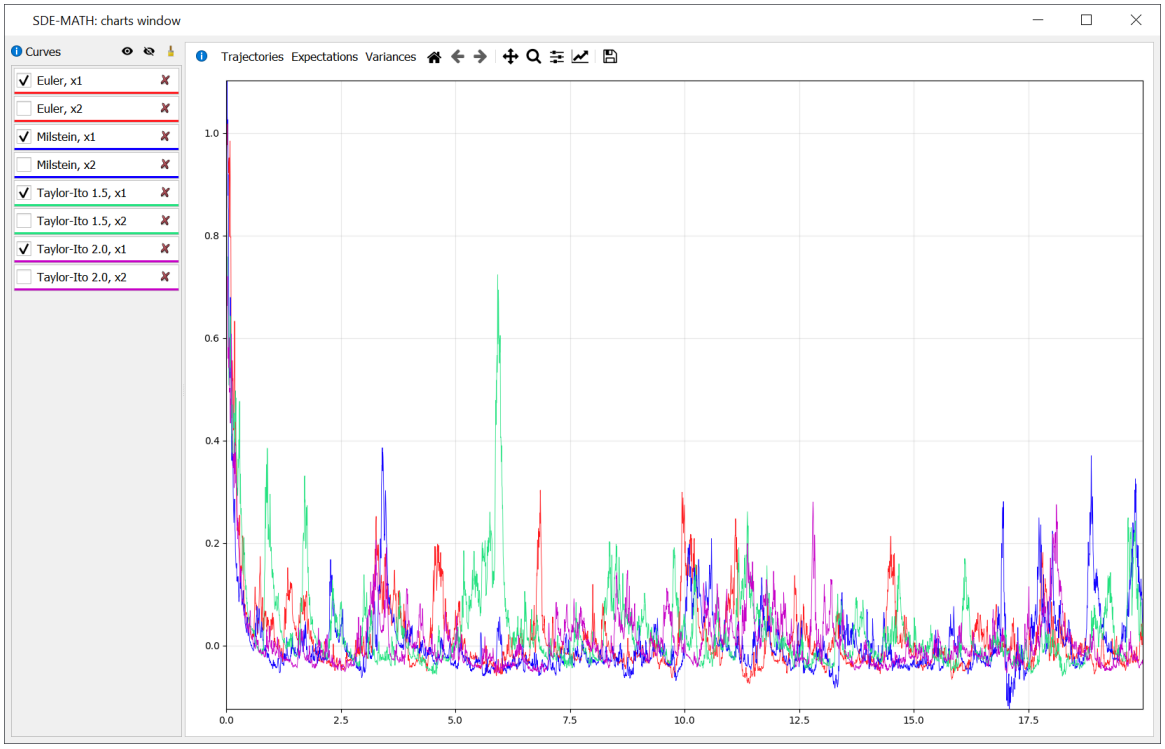


Figure 43: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(1)}$ component, $C = 500$, $dt = 0.005$)

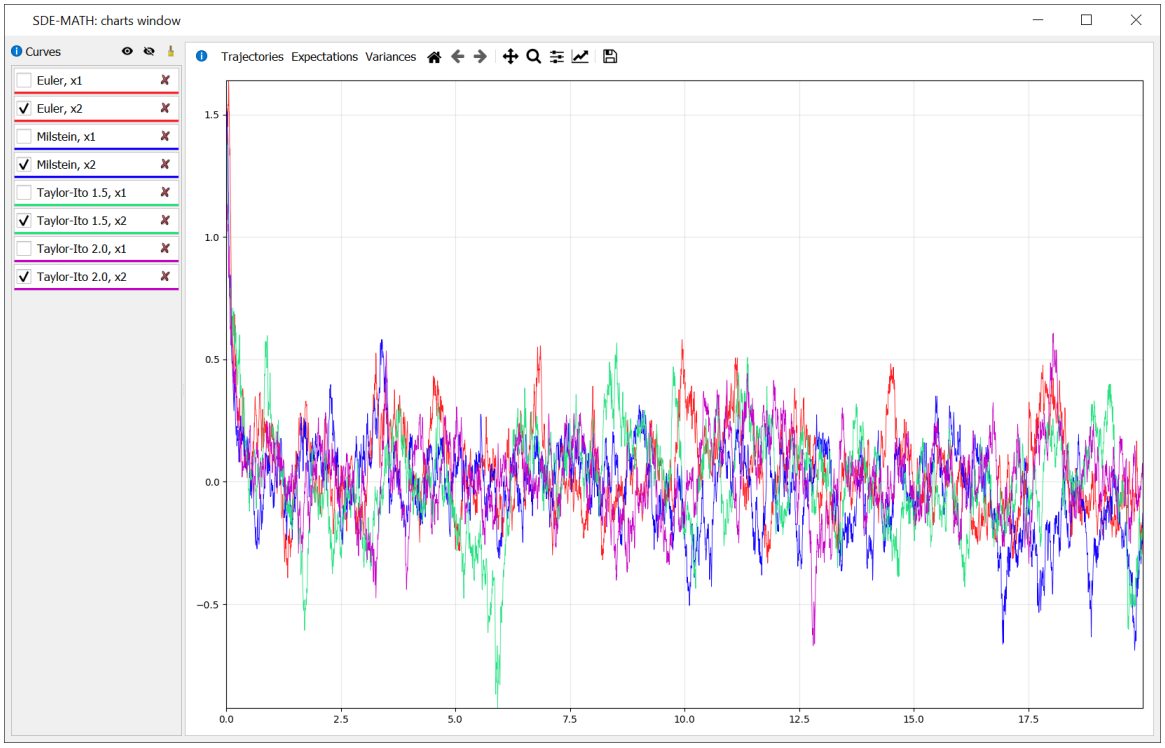


Figure 44: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(2)}$ component, $C = 500$, $dt = 0.005$)

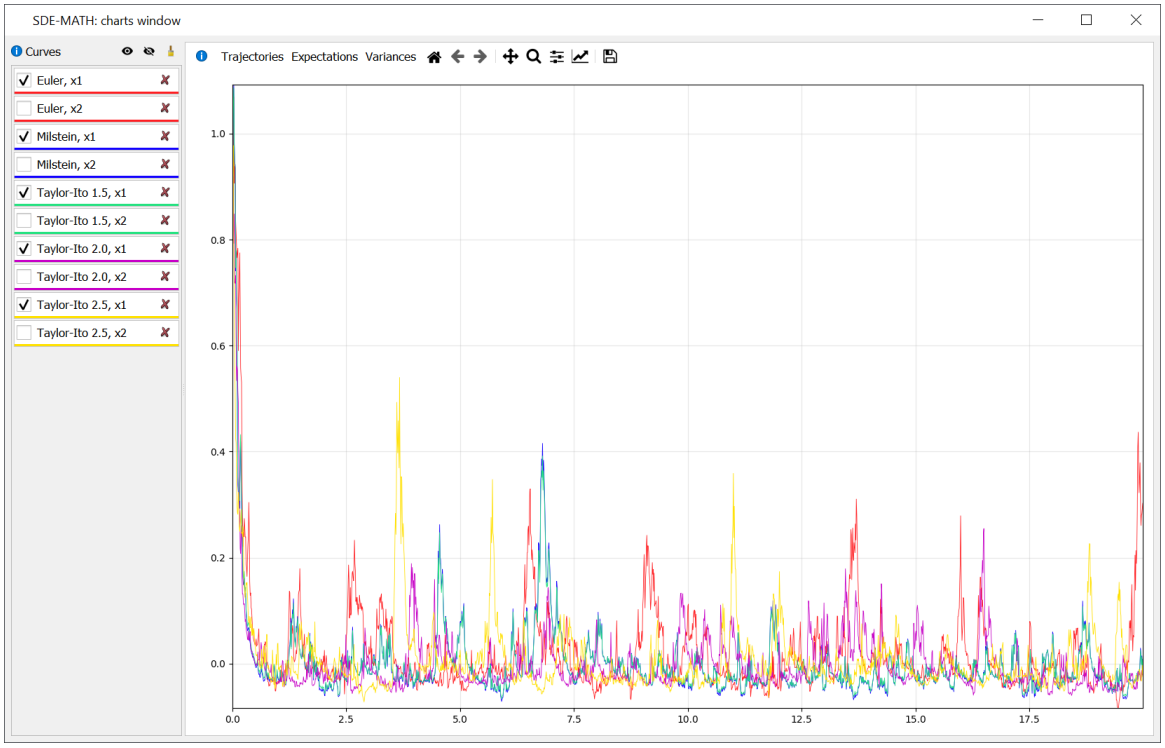
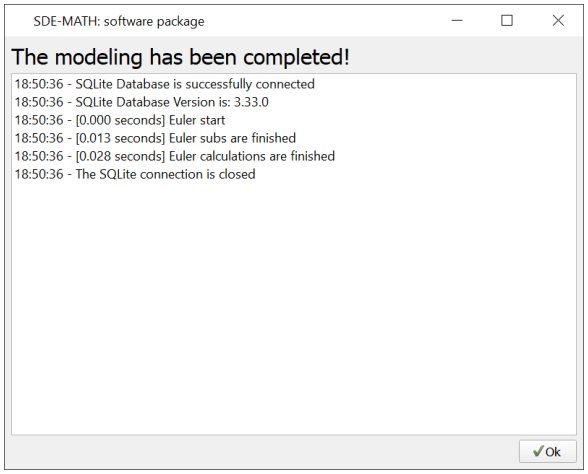
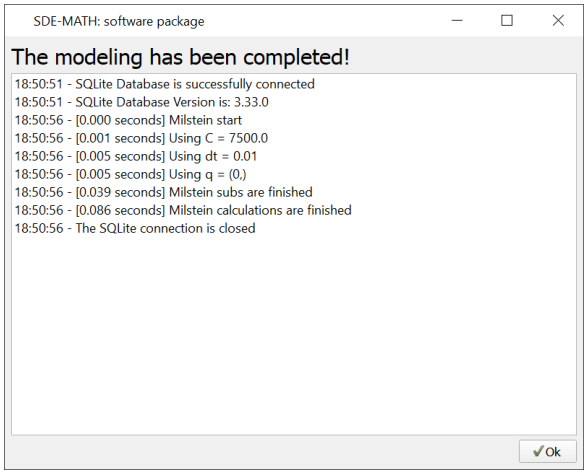


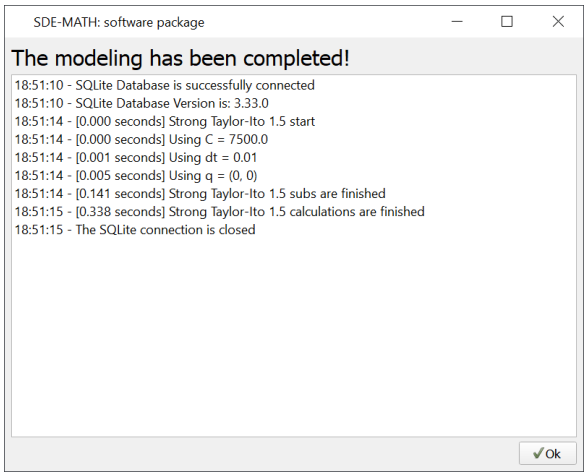
Figure 45: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(1)}$ component, $C = 7500$, $dt = 0.01$)



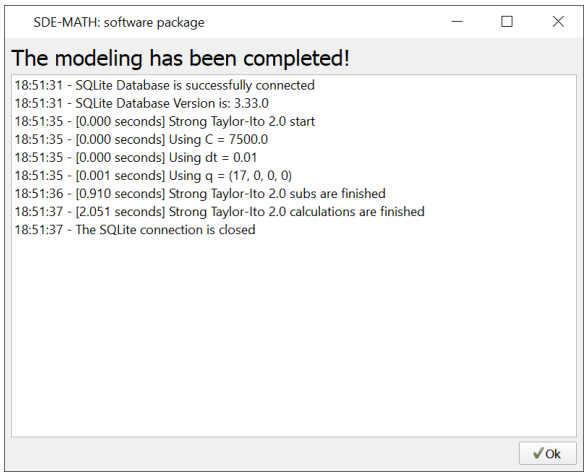
Euler scheme ($dt = 0.01$)



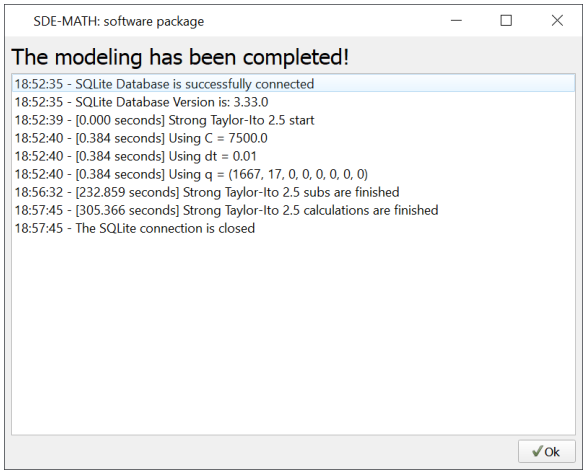
Milstein scheme ($C = 7500$, $dt = 0.01$)



Strong Taylor–Itô scheme of order 1.5 ($C = 7500$, $dt = 0.01$)



Strong Taylor–Itô scheme of order 2.0 ($C = 7500$, $dt = 0.01$)



Strong Taylor–Itô scheme of order 2.5 ($C = 7500$, $dt = 0.01$)

Figure 46: Modeling logs

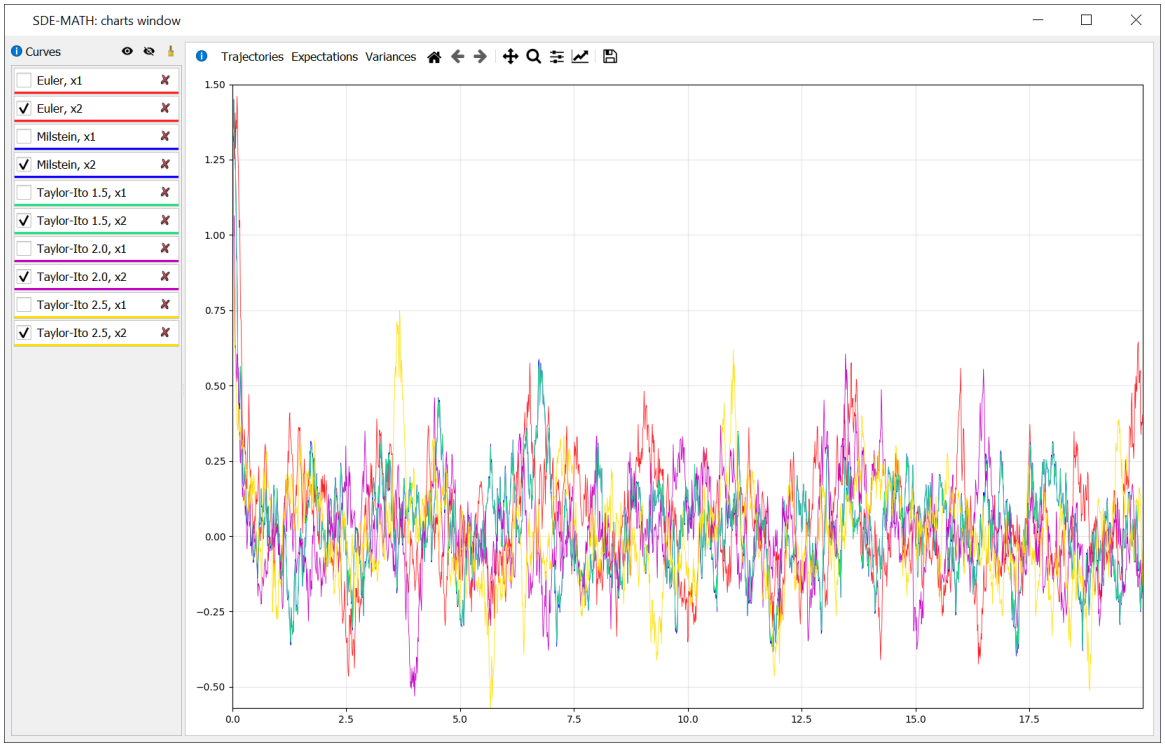


Figure 47: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(2)}$ component, $C = 7500$, $dt = 0.01$)

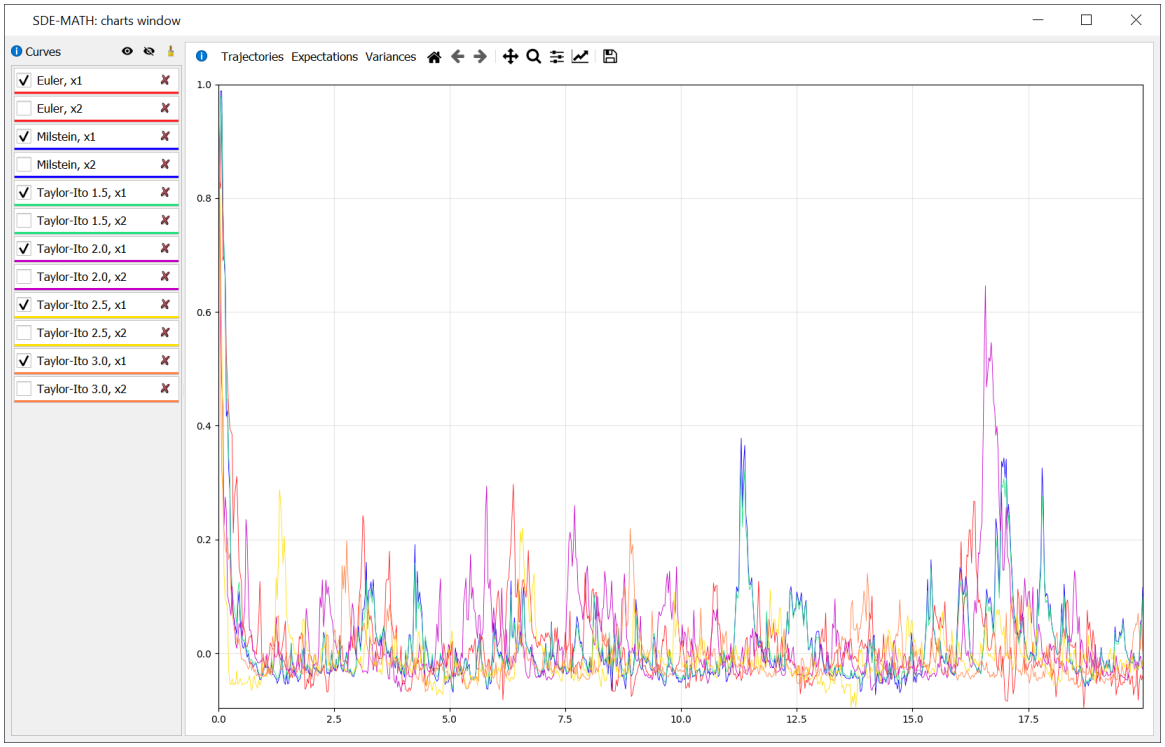
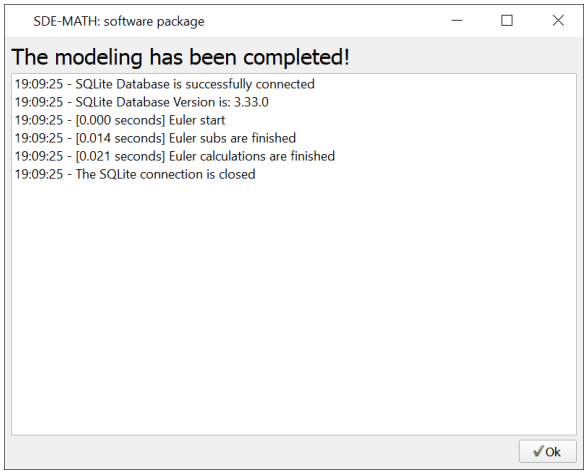
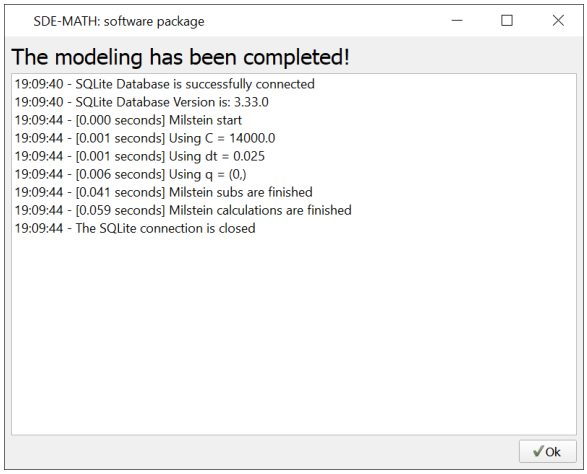


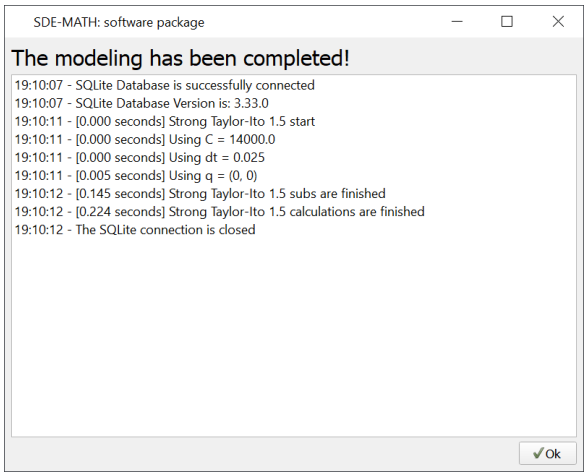
Figure 48: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(1)}$ component, $C = 14000$, $dt = 0.025$)



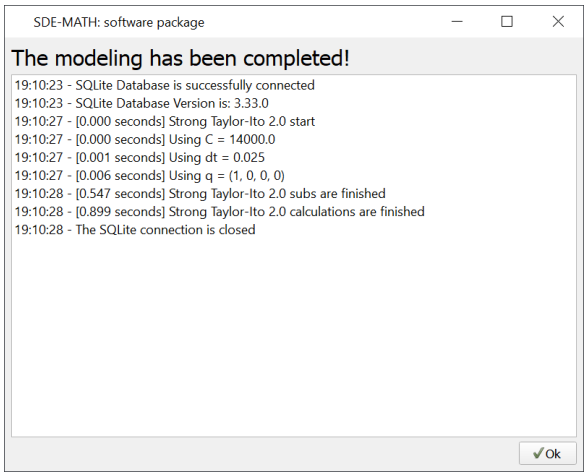
Euler scheme ($dt = 0.025$)



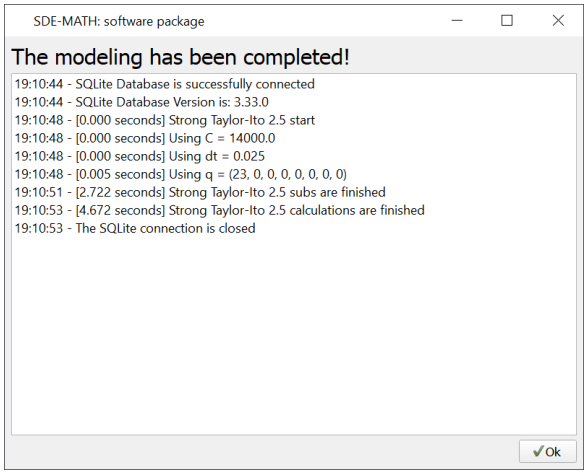
Milstein scheme ($C = 14000$, $dt = 0.025$)



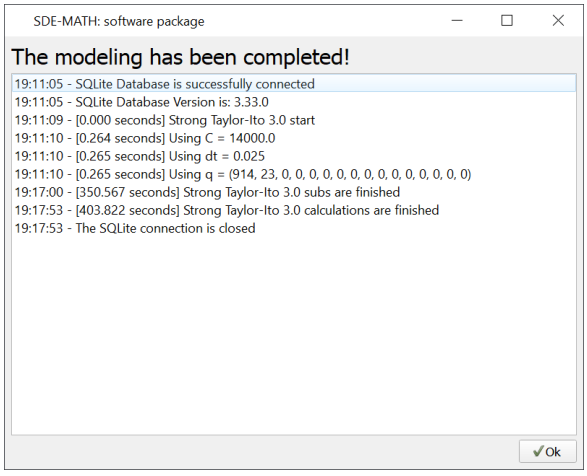
Strong Taylor–Itô scheme of order 1.5 ($C = 14000$, $dt = 0.025$)



Strong Taylor–Itô scheme of order 2.0 ($C = 14000$, $dt = 0.025$)



Strong Taylor–Itô scheme of order 2.5 ($C = 14000$, $dt = 0.025$)



Strong Taylor–Itô scheme of order 3.0 ($C = 14000$, $dt = 0.025$)

Figure 49: Modeling logs

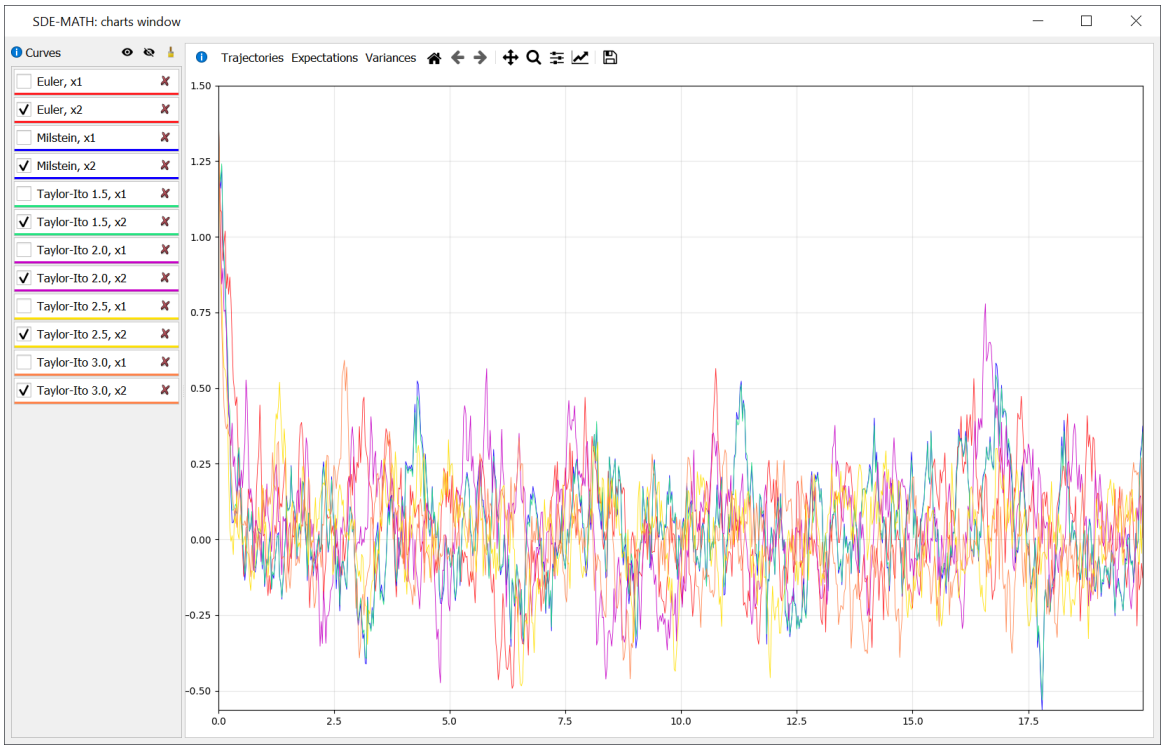
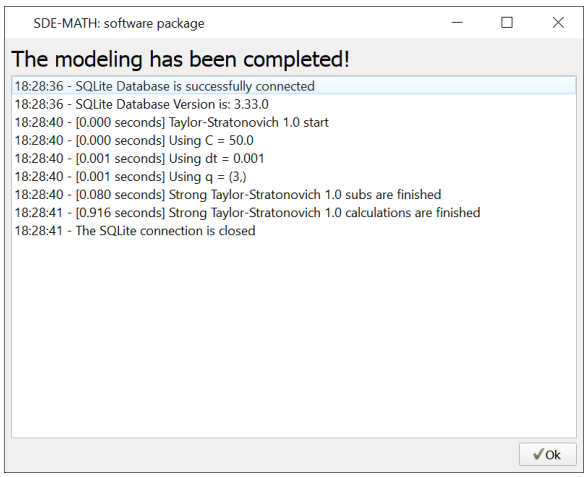
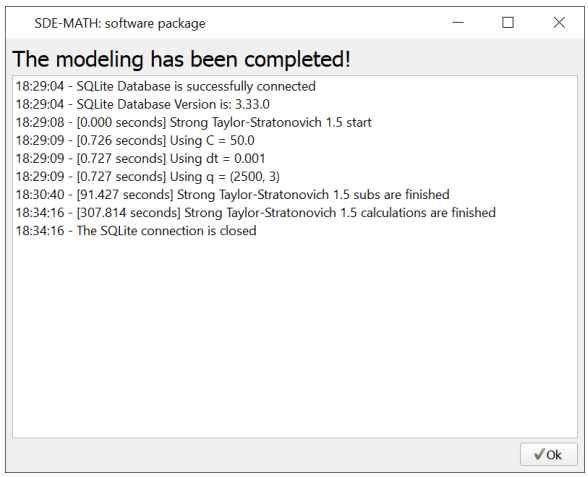


Figure 50: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(2)}$ component, $C = 14000$, $dt = 0.025$)



Strong Taylor–Stratonovich scheme of order 1.0 ($C = 50$, $dt = 0.001$)



Strong Taylor–Stratonovich scheme of order 1.5 ($C = 50$, $dt = 0.001$)

Figure 51: Modeling logs

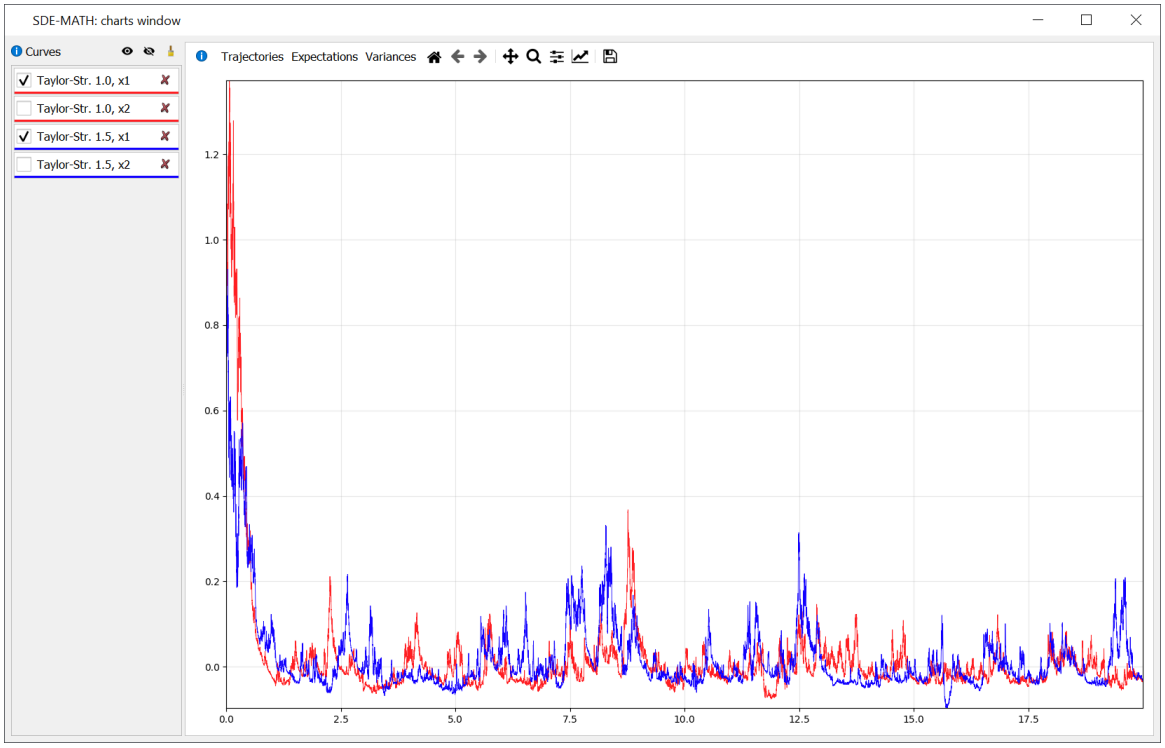


Figure 52: Strong Taylor–Stratonovich schemes of orders 1.0 and 1.5 ($\mathbf{x}_t^{(1)}$ component, $C = 50$, $dt = 0.001$)

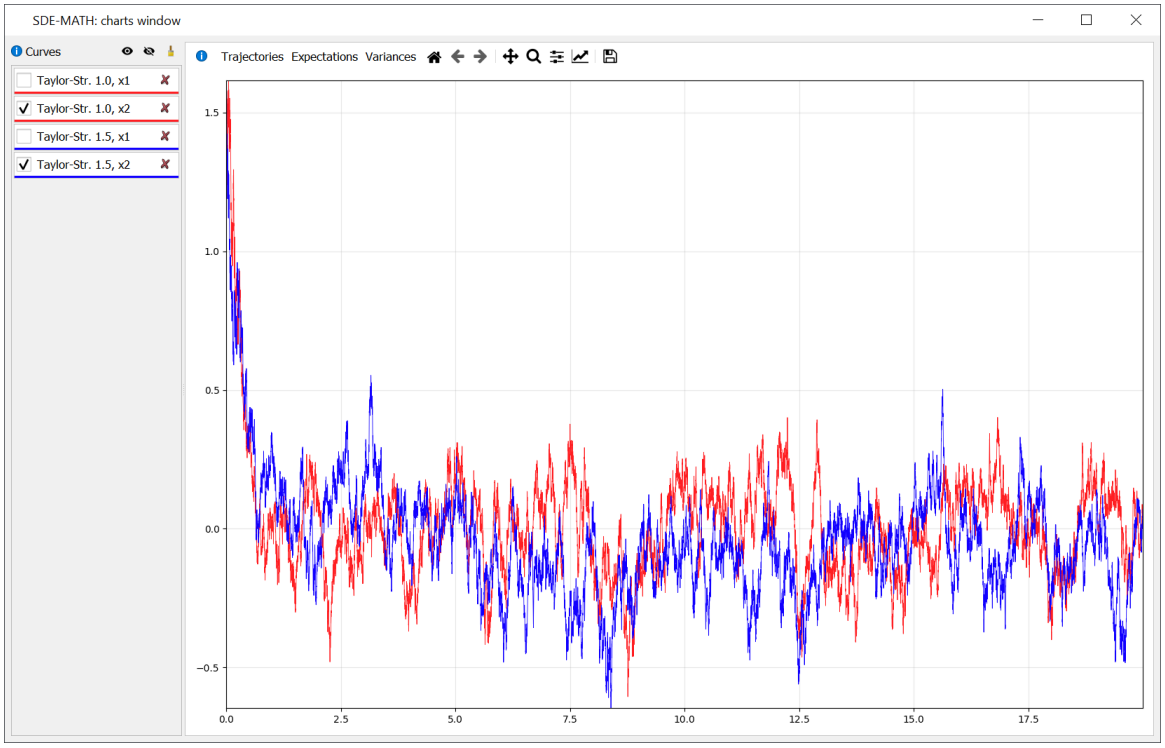


Figure 53: Strong Taylor–Stratonovich schemes of orders 1.0 and 1.5 ($\mathbf{x}_t^{(2)}$ component, $C = 50$, $dt = 0.001$)

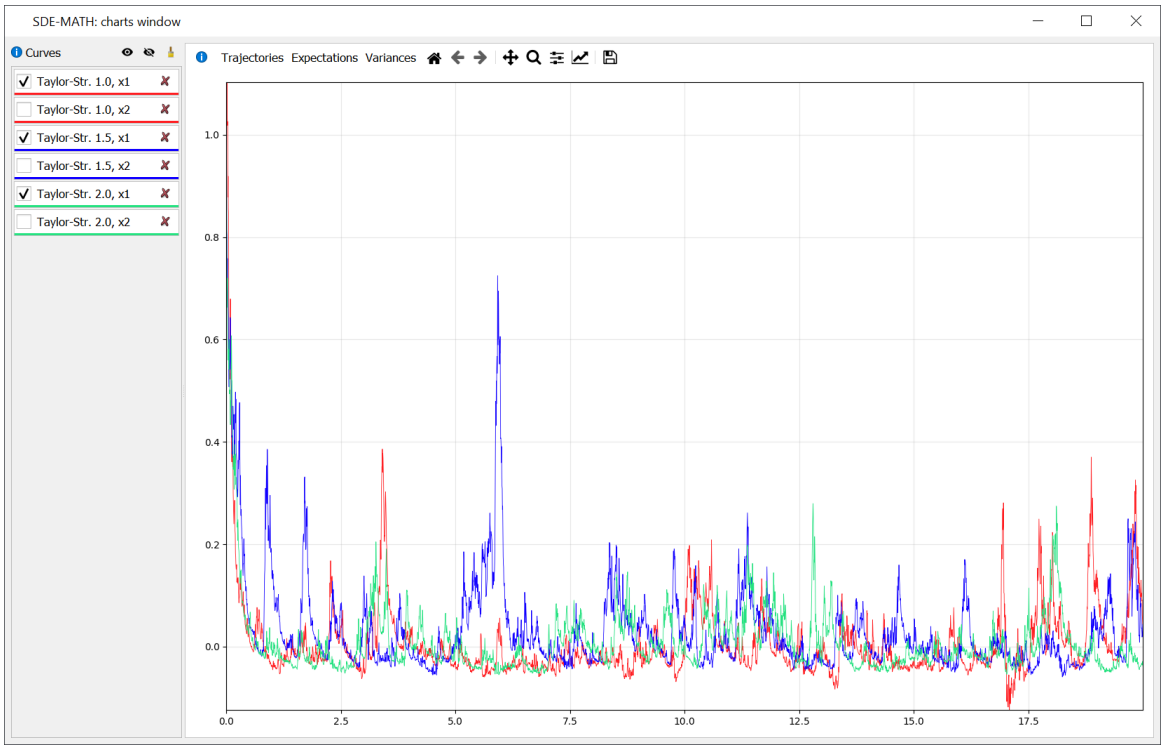
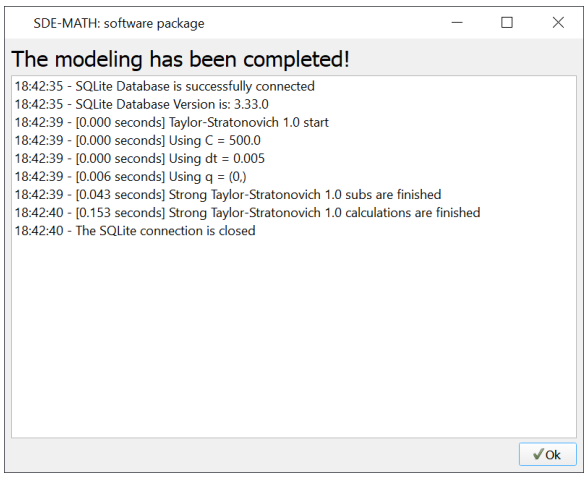
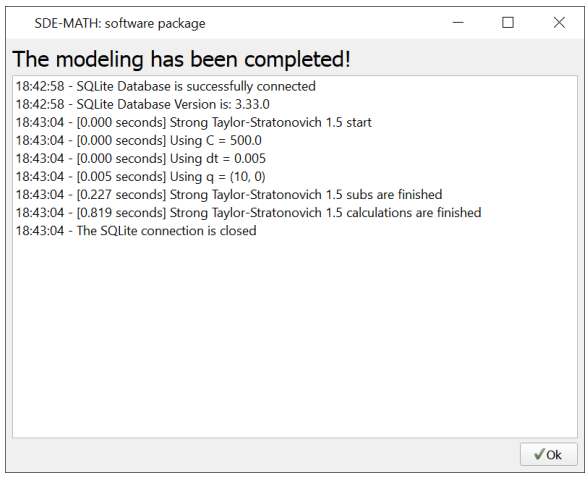


Figure 54: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(1)}$ component, $C = 500$, $dt = 0.005$)



Strong Taylor–Stratonovich scheme of order 1.0 ($C = 500$, $dt = 0.005$)



Strong Taylor–Stratonovich scheme of order 1.5 ($C = 500$, $dt = 0.005$)

Figure 55: Modeling logs

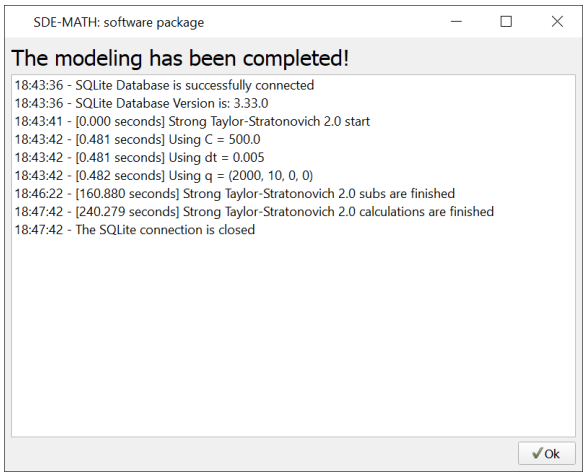


Figure 56: Strong Taylor–Stratonovich scheme of order 2.0 ($C = 500$, $dt = 0.005$)

Figure 57: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(2)}$ component, $C = 500$, $dt = 0.005$)

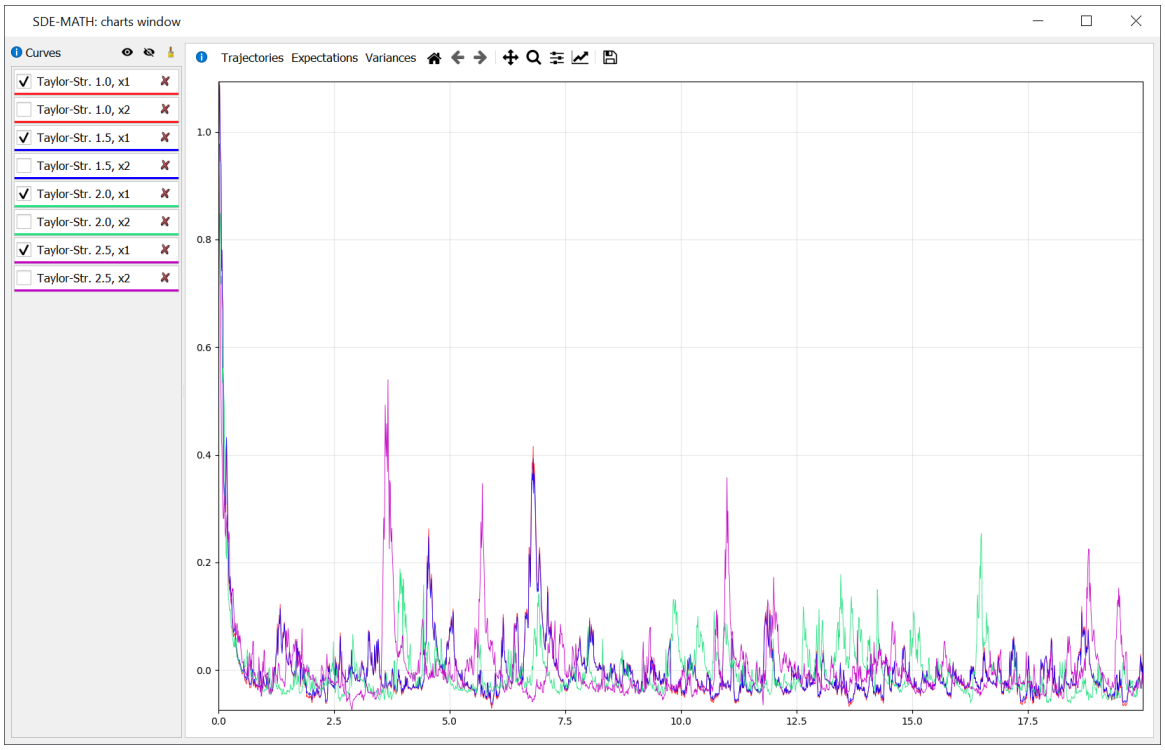
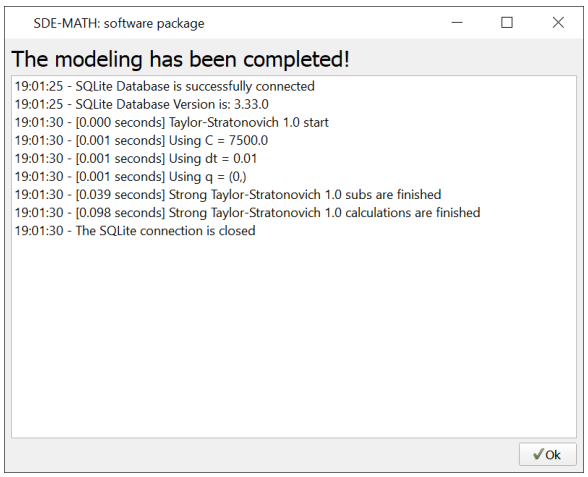
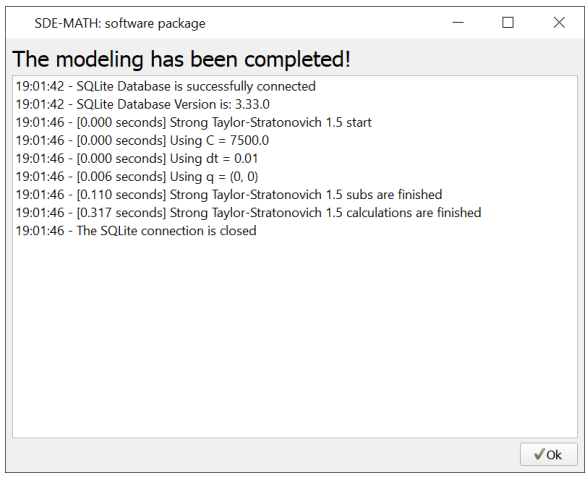


Figure 58: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(1)}$ component, $C = 7500$, $dt = 0.01$)

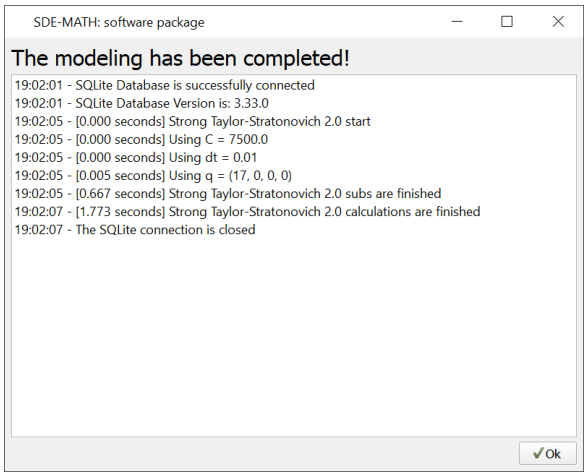


Strong Taylor–Stratonovich scheme of order 1.0 ($C = 7500$, $dt = 0.01$)

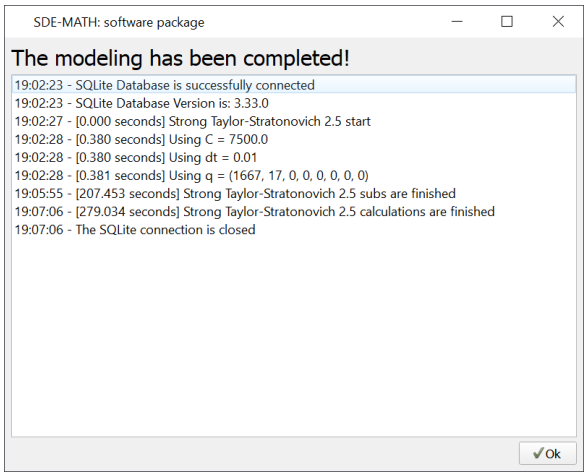


Strong Taylor–Stratonovich scheme of order 1.5 ($C = 7500$, $dt = 0.01$)

Figure 59: Modeling logs



Strong Taylor–Stratonovich scheme of order 2.0 ($C = 7500$, $dt = 0.01$)



Strong Taylor–Stratonovich scheme of order 2.5 ($C = 7500$, $dt = 0.01$)

Figure 60: Modeling logs

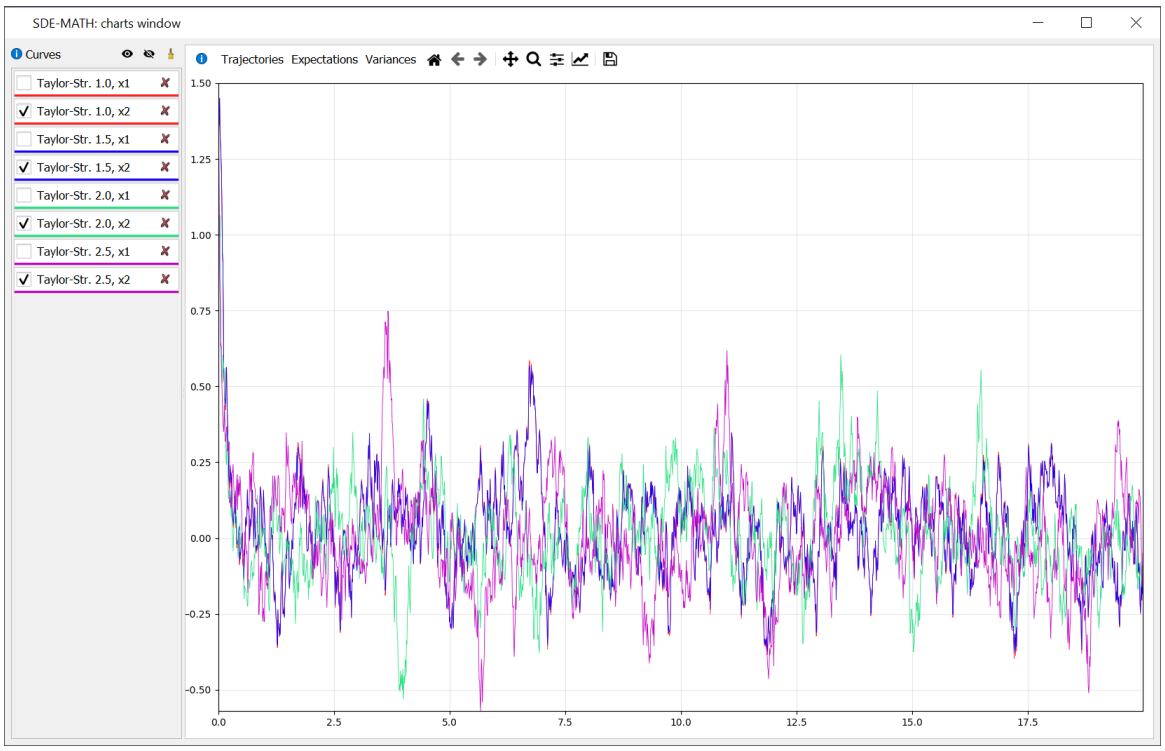
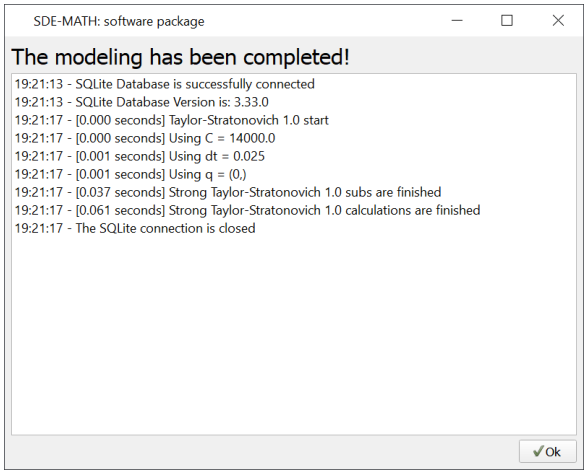
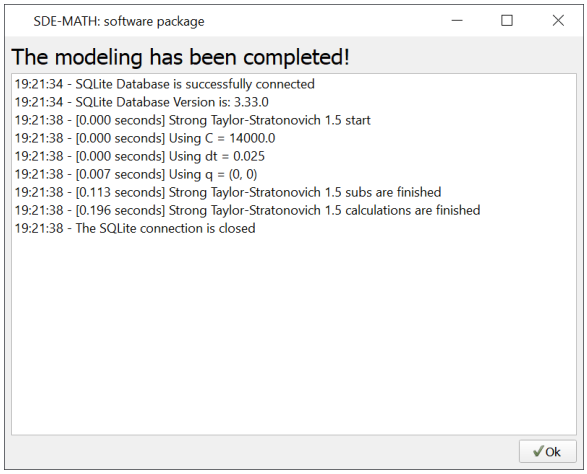


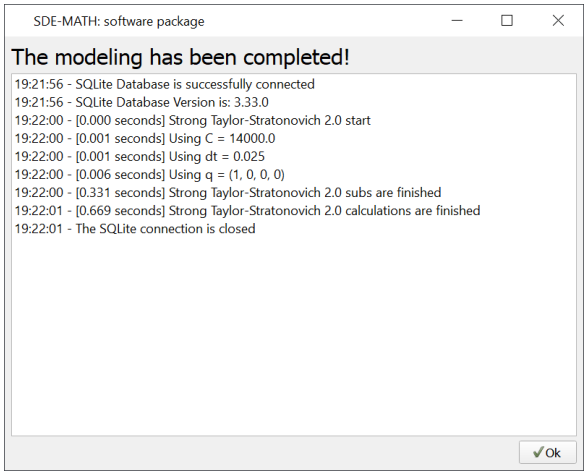
Figure 61: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(2)}$ component, $C = 7500$, $dt = 0.01$)



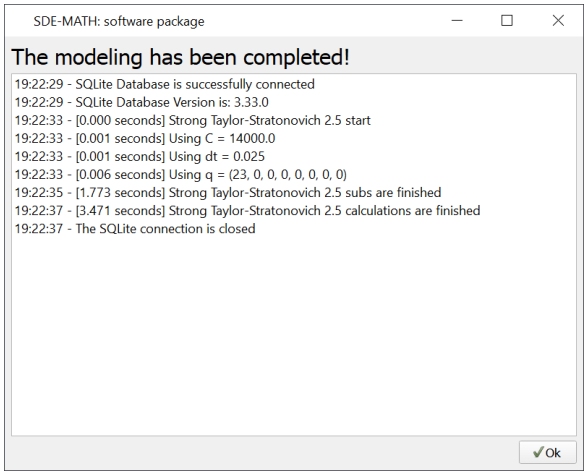
Strong Taylor–Stratonovich scheme of order 1.0 ($C = 14000$, $dt = 0.025$)



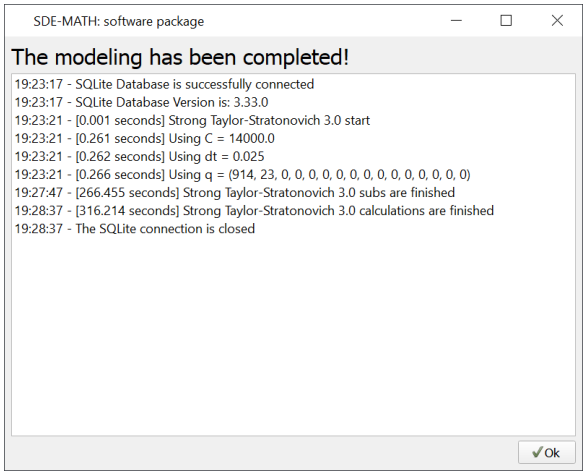
Strong Taylor–Stratonovich scheme of order 1.5 ($C = 14000$, $dt = 0.025$)



Strong Taylor–Stratonovich scheme of order 2.0 ($C = 14000$, $dt = 0.025$)



Strong Taylor–Stratonovich scheme of order 2.5 ($C = 14000$, $dt = 0.025$)



Strong Taylor–Stratonovich scheme of order 3.0 ($C = 14000$, $dt = 0.025$)

Figure 62: Modeling logs

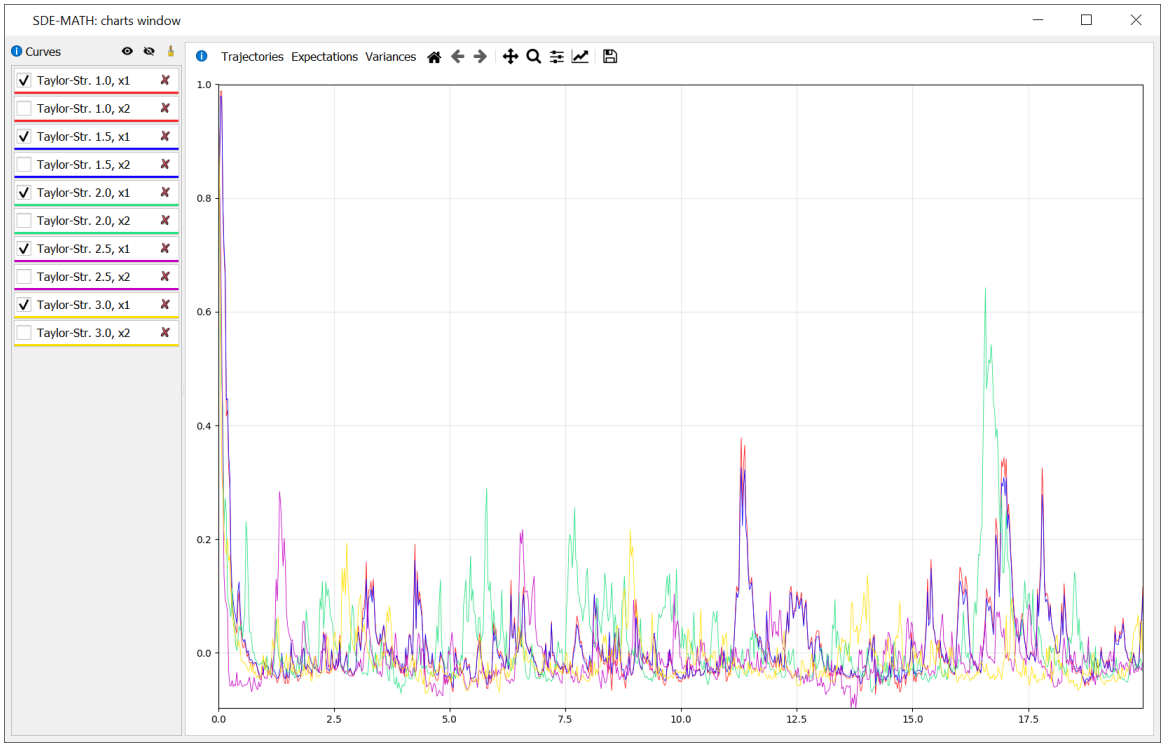


Figure 63: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(1)}$ component, $C = 14000$, $dt = 0.025$)

Figure 64: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(2)}$ component, $C = 14000$, $dt = 0.025$)

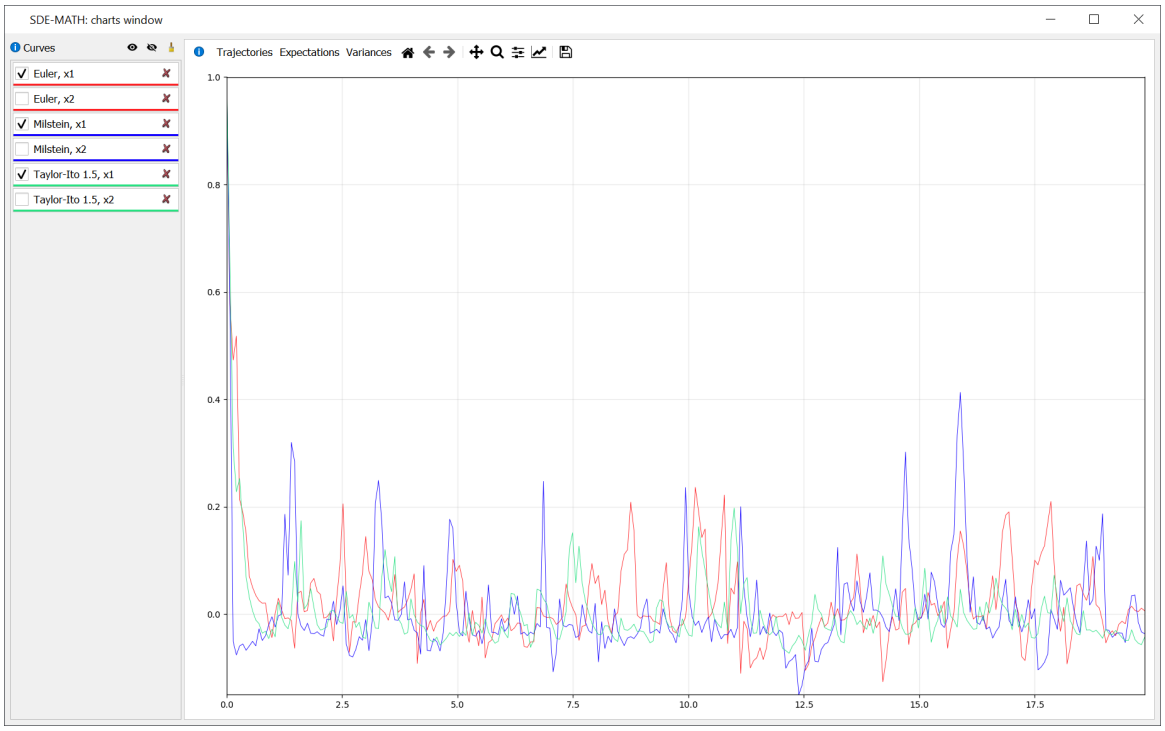
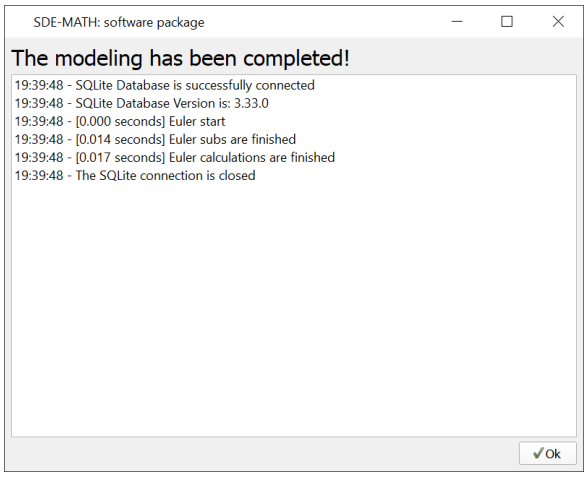
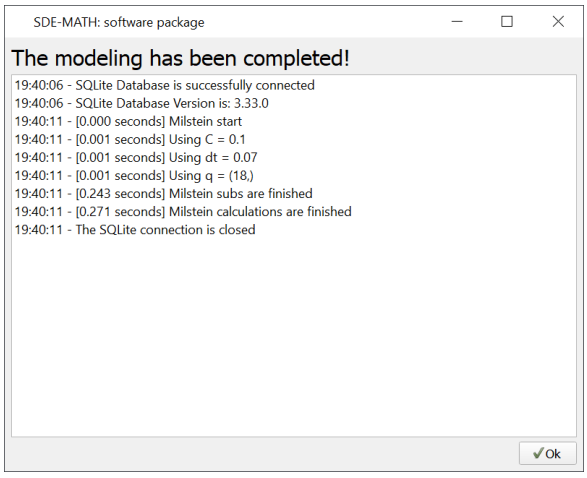


Figure 65: Strong Taylor–Itô schemes of orders 0.5, 1.0, and 1.5 ($\mathbf{x}_t^{(1)}$ component, $C = 0.1$, $dt = 0.07$)



Euler scheme ($dt = 0.07$)



Milstein scheme ($C = 0.1$, $dt = 0.07$)

Figure 66: Modeling logs

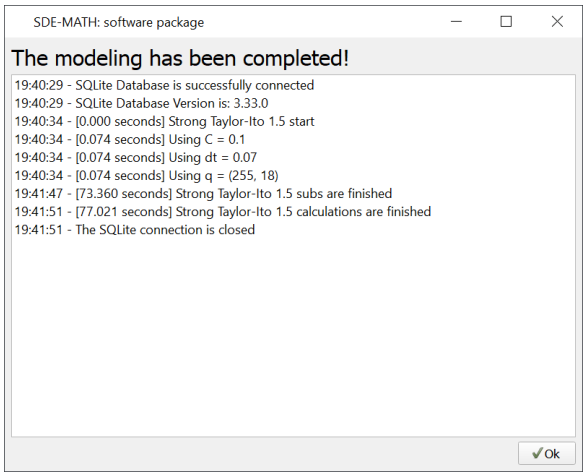


Figure 67: Strong Taylor–Itô scheme of order 1.5 ($C = 0.1$, $dt = 0.07$)

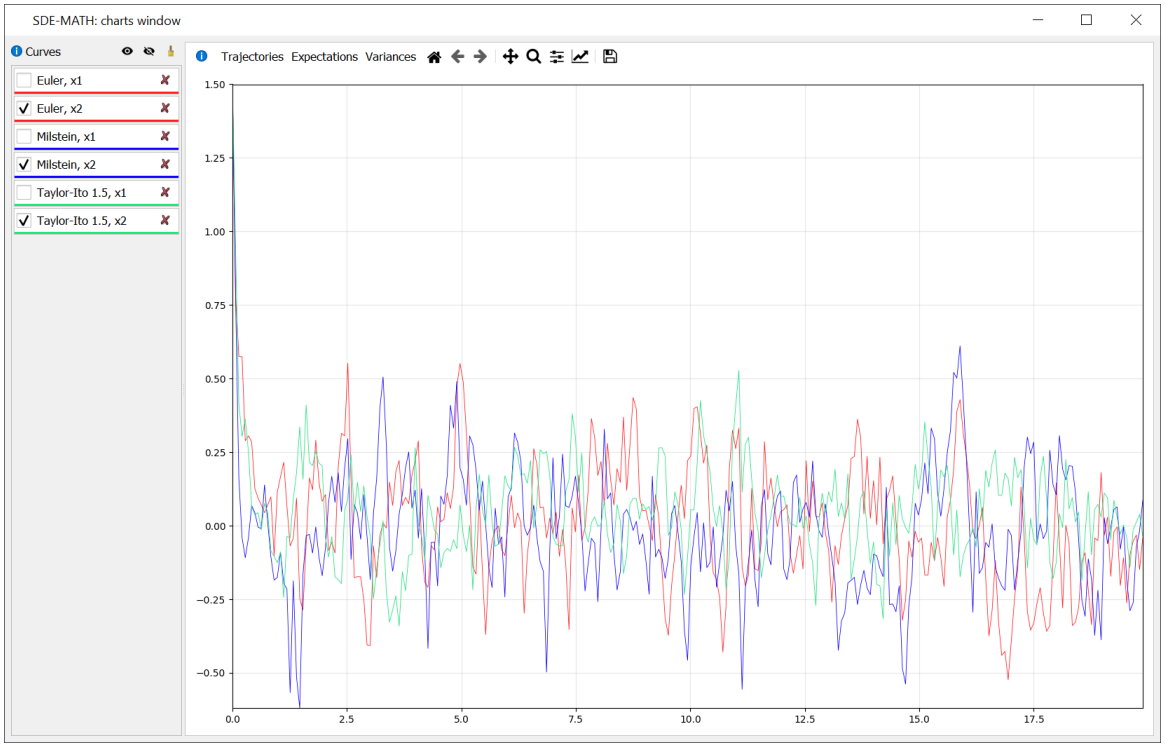
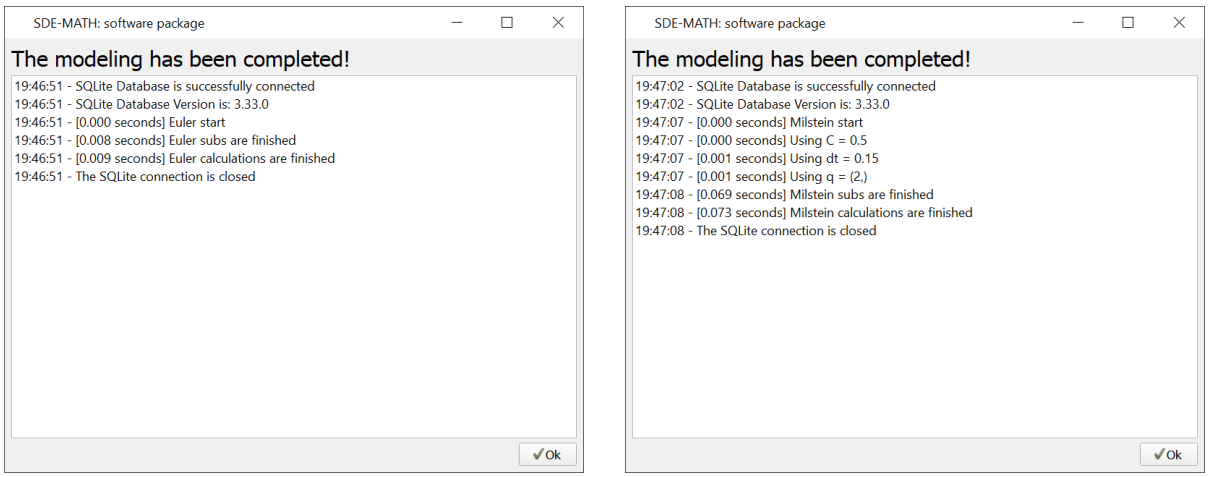


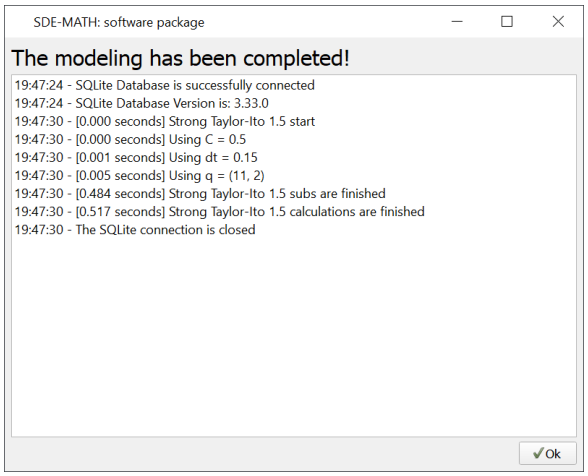
Figure 68: Strong Taylor–Itô schemes of orders 0.5, 1.0, and 1.5 ($\mathbf{x}_t^{(2)}$ component, $C = 0.1$, $dt = 0.07$)

Figure 69: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(1)}$ component, $C = 0.5$, $dt = 0.15$)

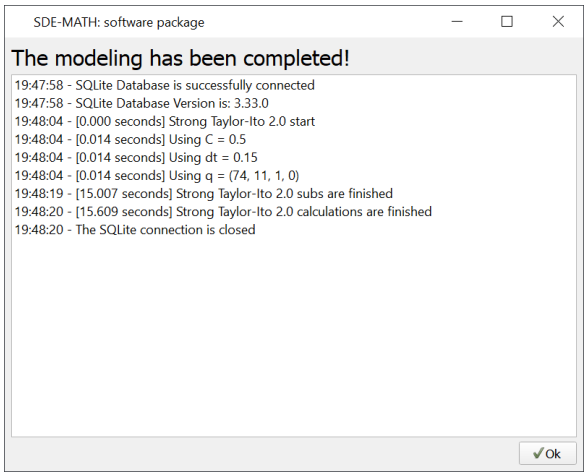


Euler scheme ($dt = 0.15$) Milstein scheme ($C = 0.5$, $dt = 0.15$)

Figure 70: Modeling logs



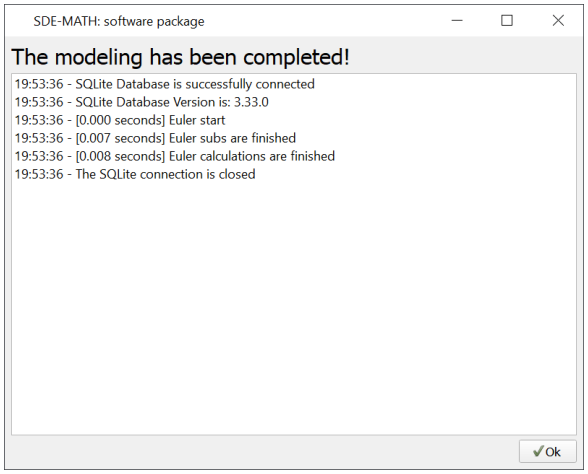
Strong Taylor–Itô scheme of order 1.5 ($C = 0.5$, $dt = 0.15$)



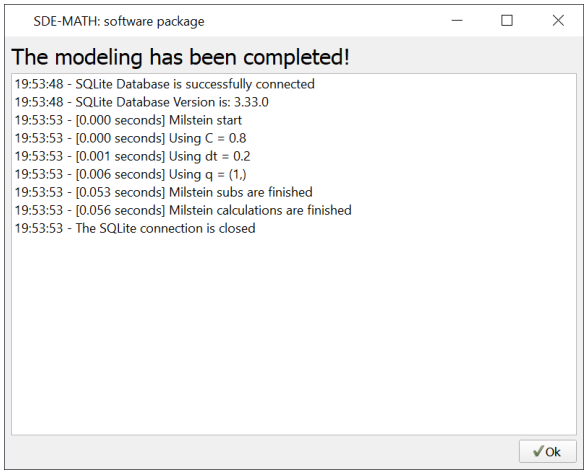
Strong Taylor–Itô scheme of order 2.0 ($C = 0.5$, $dt = 0.15$)

Figure 71: Modeling logs

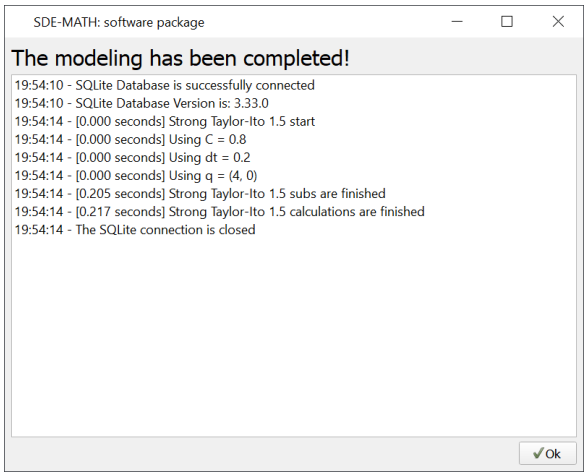
Figure 72: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(2)}$ component, $C = 0.5$, $dt = 0.15$)



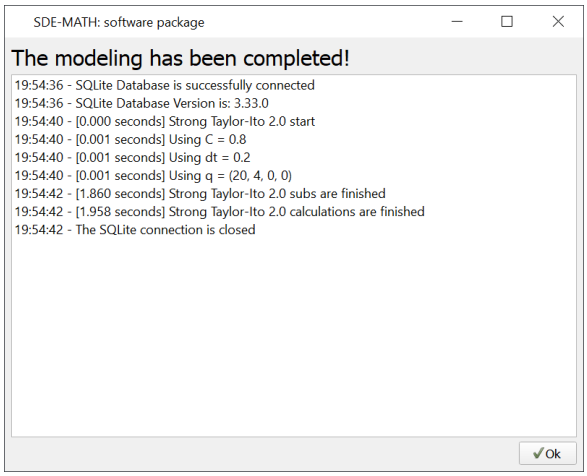
Euler scheme ($dt = 0.2$)



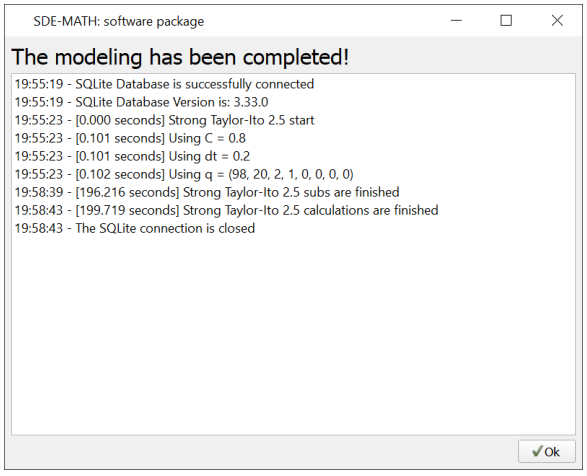
Milstein scheme ($C = 0.8$, $dt = 0.2$)



Strong Taylor–Itô scheme of order 1.5 ($C = 0.8$, $dt = 0.2$)



Strong Taylor–Itô scheme of order 2.0 ($C = 0.8$, $dt = 0.2$)



Strong Taylor–Itô scheme of order 2.5 ($C = 0.8$, $dt = 0.2$)

Figure 73: Modeling logs

Figure 74: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(1)}$ component, $C = 0.8$, $dt = 0.2$)

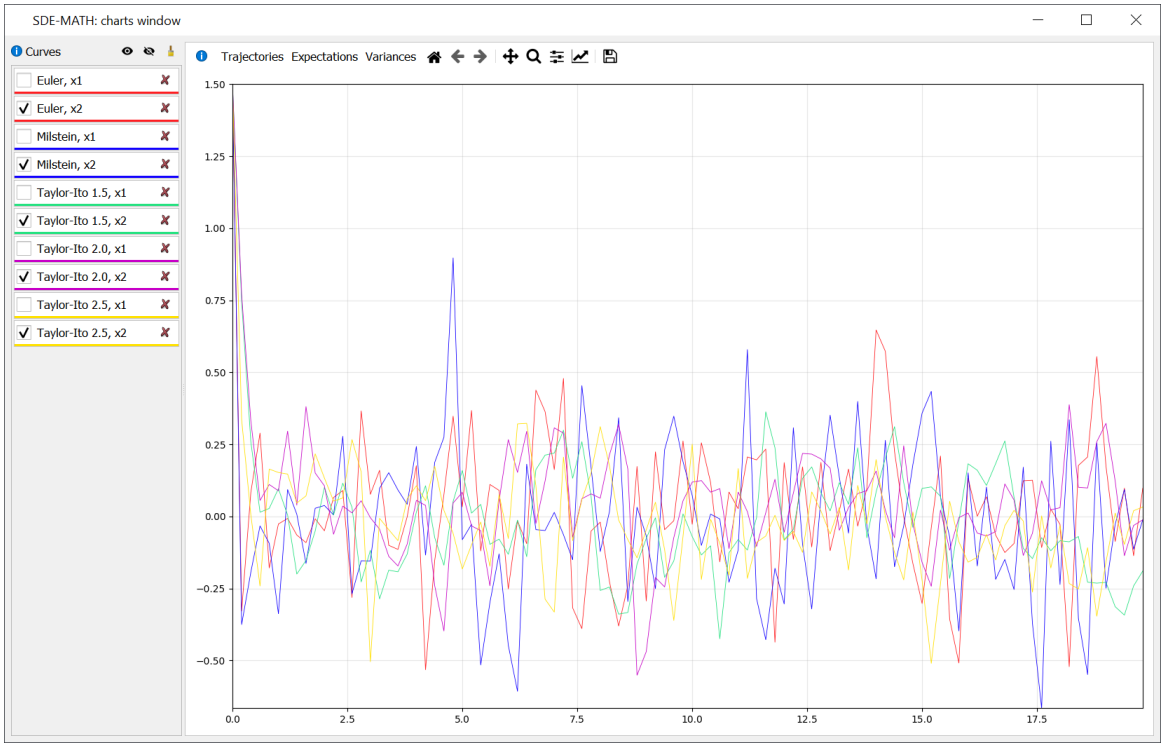
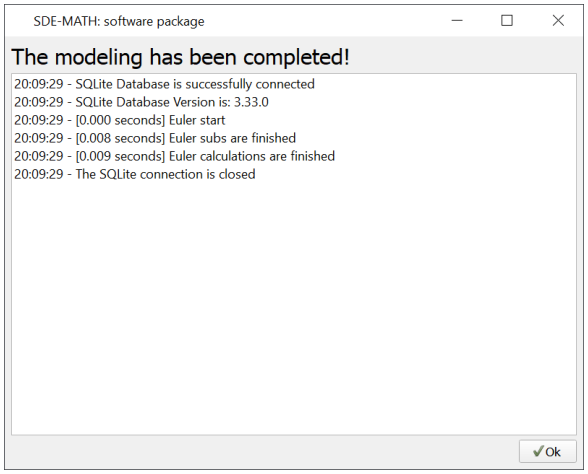
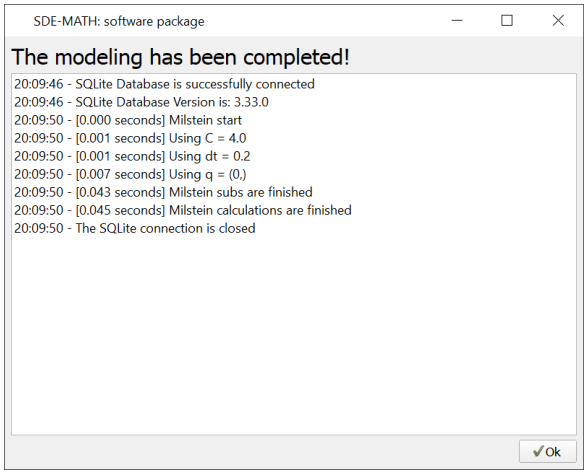


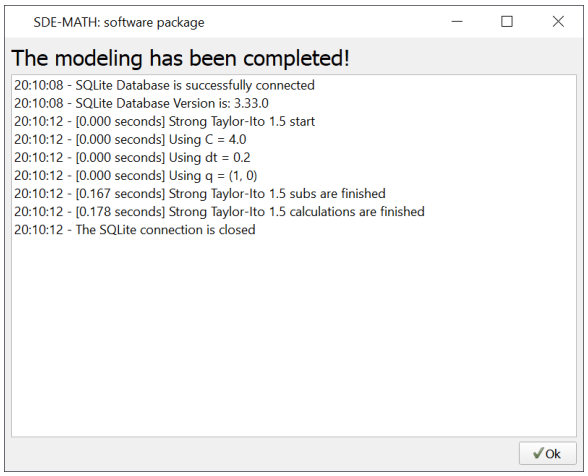
Figure 75: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(2)}$ component, $C = 0.8$, $dt = 0.2$)



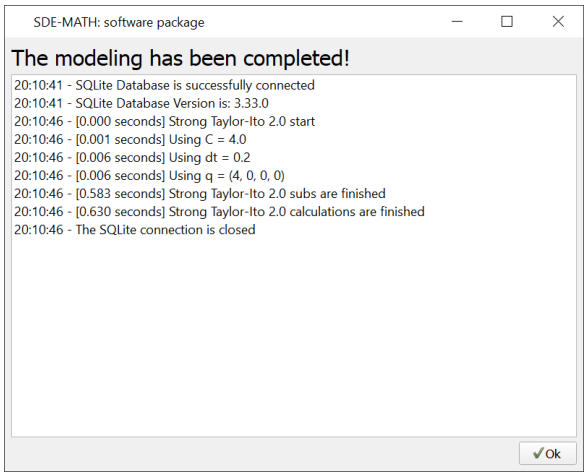
Euler scheme ($dt = 0.2$)



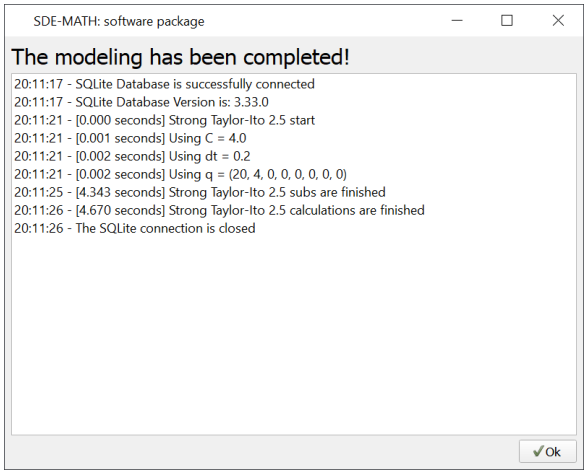
Milstein scheme ($C = 4$, $dt = 0.2$)



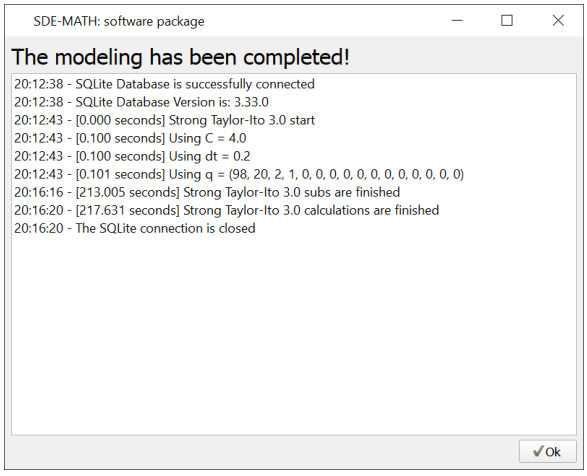
Strong Taylor–Itô scheme of order 1.5 ($C = 4$, $dt = 0.2$)



Strong Taylor–Itô scheme of order 2.0 ($C = 4$, $dt = 0.2$)



Strong Taylor–Itô scheme of order 2.5 ($C = 4$, $dt = 0.2$)



Strong Taylor–Itô scheme of order 3.0 ($C = 4$, $dt = 0.2$)

Figure 76: Modeling logs

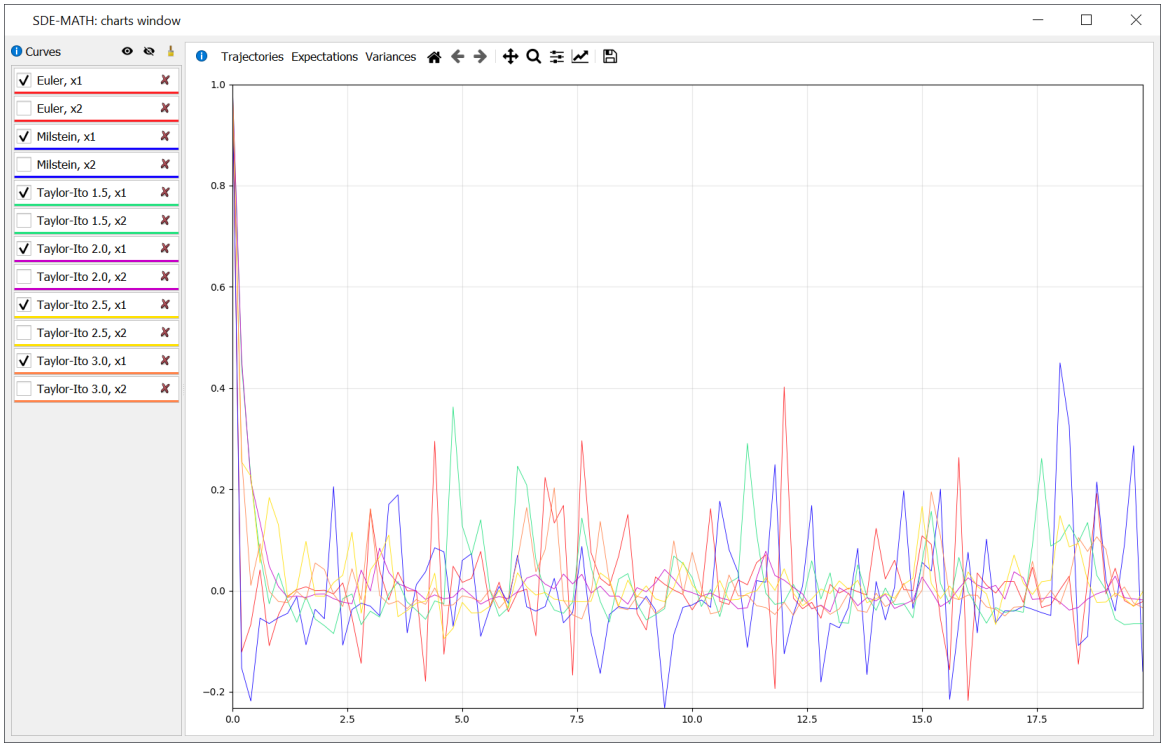


Figure 77: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(1)}$ component, $C = 4$, $dt = 0.2$)

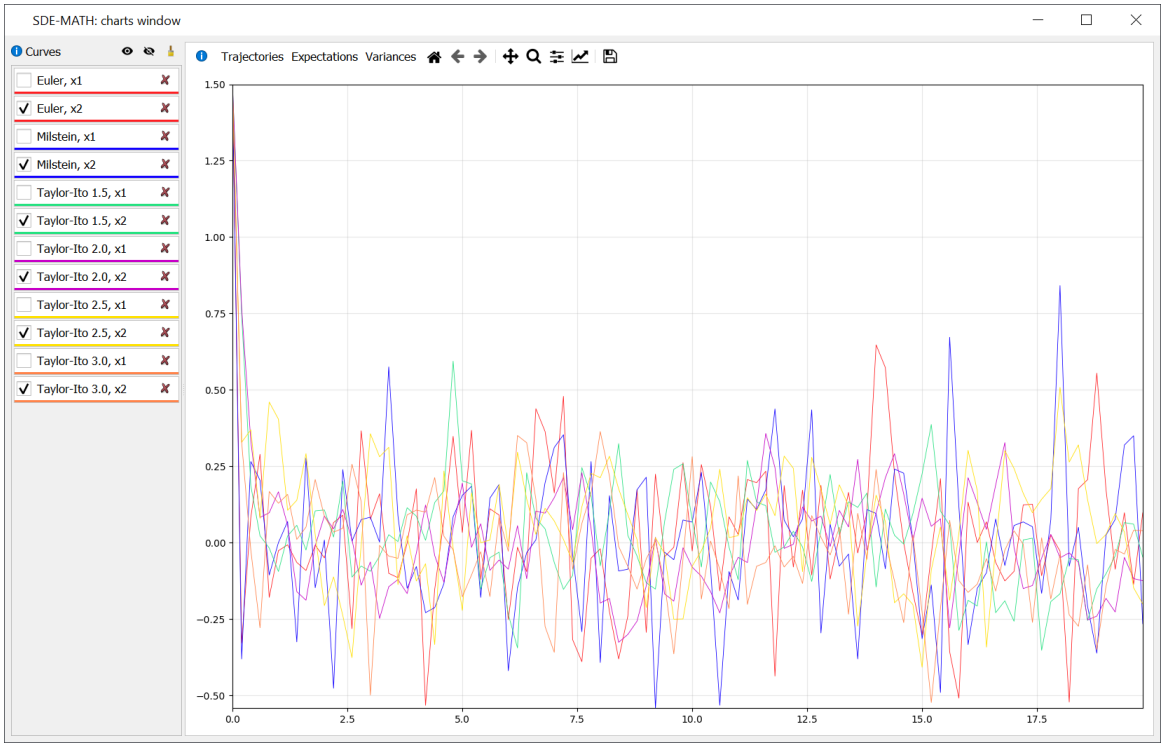
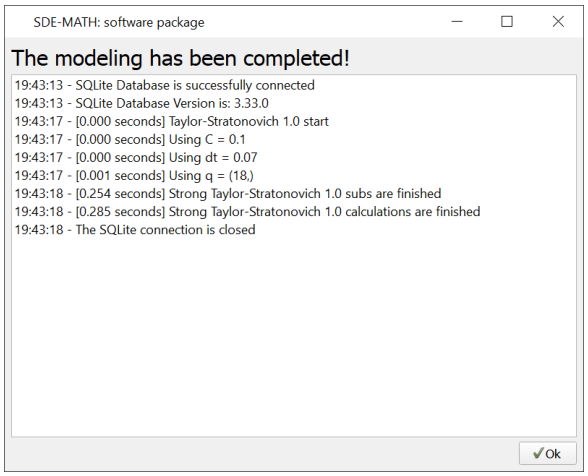
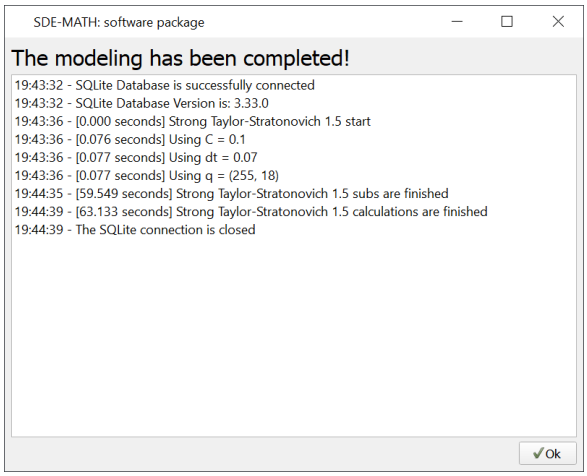


Figure 78: Strong Taylor–Itô schemes of orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(2)}$ component, $C = 4$, $dt = 0.2$)



Strong Taylor–Stratonovich scheme of order 1.0 ($C = 0.1$, $dt = 0.07$)



Strong Taylor–Stratonovich scheme of order 1.5 ($C = 0.1$, $dt = 0.07$)

Figure 79: Modeling logs

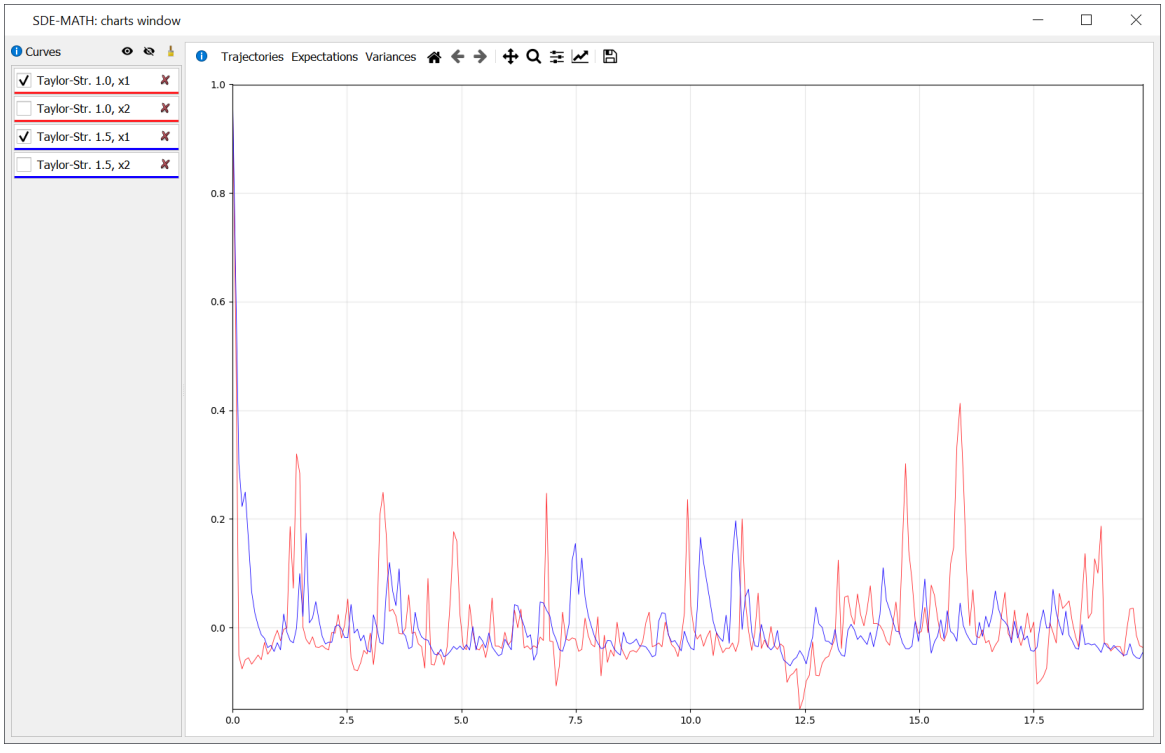


Figure 80: Strong Taylor–Stratonovich schemes of orders 1.0 and 1.5 ($\mathbf{x}_t^{(1)}$ component, $C = 0.1$, $dt = 0.07$)

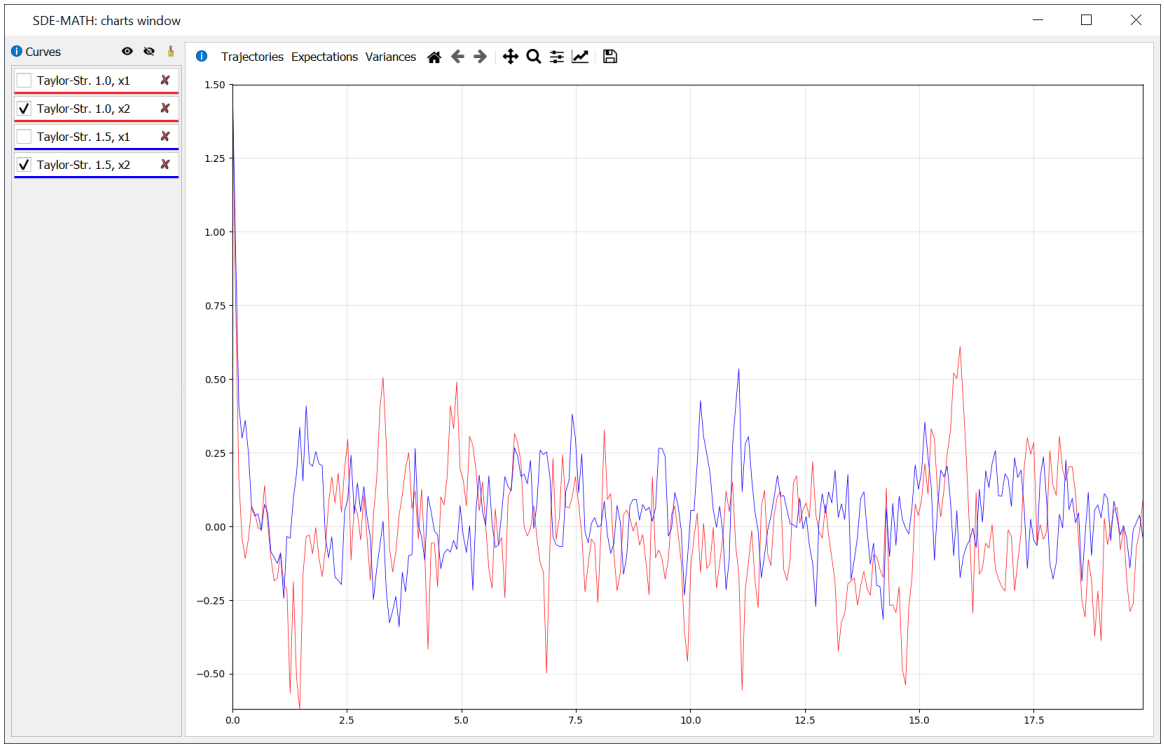
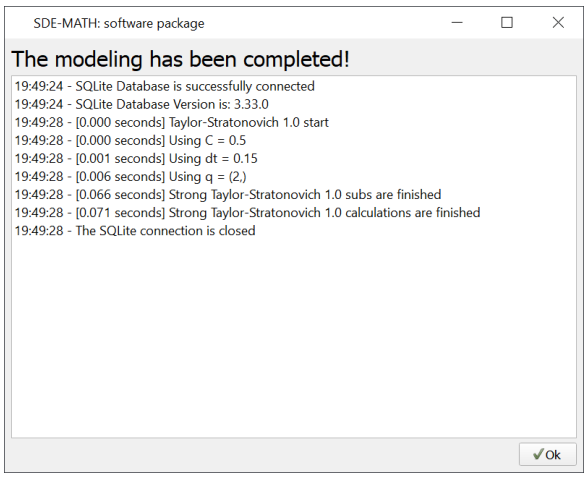
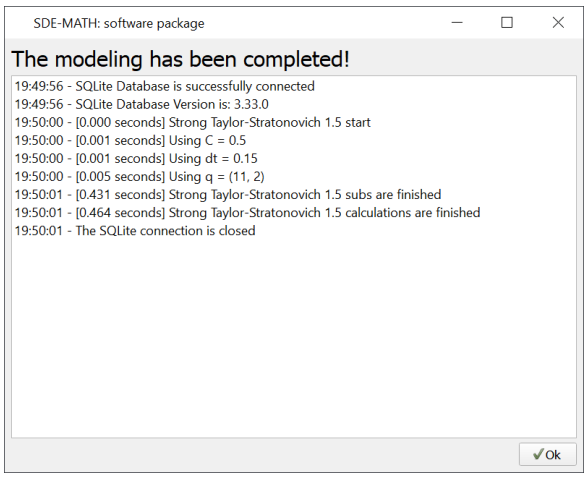


Figure 81: Strong Taylor–Stratonovich schemes of orders 1.0 and 1.5 ($\mathbf{x}_t^{(2)}$ component, $C = 0.1$, $dt = 0.07$)



Strong Taylor–Stratonovich scheme of order 1.0 ($C = 0.5$, $dt = 0.15$)



Strong Taylor–Stratonovich scheme of order 1.5 ($C = 0.5$, $dt = 0.15$)

Figure 82: Modeling logs

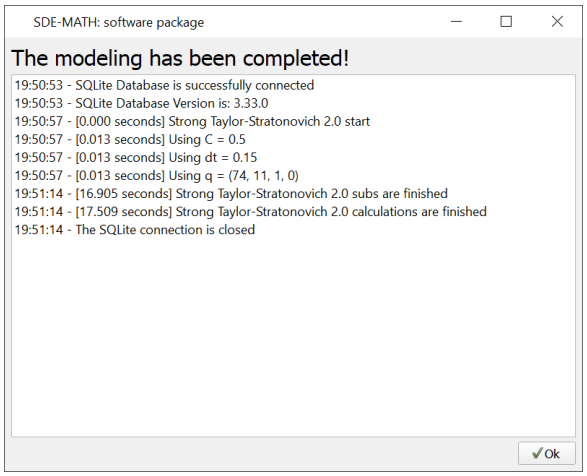


Figure 83: Strong Taylor–Stratonovich scheme of order 2.0 ($C = 0.5$, $dt = 0.15$)

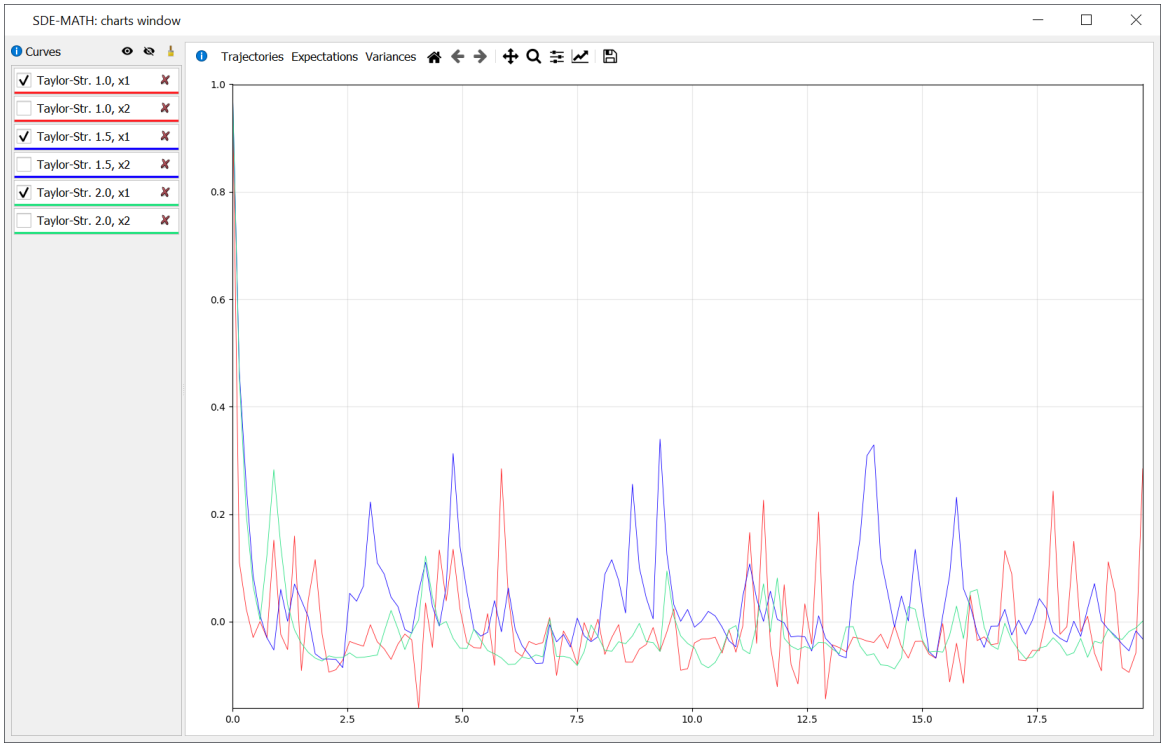
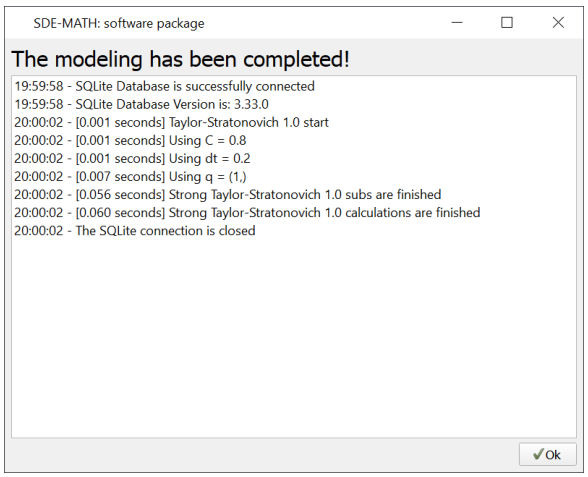
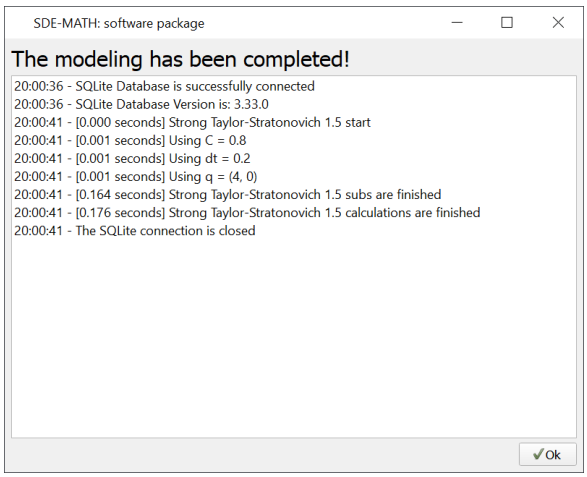


Figure 84: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(1)}$ component, $C = 0.5$, $dt = 0.15$)

Figure 85: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, and 2.0 ($\mathbf{x}_t^{(2)}$ component, $C = 0.5$, $dt = 0.15$)

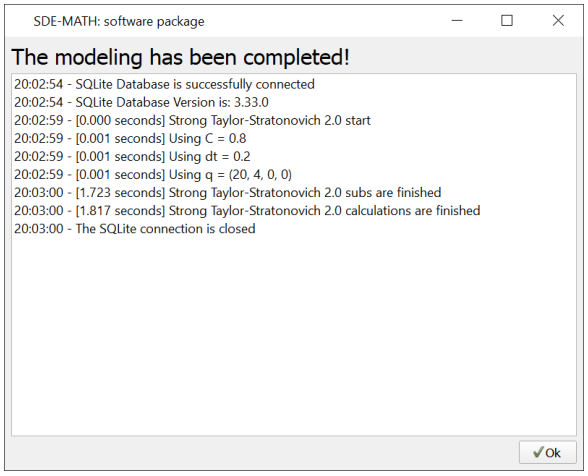


Strong Taylor–Stratonovich scheme of order 1.0 ($C = 0.8$, $dt = 0.2$)

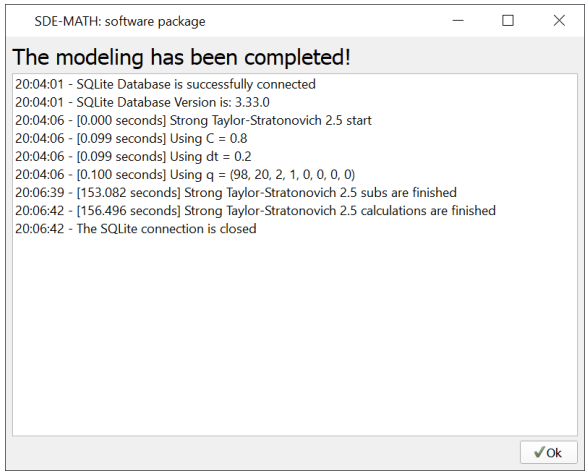


Strong Taylor–Stratonovich scheme of order 1.5 ($C = 0.8$, $dt = 0.2$)

Figure 86: Modeling logs



Strong Taylor–Stratonovich scheme of order 2.0 ($C = 0.8$, $dt = 0.2$)



Strong Taylor–Stratonovich scheme of order 2.5 ($C = 0.8$, $dt = 0.2$)

Figure 87: Modeling logs

Figure 88: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(1)}$ component, $C = 0.8$, $dt = 0.2$)

Figure 89: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, and 2.5 ($\mathbf{x}_t^{(2)}$ component, $C = 0.8$, $dt = 0.2$)

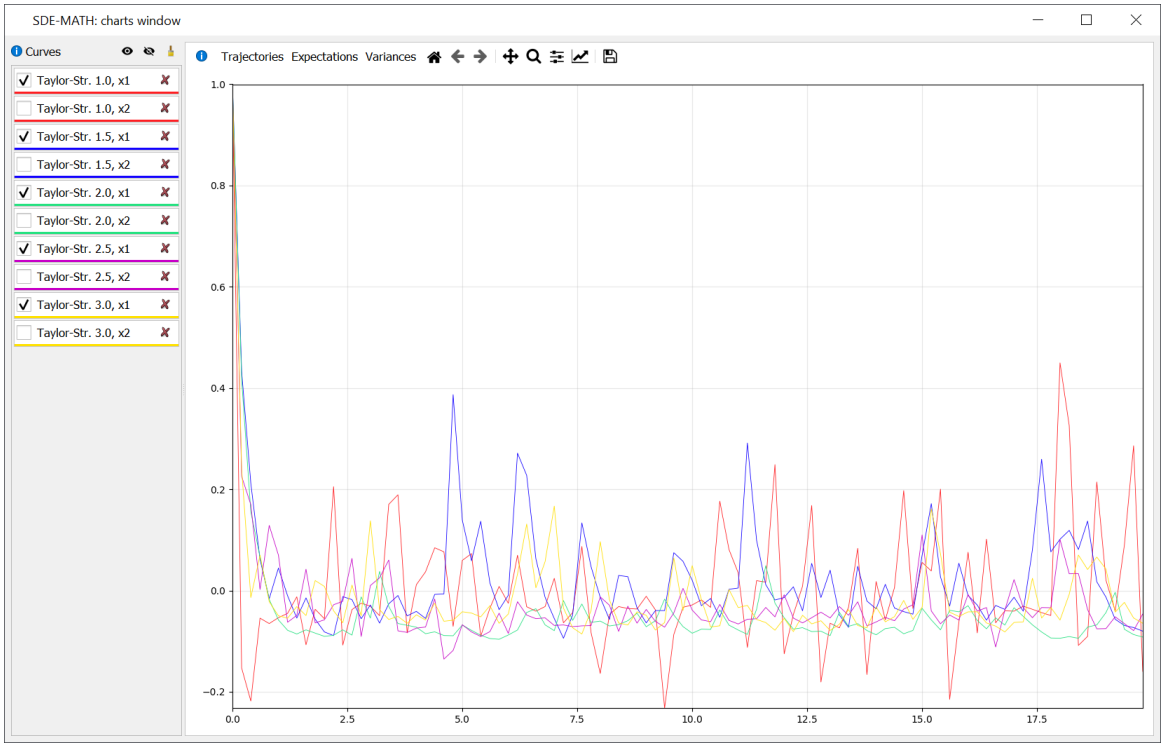
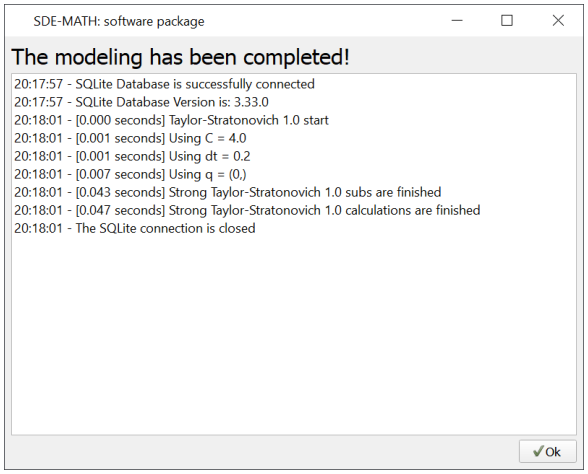
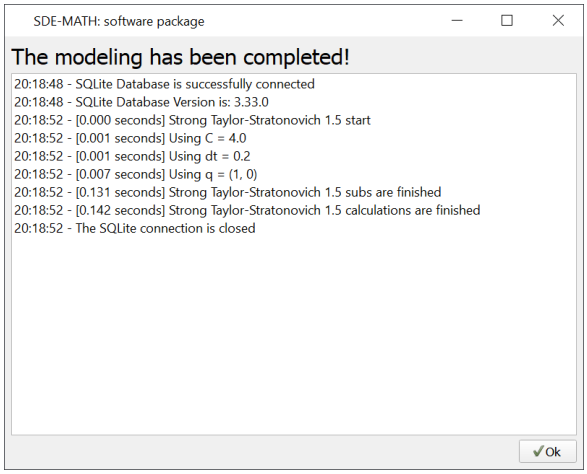


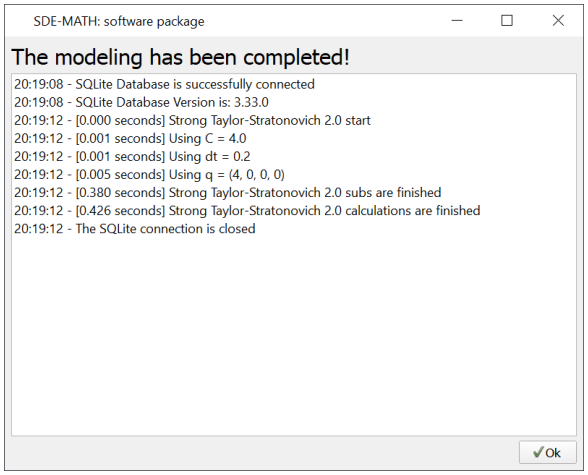
Figure 90: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(1)}$ component, $C = 4$, $dt = 0.2$)



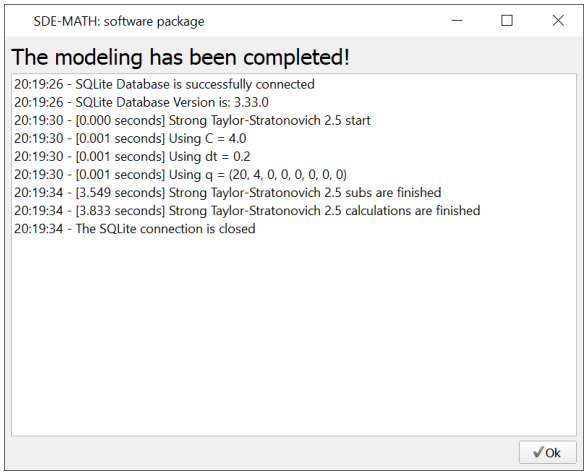
Strong Taylor–Stratonovich scheme of order 1.0 ($C = 4$, $dt = 0.2$)



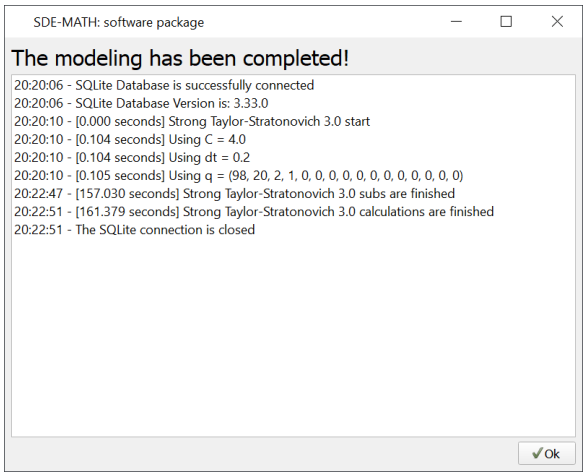
Strong Taylor–Stratonovich scheme of order 1.5 ($C = 4$, $dt = 0.2$)



Strong Taylor–Stratonovich scheme of order 2.0 ($C = 4$, $dt = 0.2$)



Strong Taylor–Stratonovich scheme of order 2.5 ($C = 4$, $dt = 0.2$)



Strong Taylor–Stratonovich scheme of order 3.0 ($C = 4$, $dt = 0.2$)

Figure 91: Modeling logs

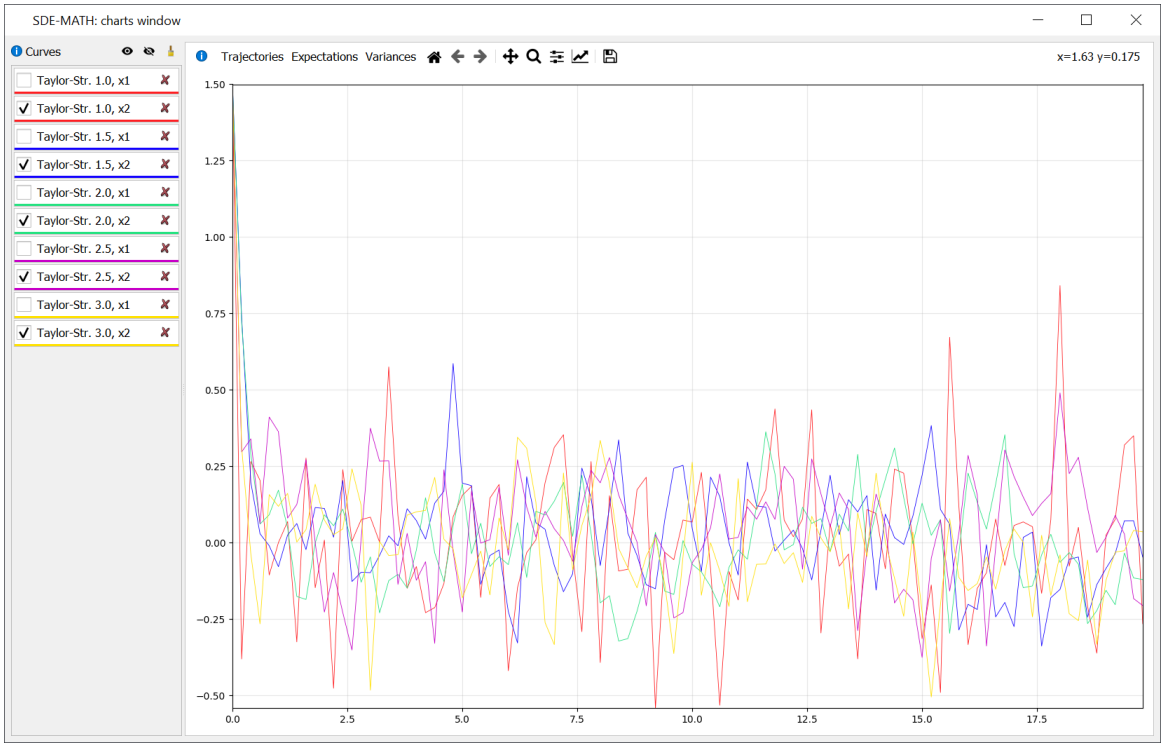


Figure 92: Strong Taylor–Stratonovich schemes of orders 1.0, 1.5, 2.0, 2.5, and 3.0 ($\mathbf{x}_t^{(2)}$ component, $C = 4$, $dt = 0.2$)

5.5 Example of Linear System of Itô SDEs (Solar Activity)

Consider a mathematical model of the solar activity without its average value in a form of the system of linear Itô SDEs (246) [4]. In (246) we choose [4] $n = 2$, $m = 1$, $k = 2$, $\mathbf{x}_0^{(1)} = 7$, $\mathbf{x}_0^{(2)} = -0.25$,

$$A = \begin{pmatrix} 0 & 1 \\ -0.3205 & -0.14 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \tag{250}$$

$$\mathbf{u}(t) \equiv \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad F = \begin{pmatrix} 0 \\ 5.08 \end{pmatrix}. \tag{251}$$

5.6 Visualization and Numerical Results for Solar Activity Model

This subsection is devoted to the visualization and numerical results for the model of solar activity (246), (250), (251).

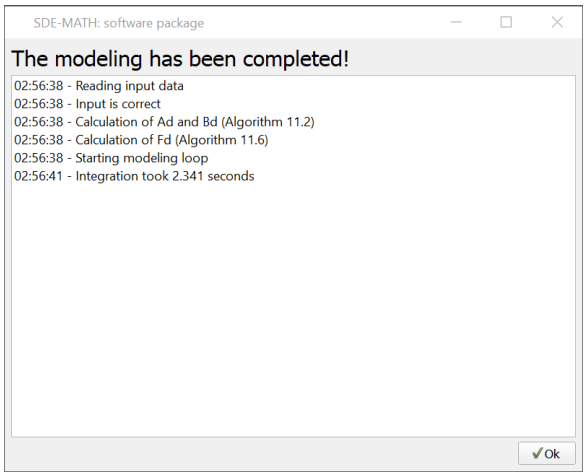


Figure 93: Modeling logs for solar activity model

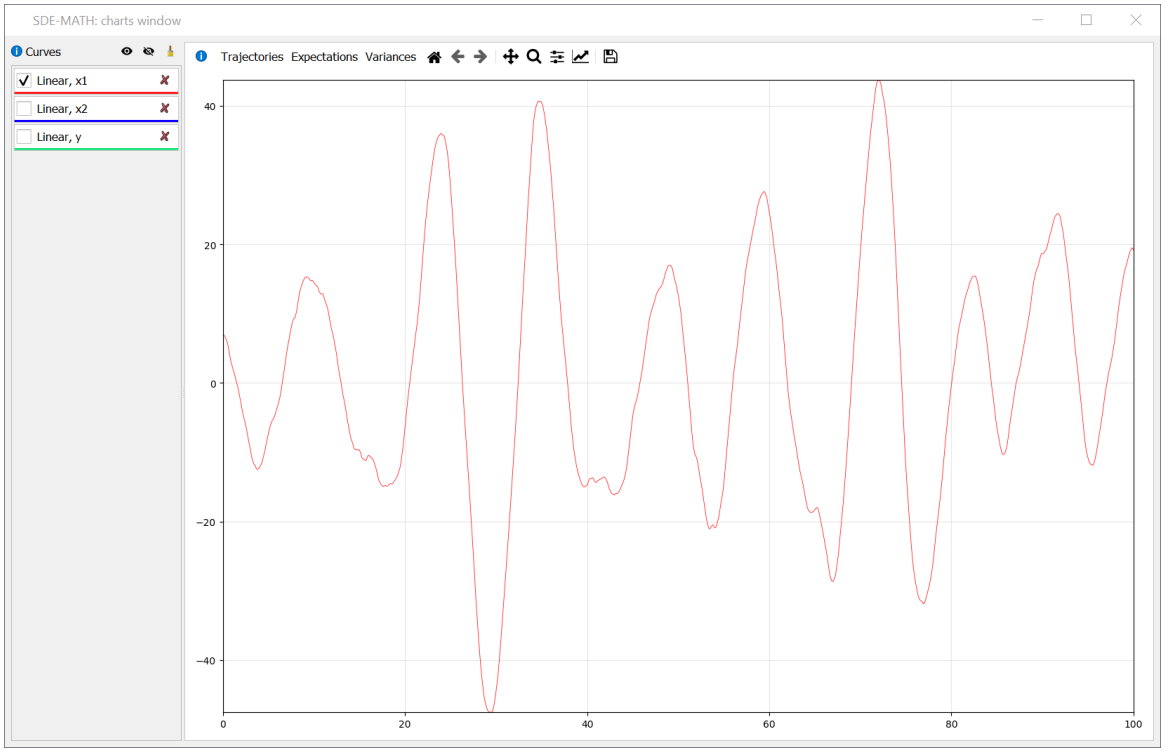


Figure 94: Solar activity model ($x_t^{(1)}$ component)

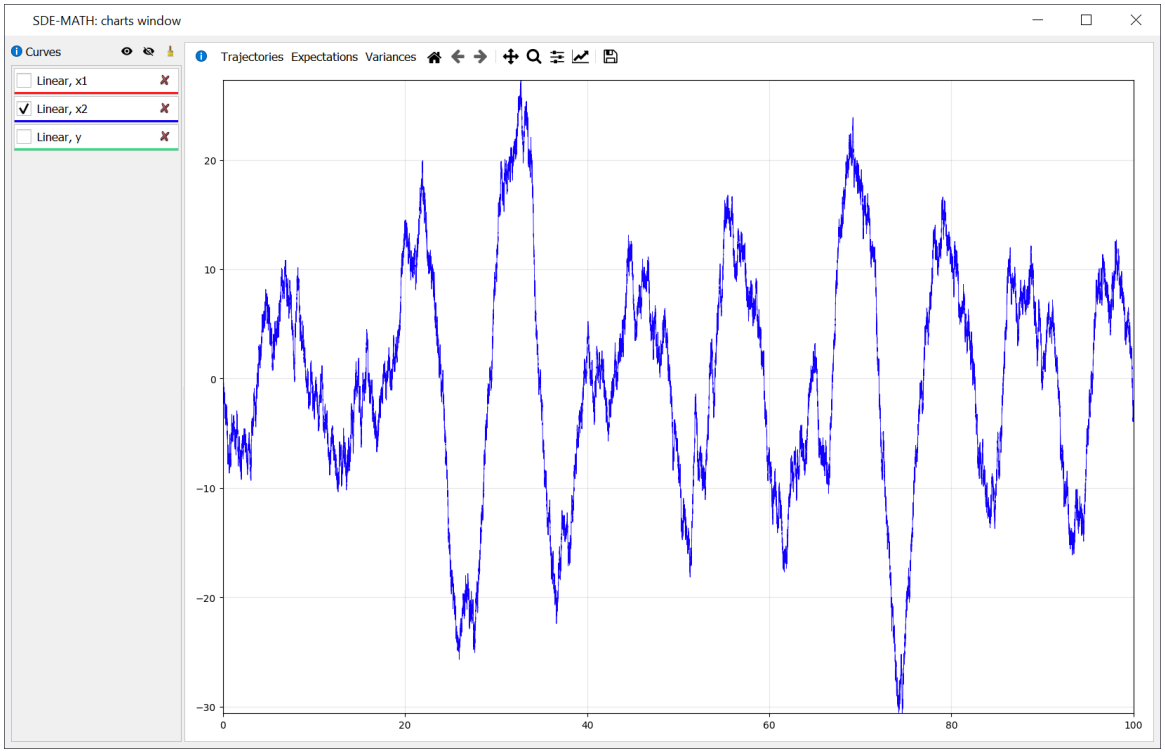


Figure 95: Solar activity model ($x_t^{(2)}$ component)

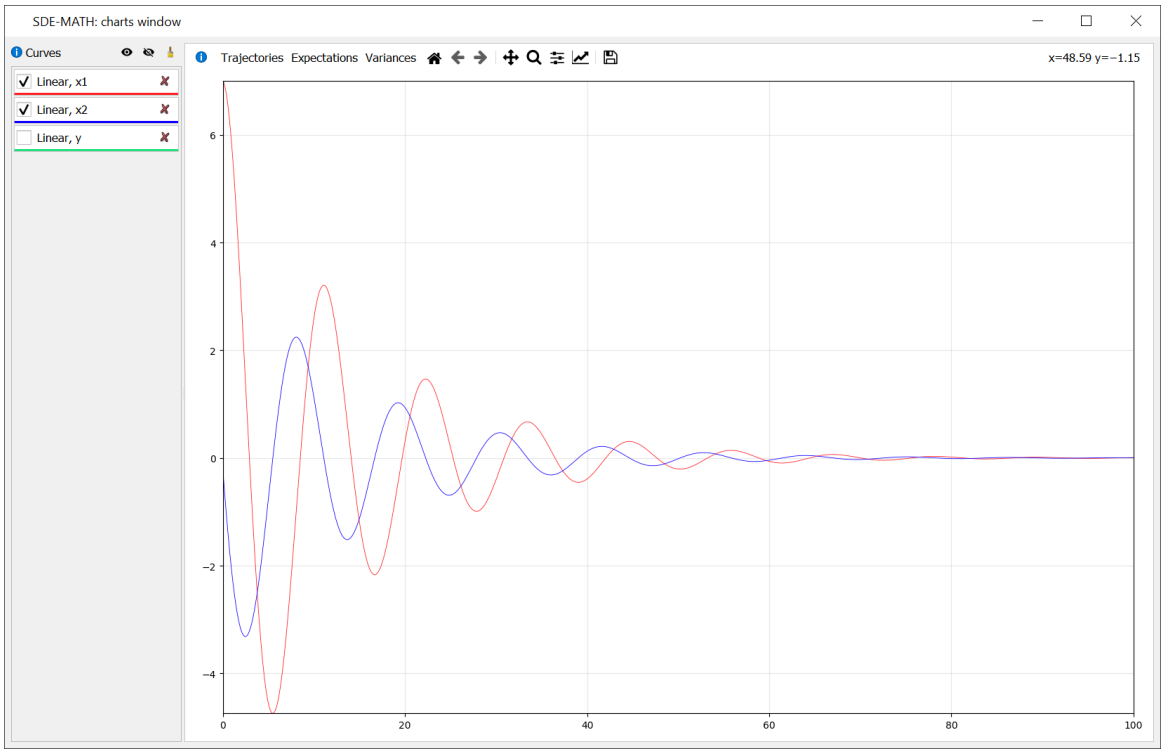


Figure 96: Solar activity model (expectations)

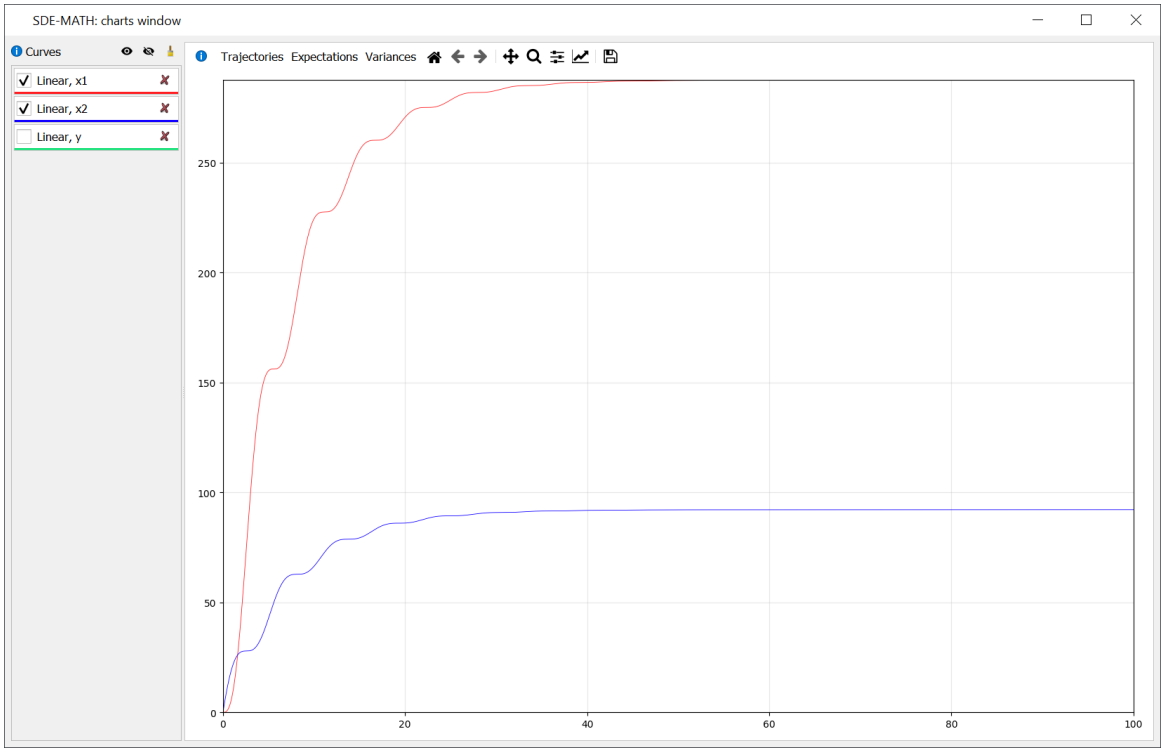


Figure 97: Solar activity model (variances)

5.7 Example of Abstract Linear System of Itô SDEs

Now consider the system of linear Itô SDEs (246) with the following data

$$n = 4, \quad m = 5, \quad k = 3, \quad \mathbf{x}_0^{(1)} = 1, \quad \mathbf{x}_0^{(2)} = 2, \quad \mathbf{x}_0^{(3)} = -1, \quad \mathbf{x}_0^{(4)} = -2, \tag{252}$$

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad F = \begin{pmatrix} 0.2 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.2 & 0.1 \end{pmatrix}, \tag{253}$$

$$\mathbf{u}(t) \equiv \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^{\mathsf{T}}, \quad H = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}. \tag{254}$$

5.8 Visualization and Numerical Results for Abstract Linear System of Itô SDEs Obtained via the SDE-MATH Software Package

This subsection is devoted to the visualization and numerical results for the model (246), (252)–(254).

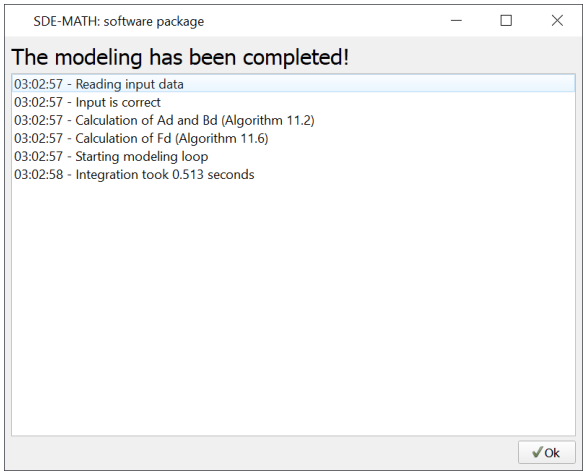


Figure 98: Modeling logs (linear system of Itô SDEs (246), (252)–(254))

Figure 99: Linear system of Itô SDEs (246), (252)–(254) (components of solution)

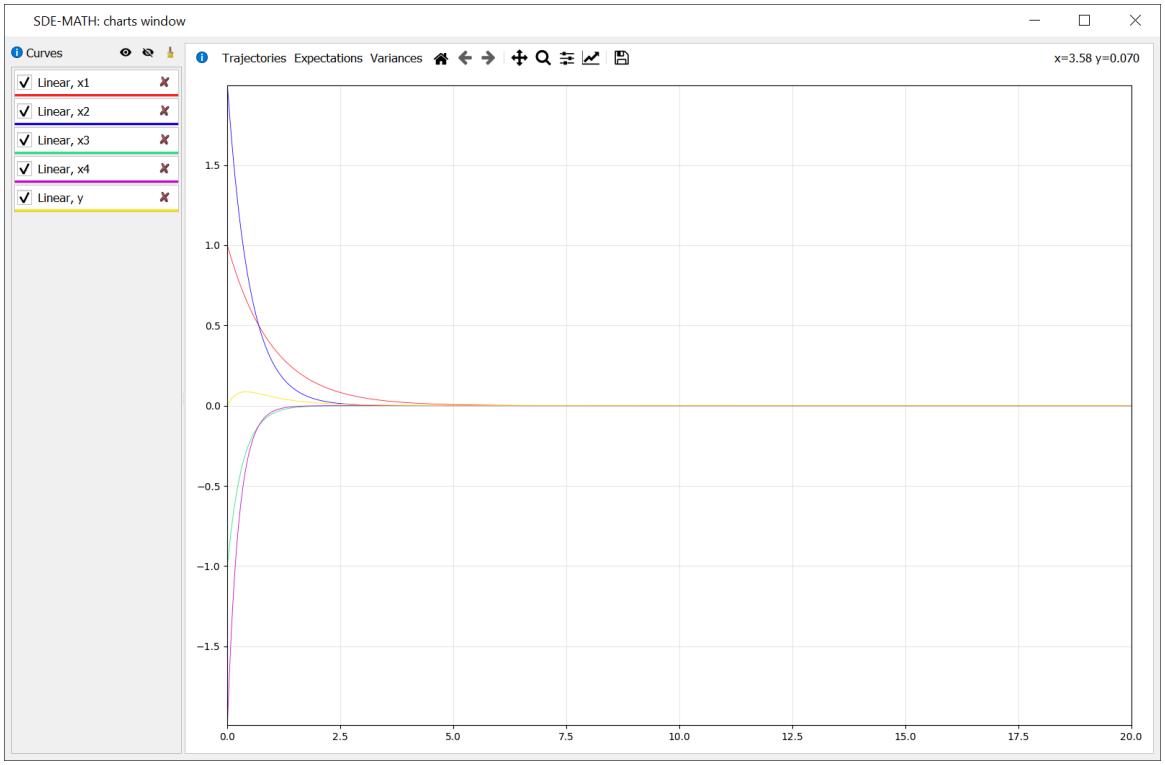


Figure 100: Linear system of Itô SDEs (246), (252)–(254) (expectations)

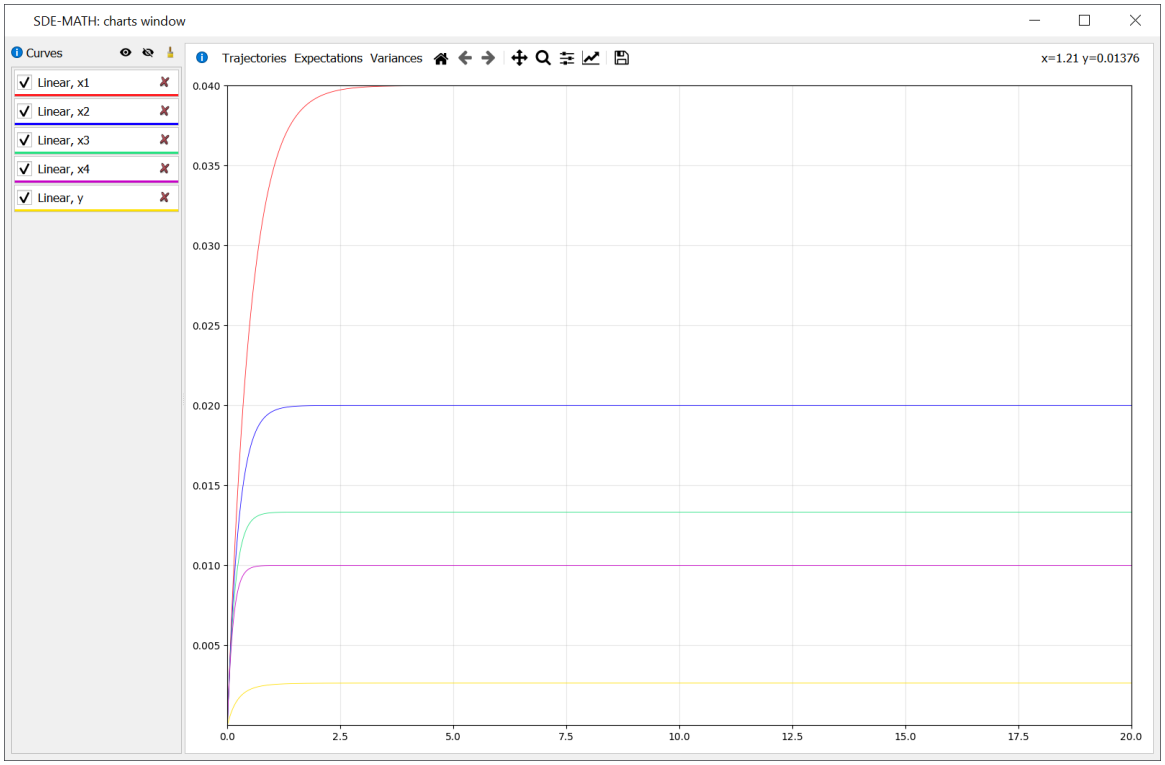


Figure 101: Linear system of Itô SDEs (246), (252)–(254) (variances)

6 Source Codes of the SDE-MATH Software Package in the Python Programming Language

6.1 Source Codes of Graphical User Interface

6.1.1 Source Codes of Main Menu

Listing 7: **Configuration file example**

```
"""
Configuration file example, change this paths to yours
"""

# Paths to resources
resources = "./resources/"

# Path to database
database = "./resources/database.db"

# Size of read buffer for the Fourier-Legendre coefficients
read_buffer_size = 8192

# Recursion limit for the Fourier-Legendre calculations
recursion_limit = 10 ** 8
```

Listing 8: **Program entry**

```
#!/usr/bin/env python
import logging
import os
import sys

from PyQt5 import QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWinExtras import QWinTaskbarButton

from config import database, images
from mathematics.sde.nonlinear.symbolic.coefficients.c import C
from tools.database import connect, disconnect
from ui.main.main_window import MainWindow


def main():

  logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
    datefmt="%H:%M:%S"
```

```
    )

    app = QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(os.path.join(images, "function.png")))
    app.setStyle(QtWidgets.QStyleFactory.create('Fusion'))

    main_window = MainWindow()

    main_window.taskbar_button = QWinTaskbarButton()
    main_window.taskbar_button.setOverlayIcon(QtGui.QIcon("resources/function.svg"))

    exit(app.exec())


if __name__ == "__main__":
    main()
```

Listing 9: **Main window**

```
from PyQt5.QtCore import QThreadPool, pyqtSignal
from PyQt5.QtWidgets import QStackedWidget, QMainWindow
from sympy.physics.mechanics.tests.test_system import lam

from init.initialization import initialization
from tools.fsys import is_locked
from ui.async_calls.worker import Worker
from ui.charts.charts_window import PlotWindow
from ui.main.greetings import GreetingsWidget
from ui.main.menu.base import MainMenuWidget
from ui.main.modeling.linear.base import LinearModelingWidget
from ui.main.modeling.nonliear.base import NonlinearModelingWidget
from ui.main.progress.complex_progress import ComplexProgressWidget
from ui.main.progress.simple_progress import SimpleProgressWidget


class MainWindow(QMainWindow):

    main_window_close = pyqtSignal()
    start_simple_progress = pyqtSignal(str)
    stop_simple_progress = pyqtSignal(str)

    def __init__(self):
        super(QMainWindow, self).__init__()

        self.plot_window = PlotWindow()

        self.stack_widget = QStackedWidget(self)

        self.main_menu = MainMenuWidget(self.stack_widget)
        self.complex_progress = ComplexProgressWidget(self.stack_widget)
        self.simple_progress = SimpleProgressWidget(self.stack_widget)
        self.greetings = GreetingsWidget(self.stack_widget)
        self.linear_modeling = LinearModelingWidget(self.stack_widget)
```

```
35      self.nonlinear_modeling = NonlinearModelingWidget(self.stack_widget)
36
37      self.stack_widget.addWidget(self.main_menu)
38      self.stack_widget.addWidget(self.nonlinear_modeling)
39      self.stack_widget.addWidget(self.linear_modeling)
40      self.stack_widget.addWidget(self.greetings)
41      self.stack_widget.addWidget(self.simple_progress)
42      self.stack_widget.addWidget(self.complex_progress)
43
44      self.setCentralWidget(self.stack_widget)
45
46      self.exec_init()
47
48      self.setWindowTitle("SDE-MATH: software package")
49      self.setMinimumSize(640, 480)
50      self.resize(800, 600)
51      self.show()
52
53      self.main_menu.group1.show_nonlinear_dialog.connect(self.show_nonlinear)
54      self.main_menu.group2.show_nonlinear_dialog.connect(self.show_nonlinear)
55      self.main_menu.group3.show_linear_dialog.connect(
56        lambda: self.stack_widget.setCurrentWidget(self.linear_modeling))
57
58      self.nonlinear_modeling.show_main_menu.connect(
59        lambda: self.stack_widget.setCurrentWidget(self.main_menu))
60      self.nonlinear_modeling.start_progress.connect(
61        lambda: self.stack_widget.setCurrentWidget(self.complex_progress))
62
63      self.linear_modeling.show_main_menu.connect(
64        lambda: self.stack_widget.setCurrentWidget(self.main_menu))
65      self.linear_modeling.start_progress.connect(
66        lambda: self.stack_widget.setCurrentWidget(self.complex_progress))
67
68      self.greetings.show_main_menu.connect(
69        lambda: self.stack_widget.setCurrentWidget(self.main_menu))
70
71      self.greetings.show_main_menu.connect(
72        lambda: self.stack_widget.setCurrentWidget(self.main_menu))
73
74      self.greetings.show_main_menu.connect(
75        lambda: self.stack_widget.setCurrentWidget(self.main_menu))
76
77      self.complex_progress.back_btn.clicked.connect(
78        lambda: self.stack_widget.setCurrentWidget(self.main_menu))
79
80      # plot events
81
82      self.main_window_close.connect(self.plot_window.close)
83
84      self.main_menu.charts_check.clicked.connect(self.plot_window.setVisible)
85      self.plot_window.charts_show.connect(
86        lambda: self.main_menu.charts_check.setChecked(True))
87      self.plot_window.charts_hide.connect(
88        lambda: self.main_menu.charts_check.setChecked(False))
89
```

```
90      self.nonlinear_modeling.draw_chart.connect(self.plot_window.charts_list.new_items)
91      self.nonlinear_modeling.draw_chart.connect(self.plot_window.plot_widget.new_items)
92      self.nonlinear_modeling.draw_chart.connect(self.plot_window.show)
93
94      self.nonlinear_modeling.charts_check.stateChanged.connect(self.plot_window.setVisible
        )
95      self.plot_window.charts_show.connect(
96        lambda: self.nonlinear_modeling.charts_check.setChecked(True))
97      self.plot_window.charts_hide.connect(
98        lambda: self.nonlinear_modeling.charts_check.setChecked(False))
99
100     self.linear_modeling.draw_chart.connect(self.plot_window.charts_list.new_items)
101     self.linear_modeling.draw_chart.connect(self.plot_window.plot_widget.new_items)
102     self.linear_modeling.draw_chart.connect(self.plot_window.show)
103
104     self.linear_modeling.charts_check.clicked.connect(self.plot_window.setVisible)
105     self.plot_window.charts_show.connect(
106       lambda: self.linear_modeling.charts_check.setChecked(True))
107     self.plot_window.charts_hide.connect(
108       lambda: self.linear_modeling.charts_check.setChecked(False))
109
110     self.linear_modeling.start_progress.connect(self.complex_progress.spin)
111     self.nonlinear_modeling.start_progress.connect(self.complex_progress.spin)
112     self.linear_modeling.stop_progress.connect(self.complex_progress.stop)
113     self.nonlinear_modeling.stop_progress.connect(self.complex_progress.stop)
114
115   def closeEvent(self, event):
116     self.main_window_close.emit()
117
118   def exec_init(self):
119     self.simple_progress.spin("Preparing the database...")
120     self.stack_widget.setCurrentWidget(self.simple_progress)
121
122     worker = Worker(initialization)
123     worker.signals.finished.connect(self.init_done)
124
125     QThreadPool.globalInstance().start(worker)
126
127   def init_done(self):
128     if not is_locked(".welcome.lock"):
129       self.stack_widget.setCurrentWidget(self.greetings)
130     else:
131       self.stack_widget.setCurrentWidget(self.main_menu)
132     self.simple_progress.stop()
133
134   def show_nonlinear(self, scheme_id):
135     self.nonlinear_modeling.set_scheme(scheme_id)
136     self.stack_widget.setCurrentWidget(self.nonlinear_modeling)
```

Listing 10: **Greetings window**

```
1  from PyQt5.QtCore import pyqtSignal, Qt
2  from PyQt5.QtWidgets import QPushButton, QVBoxLayout, QWidget, QSizePolicy, \
```

```
3     QSpacerItem, QHBoxLayout, QLabel, QCheckBox, QApplication, QStyle
4
5   from tools.fsys import lock, unlock
6   from ui.main.svg import SVG
7
8
9   class GreetingsWidget(QWidget):
10
11    show_main_menu = pyqtSignal()
12
13    def __init__(self, parent=None):
14      super(QWidget, self).__init__(parent)
15
16      header = QLabel("Welcome to SDE-MATH Software Package for "
17              "the Numerical Solution of Systems of Ito SDEs")
18      font = header.font()
19      font.setPointSize(15)
20      header.setAlignment(Qt.AlignJustify)
21      header.setWordWrap(True)
22      header.setFont(font)
23
24      welcome = QLabel(
25        "Exact solutions of Ito SDEs are known in rare cases. For this "
26        "reason, it becomes necessary to construct numerical methods for "
27        "Ito SDEs. Moreover, the problem of numerical solution of Ito SDEs "
28        "often occurs even in cases when the exact solution of Ito SDE is known. "
29        "This means that in some cases, knowing the exact solution to the Ito "
30        "SDE does not allow us to simulate it numerically in a simple way.", self
31      )
32
33      font = welcome.font()
34      welcome.setFont(font)
35
36      welcome.setAlignment(Qt.AlignJustify)
37      welcome.setWordWrap(True)
38      welcome.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Minimum))
39
40      check_again = QCheckBox("Do not show again", self)
41      next_btn = QPushButton("Ok", self)
42      next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_DialogApplyButton))
43
44      check_again.clicked.connect(self.check_lock)
45      next_btn.clicked.connect(lambda: self.show_main_menu.emit())
46
47      controls = QHBoxLayout()
48      controls.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
49      controls.addWidget(check_again)
50      controls.addWidget(next_btn)
51      controls.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
52
53      eq1 = QHBoxLayout()
54      eq1.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
55      eq1.addWidget(SVG("equation1.svg", scale_factor=1.))
56      eq1.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
57
```

```
58      eq2 = QHBoxLayout()
59      eq2.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
60      eq2.addWidget(SVG("equation2.svg", scale_factor=1.))
61      eq2.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
62
63      column = QVBoxLayout()
64      column.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
65      column.addWidget(header)
66      column.addItem(QSpacerItem(0, 15, QSizePolicy.Expanding, QSizePolicy.Minimum))
67      column.addLayout(eq1)
68      column.addItem(QSpacerItem(0, 15, QSizePolicy.Expanding, QSizePolicy.Minimum))
69      column.addLayout(eq2)
70      column.addItem(QSpacerItem(0, 15, QSizePolicy.Expanding, QSizePolicy.Minimum))
71      column.addWidget(welcome)
72      column.addItem(QSpacerItem(0, 30, QSizePolicy.Expanding, QSizePolicy.Minimum))
73      column.addLayout(controls)
74      column.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
75
76      layout = QHBoxLayout()
77      layout.addItem(QSpacerItem(50, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
78      layout.addLayout(column)
79      layout.addItem(QSpacerItem(50, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
80
81      self.setLayout(layout)
82
83    def check_lock(self):
84      if self.sender().isChecked():
85        lock(".welcome.lock")
86      else:
87        unlock(".welcome.lock")
```

Listing 11: **Info icon**

```
1   from PyQt5.QtCore import QSize
2   from PyQt5.QtWidgets import QWidget, QApplication, QStyle, QLabel
3
4
5   class InfoIcon(QLabel):
6
7     def __init__(self, text: str, parent=None):
8       super(QWidget, self).__init__(parent)
9
10      self.setToolTip(text)
11      self.setStyleSheet("QToolTip {background: white;}")
12      self.setPixmap(QApplication.style().standardIcon(
13        QStyle.SP_MessageBoxInformation).pixmap(QSize(16, 16)))
```

Listing 12: **Error widget**

```
1   from PyQt5.QtCore import QSize
2   from PyQt5.QtWidgets import QWidget, QApplication, QStyle, QLabel, QHBoxLayout,
      QSpacerItem, QSizePolicy
```

```
3
4
5  class ErrorWidget(QWidget):
6
7    def __init__(self, text: str, parent=None):
8      super(QWidget, self).__init__(parent)
9
10     msg_m = QLabel(text)
11     msg_m.setStyleSheet("QLabel { color: rgb(230, 0, 0); }")
12
13     msg_i = QLabel()
14     msg_i.setStyleSheet("QToolTip { background: white; }")
15     msg_i.setPixmap(QApplication.style().standardIcon(
16       QStyle.SP_MessageBoxCritical).pixmap(QSize(16, 16)))
17
18     layout = QHBoxLayout()
19     layout.setContentsMargins(0, 0, 0, 0)
20     layout.addWidget(msg_i)
21     layout.addWidget(msg_m)
22     layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
23
24     self.setLayout(layout)
```

Listing 13: **Svg picture**

```
1  import os
2
3  from PyQt5.QtCore import QSize
4  from PyQt5.QtSvg import QSvgWidget
5  from PyQt5.QtWidgets import QSizePolicy
6
7  from config import images
8
9
10 class SVG(QSvgWidget):
11
12   def __init__(self, name: str, scale_factor=1.):
13     super(QSvgWidget, self).__init__()
14
15     self.load(os.path.join(images, name))
16
17     self.scale_factor = scale_factor
18     self.setSizePolicy(QSizePolicy.Fixed, QSizePolicy.Fixed)
19
20   def sizeHint(self):
21     size = self.renderer().defaultSize()
22     return QSize(size.width() * self.scale_factor,
23             size.height() * self.scale_factor)
```

Listing 14: **Main menu (base part)**

```
1  from PyQt5.QtWidgets import QWidget, QSizePolicy, QSpacerItem, QHBoxLayout, QVBoxLayout,
```

```
    QCheckBox, QLabel

from ui.main.info import InfoIcon
from ui.main.menu.linear import LinearGroupWidget
from ui.main.menu.taylor_ito import ItoGroupWidget
from ui.main.menu.taylor_stratonovich import StratonovichGroupWidget


class MainMenuWidget(QWidget):

  def __init__(self, parent=None):
    super(QWidget, self).__init__(parent)

    self.charts_check = QCheckBox("Charts window", self)

    icon = InfoIcon("This is charts window checkbox, it will\n"
            "follow you on every application dialog, so\n"
            "you can easily open or close window with available charts")

    bar_layout = QHBoxLayout()
    bar_layout.addItem(QSpacerItem(0, 35, QSizePolicy.Expanding, QSizePolicy.Minimum))
    bar_layout.addWidget(icon)
    bar_layout.addWidget(self.charts_check)

    header = QLabel("Strong Numerical Schemes for Ito SDEs", parent=self)
    font = header.font()
    font.setPointSize(15)
    header.setFont(font)

    icon = InfoIcon("You are now in main menu, you can choose\n"
            "any scheme to perform modeling")

    header_layout = QHBoxLayout()
    header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding)
    )
    header_layout.addWidget(icon)
    header_layout.addWidget(header)
    header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding)
    )

    self.group3 = LinearGroupWidget(self)
    self.group1 = ItoGroupWidget(self)
    self.group2 = StratonovichGroupWidget(self)

    menu_layout = QHBoxLayout()
    menu_layout.addWidget(self.group1)
    menu_layout.addWidget(self.group2)
    menu_layout.addWidget(self.group3)

    layout = QVBoxLayout()
    layout.addLayout(bar_layout)
    layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
    layout.addLayout(header_layout)
    layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
    layout.addLayout(menu_layout)
```

```
54      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
55
56      self.setLayout(layout)
```

Listing 15: **Main menu (linear part)**

```
1  from PyQt5.QtCore import pyqtSignal
2  from PyQt5.QtWidgets import QPushButton, QVBoxLayout, QSizePolicy, QSpacerItem, QGroupBox
3
4
5  class LinearGroupWidget(QGroupBox):
6
7    show_linear_dialog = pyqtSignal()
8
9    def __init__(self, parent=None):
10     super(QGroupBox, self).__init__(parent)
11
12     linear_btn = QPushButton("Dispersion Spectral Decomposition")
13
14     layout = QVBoxLayout()
15     layout.addWidget(linear_btn)
16     layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
17
18     self.setLayout(layout)
19
20     self.setTitle("Linear Ito SDEs Systems Modeling")
21     self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding))
22
23     linear_btn.clicked.connect(lambda: self.show_linear_dialog.emit())
```

Listing 16: **Main menu (Taylor-Itô part)**

```
1  from PyQt5.QtCore import pyqtSignal
2  from PyQt5.QtWidgets import QPushButton, QVBoxLayout, QSizePolicy, QSpacerItem, QGroupBox
3
4
5  class ItoGroupWidget(QGroupBox):
6
7    show_nonlinear_dialog = pyqtSignal(int)
8
9    def __init__(self, parent=None):
10     super(QGroupBox, self).__init__(parent)
11
12     btn1 = QPushButton("Euler")
13     btn2 = QPushButton("Milstein")
14     btn3 = QPushButton("Convergence Order 1.5")
15     btn4 = QPushButton("Convergence Order 2.0")
16     btn5 = QPushButton("Convergence Order 2.5")
17     btn6 = QPushButton("Convergence Order 3.0")
18
19     layout = QVBoxLayout()
20     layout.addWidget(btn1)
```

```
21      layout.addWidget(btn2)
22      layout.addWidget(btn3)
23      layout.addWidget(btn4)
24      layout.addWidget(btn5)
25      layout.addWidget(btn6)
26      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
27
28      self.setLayout(layout)
29
30      self.setTitle("Taylor-Ito Schemes")
31      self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding))
32
33      btn1.clicked.connect(lambda: self.show_nonlinear_dialog.emit(0))
34      btn2.clicked.connect(lambda: self.show_nonlinear_dialog.emit(1))
35      btn3.clicked.connect(lambda: self.show_nonlinear_dialog.emit(2))
36      btn4.clicked.connect(lambda: self.show_nonlinear_dialog.emit(3))
37      btn5.clicked.connect(lambda: self.show_nonlinear_dialog.emit(4))
38      btn6.clicked.connect(lambda: self.show_nonlinear_dialog.emit(5))
```

Listing 17: **Main menu (Taylor–Stratonovich part)**

```
1   from PyQt5.QtCore import pyqtSignal
2   from PyQt5.QtWidgets import QPushButton, QVBoxLayout, QSizePolicy, QSpacerItem, QGroupBox
3
4
5   class StratonovichGroupWidget(QGroupBox):
6
7     show_nonlinear_dialog = pyqtSignal(int)
8
9     def __init__(self, parent=None):
10      super(QGroupBox, self).__init__(parent)
11
12      btn1 = QPushButton("Convergence Order 1.0")
13      btn2 = QPushButton("Convergence Order 1.5")
14      btn3 = QPushButton("Convergence Order 2.0")
15      btn4 = QPushButton("Convergence Order 2.5")
16      btn5 = QPushButton("Convergence Order 3.0")
17
18      layout = QVBoxLayout()
19      layout.addWidget(btn1)
20      layout.addWidget(btn2)
21      layout.addWidget(btn3)
22      layout.addWidget(btn4)
23      layout.addWidget(btn5)
24      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
25
26      self.setLayout(layout)
27
28      self.setTitle("Taylor-Stratonovich Schemes")
29      self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding))
30
31      btn1.clicked.connect(lambda: self.show_nonlinear_dialog.emit(6))
32      btn2.clicked.connect(lambda: self.show_nonlinear_dialog.emit(7))
```

```
33      btn3.clicked.connect(lambda: self.show_nonlinear_dialog.emit(8))
34      btn4.clicked.connect(lambda: self.show_nonlinear_dialog.emit(9))
35      btn5.clicked.connect(lambda: self.show_nonlinear_dialog.emit(10))
```

Listing 18: **Complex progress view**

```
1  import logging
2
3  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QSpacerItem, QSizePolicy,
       \
4    QListWidget, QLabel, QPushButton, QApplication, QStyle
5  from pyqtspinner.spinner import WaitingSpinner
6
7  from ui.main.progress.log_handler import LogHandler
8
9
10 class ComplexProgressWidget(QWidget):
11
12   def __init__(self, parent=None):
13     super(QWidget, self).__init__(parent)
14
15     self.spinner = WaitingSpinner(self,
16                     radius=5.0,
17                     lines=10,
18                     line_length=5.0,
19                     centerOnParent=False)
20     self.list_widget = QListWidget(self)
21     self.handler = LogHandler(self.handle_message)
22
23     self.label = QLabel(self)
24     font = self.label.font()
25     font.setPointSize(15)
26     self.label.setFont(font)
27
28     self.back_btn = QPushButton("Ok")
29     self.back_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_DialogApplyButton))
30     self.back_btn.hide()
31
32     spinner_layout = QHBoxLayout()
33     spinner_layout.addWidget(self.spinner)
34     spinner_layout.addWidget(self.label)
35     spinner_layout.addSpacerItem(QSpacerItem(0, 0, QSizePolicy.Expanding,
36                                 QSizePolicy.Minimum))
37
38     bottom_bar = QHBoxLayout()
39     bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
40     bottom_bar.addWidget(self.back_btn)
41
42     layout = QVBoxLayout()
43     layout.addLayout(spinner_layout)
44     layout.addWidget(self.list_widget)
45     layout.addLayout(bottom_bar)
46
```

```
47      self.setLayout(layout)
48
49    def handle_message(self, text):
50      self.list_widget.addItem(text)
51      self.list_widget.scrollToBottom()
52
53    def spin(self, text):
54      self.list_widget.clear()
55      logging.getLogger().addHandler(self.handler)
56      self.back_btn.hide()
57      self.spinner.start()
58      self.label.setText(text)
59
60    def stop(self, text):
61      self.back_btn.show()
62      self.list_widget.scrollToBottom()
63      self.label.setText(text)
64      self.spinner.stop()
65      logging.getLogger().removeHandler(self.handler)
```

Listing 19: **Simple progress view**

```
1   from PyQt5.QtCore import Qt
2   from PyQt5.QtWidgets import QVBoxLayout, QWidget, QLabel, QSpacerItem, QSizePolicy
3   from pyqtspinner.spinner import WaitingSpinner
4
5
6   class SimpleProgressWidget(QWidget):
7
8     def __init__(self, parent=None):
9       super(QWidget, self).__init__(parent)
10
11      self.spinner = WaitingSpinner(self, radius=15.0, lines=10, line_length=15.0)
12
13      self.label = QLabel()
14      self.label.setAlignment(Qt.AlignCenter)
15      font = self.label.font()
16      font.setPointSize(15)
17      self.label.setFont(font)
18
19      layout = QVBoxLayout()
20      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
21      layout.addWidget(self.spinner)
22      layout.addItem(QSpacerItem(0, 50, QSizePolicy.Minimum, QSizePolicy.Minimum))
23      layout.addWidget(self.label)
24      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding))
25
26      self.setLayout(layout)
27
28    def spin(self, text):
29      self.spinner.start()
30      self.label.setText(text)
31
```

```
32    def stop(self):
33      self.spinner.stop()
```

Listing 20: **Log handler for application**

```
1  import logging
2
3
4  class LogHandler(logging.Handler):
5
6    def __init__(self, callback):
7      super().__init__()
8      self.callback = callback
9      self.setFormatter(logging.Formatter("%(asctime)s - %(message)s", "%H:%M:%S"))
10
11   def handle(self, record):
12     self.callback(self.format(record))
```

Listing 21: **Matrix widget**

```
1  from PyQt5.QtWidgets import QTableWidgetItem, QTableWidget, QSizePolicy
2
3
4  class MatrixWidget(QTableWidget):
5
6    def __init__(self, parent=None):
7      super(QTableWidget, self).__init__(parent)
8
9      self.itemChanged.connect(self.item_changed)
10     self.m = [["0"]]
11
12     self.setRowCount(1)
13     self.setColumnCount(1)
14     self.setItem(0, 0, CustomItem("0"))
15
16     self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding))
17
18   def resize_w(self, w: int):
19
20     self.blockSignals(True)
21
22     old_w = self.columnCount()
23     self.setColumnCount(w)
24     h = self.rowCount()
25
26     self.m = [[self.m[i][j] if i < h and j < old_w else "0"
27           for j in range(w)]
28          for i in range(h)]
29
30     for i in range(h):
31       for j in range(w):
32         item = self.item(i, j)
```

```
33          if item is not None:
34            item.setText(self.m[i][j])
35          else:
36            self.setItem(i, j, CustomItem(self.m[i][j]))
37
38      self.blockSignals(False)
39
40    def resize_h(self, h: int):
41
42      self.blockSignals(True)
43
44      old_h = self.rowCount()
45      w = self.columnCount()
46      self.setRowCount(h)
47
48      self.m = [[self.m[i][j] if i < old_h and j < w else "0"
49            for j in range(w)]
50           for i in range(h)]
51
52      for i in range(h):
53        for j in range(w):
54          item = self.item(i, j)
55          if item is not None:
56            item.setText(self.m[i][j])
57          else:
58            self.setItem(i, j, CustomItem(self.m[i][j]))
59
60      self.blockSignals(False)
61
62    def item_changed(self, item):
63      self.m[item.row()][item.column()] = item.text()
64
65
66  class CustomItem(QTableWidgetItem):
67
68    def __init__(self, value: str):
69      super(QTableWidgetItem, self).__init__(value)
70
71      self.valid = True
```

6.1.2 Source Codes of Charts Window

Listing 22: **Charts window**

```
1  from PyQt5.QtCore import pyqtSignal
2  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QMainWindow, QSplitter
3
4  from ui.charts.side.available_charts_widget import AvailableChartsWidget
5  from ui.charts.visuals.charts_widget import ChartsWidget
6
7
```

```
class PlotWindow(QMainWindow):

  charts_show = pyqtSignal()
  charts_hide = pyqtSignal()

  def __init__(self):
    super(QMainWindow, self).__init__()

    self.plot_widget = ChartsWidget(self)
    self.charts_list = AvailableChartsWidget(self)

    splitter = QSplitter()
    splitter.addWidget(self.charts_list)
    splitter.addWidget(self.plot_widget)
    splitter.setSizes([splitter.width() / 0.85,
               splitter.width() / 0.15])

    layout = QHBoxLayout(self)
    layout.addWidget(splitter)

    central_widget = QWidget()
    central_widget.setLayout(layout)
    self.setCentralWidget(central_widget)

    self.setWindowTitle("SDE-MATH: charts window")
    self.resize(1200, 800)

    self.charts_list.on_show_all.connect(self.plot_widget.show_all)
    self.charts_list.on_hide_all.connect(self.plot_widget.hide_all)
    self.charts_list.on_remove_all.connect(self.plot_widget.delete_all)

  def showEvent(self, event):
    self.charts_show.emit()

  def closeEvent(self, event):
    self.charts_hide.emit()
```

Listing 23: **Curves list**

```
import os

from PyQt5 import QtGui
from PyQt5.QtCore import pyqtSignal
from PyQt5.QtWidgets import QVBoxLayout, QWidget, QSizePolicy, QSpacerItem, \
   QScrollArea, QLabel, QHBoxLayout, QPushButton, QApplication, QStyle

from config import images
from ui.charts.side.item_widget import ItemWidget
from ui.main.info import InfoIcon


class AvailableChartsWidget(QWidget):

```

```
15    on_hide_all = pyqtSignal()
16    on_show_all = pyqtSignal()
17    on_remove_all = pyqtSignal()
18
19    def __init__(self, parent=None):
20      super(QWidget, self).__init__(parent)
21
22      self.items = dict()
23
24      self.spacer = QSpacerItem(0, 0, QSizePolicy.Minimum, QSizePolicy.Expanding)
25      self.plot_widget = self.parent().plot_widget
26
27      remove_all = QPushButton()
28      remove_all.setFlat(True)
29      remove_all.setIcon(
30        QApplication.style().standardIcon(QStyle.SP_DialogResetButton))
31
32      hide_all = QPushButton()
33      hide_all.setFlat(True)
34      hide_all.setIcon(QtGui.QIcon(os.path.join(images, "crossed.png")))
35
36      show_all = QPushButton()
37      show_all.setFlat(True)
38      show_all.setIcon(QtGui.QIcon(os.path.join(images, "eye.png")))
39
40      header_layout = QHBoxLayout()
41      header_layout.addWidget(
42        InfoIcon("Here You will see all modeling series\n"
43             "You can hide them or delete, if you need to"))
44      header_layout.addItem(
45        QSpacerItem(5, 0, QSizePolicy.Minimum, QSizePolicy.Minimum))
46      header_layout.addWidget(QLabel("Curves"))
47      header_layout.addItem(
48        QSpacerItem(5, 0, QSizePolicy.Minimum, QSizePolicy.Minimum))
49      header_layout.setContentsMargins(0, 0, 0, 0)
50      header_layout.setSpacing(0)
51      header_layout.addItem(
52        QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
53      header_layout.addWidget(show_all)
54      header_layout.addWidget(hide_all)
55      header_layout.addWidget(remove_all)
56
57      self.layout = QVBoxLayout()
58      self.layout.setContentsMargins(3, 3, 3, 3)
59      self.layout.setSpacing(2)
60
61      scroll_widget = QWidget(self)
62      scroll_widget.setLayout(self.layout)
63
64      scroll_area = QScrollArea(self)
65      scroll_area.setWidgetResizable(True)
66      scroll_area.setWidget(scroll_widget)
67
68      self.layout.addItem(self.spacer)
69
```

```
70      layout = QVBoxLayout()
71      layout.setContentsMargins(0, 0, 0, 0)
72      layout.addLayout(header_layout)
73      layout.addWidget(scroll_area)
74
75      self.setLayout(layout)
76
77      self.setSizePolicy(
78        QSizePolicy(QSizePolicy.MinimumExpanding,
79              QSizePolicy.MinimumExpanding))
80
81      show_all.clicked.connect(self.show_all)
82      hide_all.clicked.connect(self.hide_all)
83      remove_all.clicked.connect(self.delete_all)
84
85    def new_items(self, lines: list):
86      self.layout.removeItem(self.spacer)
87
88      for i in range(len(lines)):
89        item_widget = ItemWidget(lines[i].name, lines[i].color, parent=self)
90        item_widget.uid = lines[i].uid
91        item_widget.on_show.connect(self.plot_widget.show_item)
92        item_widget.on_hide.connect(self.plot_widget.hide_item)
93        item_widget.on_delete.connect(self.plot_widget.delete_item)
94        item_widget.on_delete.connect(self.delete_item)
95        self.items[lines[i].uid] = item_widget
96
97        self.plot_widget.hide_label.connect(lambda uid: self.items[uid].hide())
98        self.plot_widget.show_label.connect(lambda uid: self.items[uid].show())
99
100       self.layout.addWidget(item_widget)
101
102     self.layout.addItem(self.spacer)
103
104   def delete_item(self):
105     s = self.sender()
106     s.setParent(None)
107     self.items.pop(s.uid)
108     self.layout.removeWidget(s)
109
110   def delete_all(self):
111     for item in self.items.values():
112       item.setParent(None)
113       self.layout.removeWidget(item)
114     self.items.clear()
115     self.on_remove_all.emit()
116
117   def hide_all(self):
118     for item in self.items.values():
119       item.checkbox.blockSignals(True)
120       item.checkbox.setChecked(False)
121       item.checkbox.blockSignals(False)
122     self.on_hide_all.emit()
123
124   def show_all(self):
```

```
125     for item in self.items.values():
126       item.checkbox.blockSignals(True)
127       item.checkbox.setChecked(True)
128       item.checkbox.blockSignals(False)
129     self.on_show_all.emit()
```

Listing 24: **Curves list item**

```
1  from PyQt5.QtCore import pyqtSignal
2  from PyQt5.QtWidgets import QPushButton, QSizePolicy, QHBoxLayout, QStyle, \
3    QApplication, QCheckBox, QVBoxLayout, QLabel, QSpacerItem, QFrame
4
5
6  class ItemWidget(QFrame):
7
8    on_delete = pyqtSignal(object)
9    on_hide = pyqtSignal(int)
10   on_show = pyqtSignal(int)
11
12   def __init__(self, name, color, parent=None):
13     super(QFrame, self).__init__(parent)
14
15     self.uid = 0
16
17     self.setFrameShape(QFrame.StyledPanel)
18     self.setStyleSheet("QFrame { background: white; }")
19
20     self.checkbox = QCheckBox(name)
21     self.checkbox.setChecked(True)
22
23     btn = QPushButton()
24     btn.setFlat(True)
25     btn.setIcon(QApplication.style().standardIcon(QStyle.SP_DialogCancelButton))
26
27     underline = QLabel()
28     underline.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Maximum))
29     underline.setMaximumHeight(3)
30     underline.setStyleSheet(f"QLabel {{ background: {color}; }}")
31
32     layout = QHBoxLayout()
33     layout.setContentsMargins(3, 3, 3, 3)
34     layout.setSpacing(0)
35     layout.addWidget(self.checkbox)
36     layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
37     layout.addWidget(btn)
38
39     layout_underlined = QVBoxLayout()
40     layout_underlined.setContentsMargins(0, 0, 0, 0)
41     layout_underlined.setSpacing(0)
42     layout_underlined.addLayout(layout)
43     layout_underlined.addWidget(underline)
44
45     self.setLayout(layout_underlined)
```

```
46
47      self.checkbox.stateChanged.connect(self.checkbox_changed)
48      btn.clicked.connect(lambda: self.on_delete.emit(self.uid))
49
50    def checkbox_changed(self, v):
51      if v > 0:
52        self.on_show.emit(self.uid)
53      else:
54        self.on_hide.emit(self.uid)
```

Listing 25: **Charts area**

```
1  import matplotlib.pyplot as plt
2  from PyQt5.QtCore import pyqtSignal
3  from PyQt5.QtWidgets import QSizePolicy, QFrame, QHBoxLayout, QPushButton, \
4    QSpacerItem
5  from PyQt5.QtWidgets import QVBoxLayout
6  from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
7
8  from ui.charts.visuals.color import Color
9  from ui.charts.visuals.toolbar import ToolBar
10 from ui.main.info import InfoIcon
11
12
13 class ChartsWidget(QFrame):
14
15   hide_label = pyqtSignal(int)
16   show_label = pyqtSignal(int)
17
18   def __init__(self, parent=None):
19     super(QFrame, self).__init__(parent)
20
21     self.plots = dict()
22     self.mode = 0
23
24     self.setFrameStyle(QFrame.StyledPanel)
25     self.setStyleSheet("QFrame { background: white; }")
26
27     self.figure = plt.figure()
28     self.canvas = FigureCanvas(self.figure)
29     self.canvas.mpl_connect('resize_event', self.on_resize)
30     self.toolbar = ToolBar(self.canvas, self)
31
32     self.ax = self.figure.add_subplot(111)
33     self.ax.margins(0)
34     self.ax.grid(axis='both', alpha=.3)
35     self.ax.relim(visible_only=True)
36     self.ax.autoscale()
37
38     self.figure.tight_layout()
39     self.canvas.draw()
40
41     self.btn_to_fn = QPushButton("Trajectories")
```

```
42      self.btn_to_fn.setFlat(True)
43      self.btn_to_mx = QPushButton("Expectations")
44      self.btn_to_mx.setFlat(True)
45      self.btn_to_dx = QPushButton("Variances")
46      self.btn_to_dx.setFlat(True)
47
48      toolbar_layout = QHBoxLayout()
49      toolbar_layout.setContentsMargins(0, 0, 0, 0)
50      toolbar_layout.setSpacing(0)
51      toolbar_layout.addItem(QSpacerItem(15, 0, QSizePolicy.Minimum, QSizePolicy.Minimum))
52      toolbar_layout.addWidget(InfoIcon("Click this buttons to switch plot modes\n"
53                      "between trajectories, expectations and variances"))
54      toolbar_layout.addItem(QSpacerItem(15, 0, QSizePolicy.Minimum, QSizePolicy.Minimum))
55      toolbar_layout.addWidget(self.btn_to_fn)
56      toolbar_layout.addWidget(self.btn_to_mx)
57      toolbar_layout.addWidget(self.btn_to_dx)
58      toolbar_layout.addWidget(self.toolbar)
59      toolbar_layout.addItem(QSpacerItem(15, 0, QSizePolicy.Minimum, QSizePolicy.Minimum))
60
61      layout = QVBoxLayout()
62      layout.setContentsMargins(0, 0, 0, 0)
63      layout.setSpacing(0)
64      layout.addLayout(toolbar_layout)
65      layout.addWidget(self.canvas)
66
67      self.setLayout(layout)
68
69      self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding))
70
71      self.btn_to_fn.pressed.connect(self.fn_mode)
72      self.btn_to_mx.pressed.connect(self.mx_mode)
73      self.btn_to_dx.pressed.connect(self.dx_mode)
74
75    def rescale(self):
76      self.ax.relim(visible_only=True)
77      self.ax.autoscale()
78      self.canvas.draw()
79
80    def clear(self):
81      for f in self.plots.values():
82        self.hide_label.emit(f.uid)
83        if f.line_fn is not None:
84          f.line_fn.set_visible(False)
85        if f.line_mx is not None:
86          f.line_mx.set_visible(False)
87        if f.line_dx is not None:
88          f.line_dx.set_visible(False)
89
90    def fn_mode(self):
91
92      self.mode = 0
93
94      self.clear()
95
96      for f in self.plots.values():
```

```
97        if f.line_fn is not None:
98          self.show_label.emit(f.uid)
99          if f.visible:
100           f.line_fn.set_visible(True)
101
102     self.rescale()
103
104   def mx_mode(self):
105
106     self.mode = 1
107
108     self.clear()
109
110     for f in self.plots.values():
111       if f.line_mx is not None:
112         self.show_label.emit(f.uid)
113         if f.visible:
114           f.line_mx.set_visible(True)
115
116     self.rescale()
117
118   def dx_mode(self):
119
120     self.mode = 2
121
122     self.clear()
123
124     for f in self.plots.values():
125       if f.line_dx is not None:
126         self.show_label.emit(f.uid)
127         if f.visible:
128           f.line_dx.set_visible(True)
129
130     self.rescale()
131
132   def new_items(self, lines: list):
133
134     for line in lines:
135       self.plots[line.uid] = line
136
137     for f in self.plots.values():
138       if f.line_fn is None and f.fn is not None:
139         f.line_fn = self.ax.plot(f.t, f.fn, linewidth=1, color=f.color)[0]
140       if f.line_mx is None and f.mx is not None:
141         f.line_mx = self.ax.plot(f.t, f.mx, linewidth=1, color=f.color)[0]
142       if f.line_dx is None and f.dx is not None:
143         f.line_dx = self.ax.plot(f.t, f.dx, linewidth=1, color=f.color)[0]
144
145     if self.mode == 0:
146       self.fn_mode()
147
148     if self.mode == 1:
149       self.mx_mode()
150
151     if self.mode == 2:
```

```
152         self.dx_mode()
153
154     def delete_item(self, uid: int):
155       item = self.plots.pop(uid)
156       Color.free(item.color)
157       if item.line_fn is not None:
158         item.line_fn.remove()
159       if item.line_mx is not None:
160         item.line_mx.remove()
161       if item.line_dx is not None:
162         item.line_dx.remove()
163
164       self.rescale()
165
166     def hide_item(self, uid: int):
167       item = self.plots[uid]
168       if item.line_fn is not None and self.mode == 0:
169         item.line_fn.set_visible(False)
170       if item.line_mx is not None and self.mode == 1:
171         item.line_mx.set_visible(False)
172       if item.line_dx is not None and self.mode == 2:
173         item.line_dx.set_visible(False)
174       item.visible = False
175
176       self.rescale()
177
178     def show_item(self, uid: int):
179       item = self.plots[uid]
180       if item.line_fn is not None and self.mode == 0:
181         item.line_fn.set_visible(True)
182       if item.line_mx is not None and self.mode == 1:
183         item.line_mx.set_visible(True)
184       if item.line_dx is not None and self.mode == 2:
185         item.line_dx.set_visible(True)
186       item.visible = True
187
188       self.rescale()
189
190     def show_all(self):
191       for item in self.plots.values():
192         if item.line_fn is not None and self.mode == 0:
193           item.line_fn.set_visible(True)
194         if item.line_mx is not None and self.mode == 1:
195           item.line_mx.set_visible(True)
196         if item.line_dx is not None and self.mode == 2:
197           item.line_dx.set_visible(True)
198         item.visible = True
199
200       self.rescale()
201
202     def hide_all(self):
203       for item in self.plots.values():
204         if item.line_fn is not None and self.mode == 0:
205           item.line_fn.set_visible(False)
206         if item.line_mx is not None and self.mode == 1:
```

```
207         item.line_mx.set_visible(False)
208       if item.line_dx is not None and self.mode == 2:
209         item.line_dx.set_visible(False)
210       item.visible = False
211
212     self.rescale()
213
214   def delete_all(self):
215     for item in reversed(self.plots.values()):
216       Color.free(item.color)
217       if item.line_fn is not None:
218         item.line_fn.remove()
219       if item.line_mx is not None:
220         item.line_mx.remove()
221       if item.line_dx is not None:
222         item.line_dx.remove()
223
224     self.plots.clear()
225
226     self.rescale()
227
228   def on_resize(self, event):
229     self.figure.tight_layout()
230     self.canvas.draw()
```

Listing 26: **Curves color**

```
1  from random import choice
2
3
4  class Color:
5
6    reserved_colors = [
7      "#ff834a",
8      "#ffe100",
9      "#c700c7",
10     "#24e280",
11     "#1100ff",
12     "#ff1e22",
13   ]
14   available_colors = [
15     "#ff834a",
16     "#ffe100",
17     "#c700c7",
18     "#24e280",
19     "#1100ff",
20     "#ff1e22",
21   ]
22
23   def __new__(cls, *args, **kwargs):
24
25     try:
26       return cls.available_colors.pop()
```

```
27
28      except IndexError:
29        return f"#{''.join([choice('0123456789ABCDEF') for j in range(6)])}"
30
31    @classmethod
32    def free(cls, code: str):
33
34      if code in cls.reserved_colors:
35        cls.available_colors.append(code)
```

Listing 27: **Curve**

```
1  from ui.charts.visuals.color import Color
2
3
4  class Line:
5    count = 0
6
7    def __init__(self, name, t, fn, mx=None, dx=None):
8      self.name = name
9      self.t = t
10     self.fn = fn
11     self.mx = mx
12     self.dx = dx
13     self.line_fn = None
14     self.line_mx = None
15     self.line_dx = None
16     self.visible = True
17     self.color = Color()
18
19     self.uid = Line.count
20     Line.count += 1
```

6.1.3 Source Codes of Input for Nonlinear Systems of Itô SDEs

Listing 28: **Base part of data input for nonlinear systems**

```
1  import logging
2
3  import numpy as np
4  from PyQt5.QtCore import QThreadPool, pyqtSignal
5  from PyQt5.QtWidgets import QCheckBox, QPushButton, QStyle, QApplication,\
6    QSizePolicy, QHBoxLayout, QSpacerItem, QVBoxLayout, QStackedWidget, \
7    QWidget, QLabel
8  from sympy import Matrix
9
10 import config
11 from mathematics.sde.nonlinear.drivers.euler import euler
12 from mathematics.sde.nonlinear.drivers.milstein import milstein
13 from mathematics.sde.nonlinear.drivers.strong_taylor_ito_1p5 import strong_taylor_ito_1p5
```

```
14 from mathematics.sde.nonlinear.drivers.strong_taylor_ito_2p0 import strong_taylor_ito_2p0
15 from mathematics.sde.nonlinear.drivers.strong_taylor_ito_2p5 import strong_taylor_ito_2p5
16 from mathematics.sde.nonlinear.drivers.strong_taylor_ito_3p0 import strong_taylor_ito_3p0
17 from mathematics.sde.nonlinear.drivers.strong_taylor_stratonovich_1p0 import
       strong_taylor_stratonovich_1p0
18 from mathematics.sde.nonlinear.drivers.strong_taylor_stratonovich_1p5 import
       strong_taylor_stratonovich_1p5
19 from mathematics.sde.nonlinear.drivers.strong_taylor_stratonovich_2p0 import
       strong_taylor_stratonovich_2p0
20 from mathematics.sde.nonlinear.drivers.strong_taylor_stratonovich_2p5 import
       strong_taylor_stratonovich_2p5
21 from mathematics.sde.nonlinear.drivers.strong_taylor_stratonovich_3p0 import
       strong_taylor_stratonovich_3p0
22 from mathematics.sde.nonlinear.symbolic.coefficients.c import C
23 from tools import database
24 from ui.async_calls.worker import Worker
25 from ui.charts.visuals.line import Line
26 from ui.main.modeling.nonliear.step1 import Step1
27 from ui.main.modeling.nonliear.step2 import Step2
28 from ui.main.modeling.nonliear.step3 import Step3
29 from ui.main.modeling.nonliear.step4 import Step4
30 from ui.main.modeling.nonliear.step5 import Step5
31
32
33 class NonlinearModelingWidget(QWidget):
34
35   show_main_menu = pyqtSignal()
36   start_progress = pyqtSignal(str)
37   stop_progress = pyqtSignal(str)
38   draw_chart = pyqtSignal(list)
39
40   def __init__(self, parent=None):
41     super(QWidget, self).__init__(parent)
42
43     self.logger = logging.getLogger(__name__)
44     self.scheme_id = 0
45     self.schemes = [
46       (euler, "Euler", "Euler Scheme"),
47       (milstein, "Milstein", "Milstein Scheme"),
48       (strong_taylor_ito_1p5, "Taylor-Ito 1.5",
49        "Strong Taylor-Ito Scheme with Convergence Order 1.5"),
50       (strong_taylor_ito_2p0, "Taylor-Ito 2.0",
51        "Strong Taylor-Ito Scheme with Convergence Order 2.0"),
52       (strong_taylor_ito_2p5, "Taylor-Ito 2.5",
53        "Strong Taylor-Ito Scheme with Convergence Order 2.5"),
54       (strong_taylor_ito_3p0, "Taylor-Ito 3.0",
55        "Strong Taylor-Ito Scheme with Convergence Order 3.0"),
56       (strong_taylor_stratonovich_1p0, "Taylor-Str. 1.0",
57        "Strong Taylor-Stratonovich Scheme with Convergence Order 1.0"),
58       (strong_taylor_stratonovich_1p5, "Taylor-Str. 1.5",
59        "Strong Taylor-Stratonovich Scheme with Convergence Order 1.5"),
60       (strong_taylor_stratonovich_2p0, "Taylor-Str. 2.0",
61        "Strong Taylor-Stratonovich Scheme with Convergence Order 2.0"),
62       (strong_taylor_stratonovich_2p5, "Taylor-Str. 2.5",
63        "Strong Taylor-Stratonovich Scheme with Convergence Order 2.5"),
```

```
64          (strong_taylor_stratonovich_3p0, "Taylor-Str. 3.0",
65           "Strong Taylor-Stratonovich Scheme with Convergence Order 3.0"),
66      ]
67
68      self.stack_widget = QStackedWidget(self)
69
70      self.step1 = Step1()
71      self.step2 = Step2()
72      self.step3 = Step3()
73      self.step4 = Step4()
74      self.step5 = Step5()
75
76      back_btn = QPushButton("Back", self)
77      back_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
78
79      self.charts_check = QCheckBox("Charts window", self)
80
81      self.scheme_name = QLabel()
82      self.scheme_name.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Minimum)
83
84      bar_layout = QHBoxLayout()
85      bar_layout.addWidget(back_btn)
86      bar_layout.addItem(QSpacerItem(10, 35, QSizePolicy.Minimum, QSizePolicy.Minimum))
87      bar_layout.addWidget(self.scheme_name)
88      bar_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
89      bar_layout.addWidget(self.charts_check)
90
91      self.stack_widget.addWidget(self.step1)
92      self.stack_widget.addWidget(self.step2)
93      self.stack_widget.addWidget(self.step3)
94      self.stack_widget.addWidget(self.step4)
95      self.stack_widget.addWidget(self.step5)
96
97      layout = QVBoxLayout()
98      layout.addLayout(bar_layout)
99      layout.addWidget(self.stack_widget)
100
101     self.setLayout(layout)
102
103     back_btn.clicked.connect(self.show_main_menu.emit)
104     back_btn.clicked.connect(
105       lambda: self.stack_widget.setCurrentWidget(self.step1))
106
107     self.step1.next_btn.clicked.connect(
108       lambda: self.stack_widget.setCurrentWidget(self.step2))
109     self.step2.prev_btn.clicked.connect(
110       lambda: self.stack_widget.setCurrentWidget(self.step1))
111     self.step2.next_btn.clicked.connect(
112       lambda: self.stack_widget.setCurrentWidget(self.step3))
113     self.step3.prev_btn.clicked.connect(
114       lambda: self.stack_widget.setCurrentWidget(self.step2))
115     self.step3.next_btn.clicked.connect(
116       lambda: self.stack_widget.setCurrentWidget(self.step4))
117     self.step4.prev_btn.clicked.connect(
118       lambda: self.stack_widget.setCurrentWidget(self.step3))
```

```
119     self.step4.next_btn.clicked.connect(
120       lambda: self.stack_widget.setCurrentWidget(self.step5))
121     self.step5.prev_btn.clicked.connect(
122       lambda: self.stack_widget.setCurrentWidget(self.step4))
123
124     self.step5.run_btn.clicked.connect(
125       lambda: self.run_modeling())
126     self.step5.run_btn.clicked.connect(
127       lambda: self.stack_widget.setCurrentWidget(self.step1))
128
129     self.step1.n_valid.connect(self.step2.matrix.resize_h)
130     self.step1.n_valid.connect(self.step3.matrix.resize_h)
131     self.step1.n_valid.connect(self.step4.matrix.resize_h)
132
133     self.step1.m_valid.connect(self.step3.matrix.resize_w)
134
135   def set_scheme(self, scheme_id):
136     self.scheme_id = scheme_id
137     self.scheme_name.setText(self.schemes[scheme_id][2])
138     if scheme_id == 0:
139       self.step5.count_c(False)
140     else:
141       self.step5.count_c(True)
142
143   def run_modeling(self):
144     self.start_progress.emit("The modeling is being performed...")
145
146     worker = Worker(self.routine)
147     worker.signals.result.connect(self.on_modeling_finish)
148     worker.signals.error.connect(self.on_modeling_corrupted)
149     QThreadPool.globalInstance().start(worker)
150
151   def routine(self):
152
153     scheme = self.schemes[self.scheme_id]
154
155     a = Matrix(self.step2.matrix.m)
156     b = Matrix(self.step3.matrix.m)
157
158     x0 = np.ndarray(shape=(self.step4.matrix.rowCount(),
159                 self.step4.matrix.columnCount()), dtype=float)
160     for i in range(self.step4.matrix.rowCount()):
161       for j in range(self.step4.matrix.columnCount()):
162         x0[i][j] = float(self.step4.matrix.m[i][j])
163
164     if self.step5.s != 0:
165       np.random.seed(self.step5.s)
166
167     database.connect(config.database)
168
169     if self.scheme_id == 0:
170       result = scheme[0](
171         x0, a, b,
172         (self.step5.t0,
173          self.step5.dt,
```

```
174             self.step5.t1)
175         )
176       else:
177         C.preload(56, 56, 56, 56, 56)
178         result = scheme[0](
179           x0, a, b, self.step5.c,
180           (self.step5.t0,
181            self.step5.dt,
182            self.step5.t1)
183         )
184
185       database.disconnect()
186
187       lines = [Line(f"{scheme[1]}, x{i + 1}",
188                np.array(result[1]).astype(float),
189                np.array(result[0][i, :]).astype(float))
190              for i in range(len(result[0]))]
191
192       return lines
193
194     def on_modeling_finish(self, result):
195       self.stop_progress.emit("The modeling has been completed!")
196       self.draw_chart.emit(result)
197
198     def on_modeling_corrupted(self, result):
199
200       self.logger.error(result[0])
201       self.logger.error(result[1])
202       self.logger.error(result[2])
203
204       self.stop_progress.emit("The modeling failed!")
```

Listing 29: **Step 1 of data input for nonlinear systems**

```
1  from PyQt5.QtCore import pyqtSignal
2  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QGridLayout, QLineEdit, QLabel, \
3    QVBoxLayout, QSpacerItem, QSizePolicy, QPushButton, QApplication, QStyle
4
5  from ui.main.error import ErrorWidget
6  from ui.main.info import InfoIcon
7  from ui.main.svg import SVG
8
9
10 class Step1(QWidget):
11
12   n_valid = pyqtSignal(int)
13   m_valid = pyqtSignal(int)
14
15   def __init__(self, parent=None):
16     super(QWidget, self).__init__(parent)
17
18     self.n_is_valid = False
19     self.m_is_valid = False
```

```
20
21      self.input_stack = self.parent()
22
23      info_n = InfoIcon("Dimension of linear system of Ito SDEs")
24      info_m = InfoIcon("Dimension of vector Wiener process")
25
26      label_n = QLabel("n")
27      label_m = QLabel("m")
28
29      self.lineedit_n = QLineEdit()
30      self.lineedit_m = QLineEdit()
31
32      self.msg_n = ErrorWidget("Wrong value!")
33      self.msg_n.hide()
34
35      self.msg_m = ErrorWidget("Wrong value!")
36      self.msg_m.hide()
37
38      grid_layout = QGridLayout()
39      grid_layout.addWidget(self.msg_n, 0, 2)
40      grid_layout.addWidget(self.msg_m, 2, 2)
41      grid_layout.addWidget(info_n, 1, 0)
42      grid_layout.addWidget(info_m, 3, 0)
43      grid_layout.addWidget(label_n, 1, 1)
44      grid_layout.addWidget(label_m, 3, 1)
45      grid_layout.addWidget(self.lineedit_n, 1, 2)
46      grid_layout.addWidget(self.lineedit_m, 3, 2)
47
48      header = QLabel("Dimensions settings", parent=self)
49      font = header.font()
50      font.setPointSize(15)
51      header.setFont(font)
52
53      self.next_btn = QPushButton("Next", self)
54      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
55      self.next_btn.setEnabled(False)
56
57      header_layout = QHBoxLayout()
58      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
59      header_layout.addWidget(header)
60      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
61
62      bottom_bar = QHBoxLayout()
63      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
64      bottom_bar.addWidget(self.next_btn)
65
66      eq1 = QHBoxLayout()
67      eq1.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
68      eq1.addWidget(SVG("equation1.svg", scale_factor=1.))
69      eq1.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
70
71      eq2 = QHBoxLayout()
72      eq2.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
73      eq2.addWidget(SVG("equation2.svg", scale_factor=1.))
74      eq2.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
```

```
75
76      equalities_layout = QVBoxLayout()
77      equalities_layout.addLayout(eq1)
78      equalities_layout.addItem(QSpacerItem(0, 20, QSizePolicy.Minimum, QSizePolicy.Minimum
        ))
79      equalities_layout.addLayout(eq2)
80
81      equalities_wrap = QHBoxLayout()
82      equalities_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum)
        )
83      equalities_wrap.addLayout(equalities_layout)
84      equalities_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum)
        )
85
86      grid_wrap = QHBoxLayout()
87      grid_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
88      grid_wrap.addLayout(grid_layout)
89      grid_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
90
91      layout = QVBoxLayout()
92      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Minimum, QSizePolicy.Expanding))
93      layout.addLayout(equalities_wrap)
94      layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum))
95      layout.addLayout(header_layout)
96      layout.addItem(QSpacerItem(0, 5, QSizePolicy.Minimum, QSizePolicy.Minimum))
97      layout.addLayout(grid_wrap)
98      layout.addItem(QSpacerItem(0, 0, QSizePolicy.Minimum, QSizePolicy.Expanding))
99      layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum))
100     layout.addLayout(bottom_bar)
101
102     self.setLayout(layout)
103
104     self.lineedit_n.textChanged.connect(self.validate_n)
105     self.lineedit_m.textChanged.connect(self.validate_m)
106
107   def validate_form(self):
108     if self.n_is_valid and self.m_is_valid:
109       self.next_btn.setEnabled(True)
110     else:
111       self.next_btn.setEnabled(False)
112
113   def validate_n(self, value):
114     try:
115       typed_value = int(value)
116       if typed_value <= 0:
117         raise ValueError()
118
119       self.n_is_valid = True
120       self.n_valid.emit(typed_value)
121       self.msg_n.hide()
122
123     except ValueError:
124       self.n_is_valid = False
125       self.msg_n.show()
126
```

```
127       finally:
128         self.validate_form()
129
130     def validate_m(self, value):
131       try:
132         typed_value = int(value)
133         if typed_value <= 0:
134           raise ValueError()
135
136         self.m_is_valid = True
137         self.m_valid.emit(typed_value)
138         self.msg_m.hide()
139
140       except ValueError:
141         self.m_is_valid = False
142         self.msg_m.show()
143
144       finally:
145         self.validate_form()
```

Listing 30: **Step 2 of data input for nonlinear systems**

```
1  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QLabel, QSpacerItem, QSizePolicy, \
2    QVBoxLayout, QPushButton, QApplication, QStyle
3
4  from ui.main.error import ErrorWidget
5  from ui.main.info import InfoIcon
6  from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9  class Step2(QWidget):
10
11   def __init__(self, parent=None):
12     super(QWidget, self).__init__(parent)
13
14     self.errors = 0
15
16     header = QLabel("Setting of column a(x, t)", parent=self)
17     font = header.font()
18     font.setPointSize(15)
19     header.setFont(font)
20
21     info = InfoIcon("Elements of column a(x, t) are expected to be functions\n"
22             "Size: n x 1\n"
23             "Functions must be set in python and SymPy notation")
24
25     self.msg = ErrorWidget("Wrong values in matrix!")
26     self.msg.hide()
27
28     self.matrix = MatrixWidget(self)
29
30     self.next_btn = QPushButton("Next", self)
31     self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
```

```
32
33      self.prev_btn = QPushButton("Back", self)
34      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36      header_layout = QHBoxLayout()
37      header_layout.addWidget(info)
38      header_layout.addWidget(header)
39      header_layout.addWidget(self.msg)
40      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42      bottom_bar = QHBoxLayout()
43      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44      bottom_bar.addWidget(self.prev_btn)
45      bottom_bar.addWidget(self.next_btn)
46
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
```

Listing 31: **Step 3 of data input for nonlinear systems**

```
1  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QLabel, QSpacerItem, QSizePolicy, \
2    QVBoxLayout, QPushButton, QApplication, QStyle
3
4  from ui.main.error import ErrorWidget
5  from ui.main.info import InfoIcon
6  from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9  class Step3(QWidget):
10
11   def __init__(self, parent=None):
12     super(QWidget, self).__init__(parent)
13
14     self.errors = 0
15
16     header = QLabel("Setting of matrix B(x, t)", parent=self)
17     font = header.font()
18     font.setPointSize(15)
19     header.setFont(font)
20
21     info = InfoIcon("Elements of matrix B(x, t) are expected to be functions\n"
22             "Size: n x m\n"
23             "Functions must be set in python and SymPy notation")
24
25     self.msg = ErrorWidget("Wrong values in matrix!")
26     self.msg.hide()
27
28     self.matrix = MatrixWidget(self)
29
```

```
30      self.next_btn = QPushButton("Next", self)
31      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33      self.prev_btn = QPushButton("Back", self)
34      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36      header_layout = QHBoxLayout()
37      header_layout.addWidget(info)
38      header_layout.addWidget(header)
39      header_layout.addWidget(self.msg)
40      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42      bottom_bar = QHBoxLayout()
43      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44      bottom_bar.addWidget(self.prev_btn)
45      bottom_bar.addWidget(self.next_btn)
46
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
```

Listing 32: **Step 4 of data input for nonlinear systems**

```
1   from PyQt5.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QLabel, \
2     QSpacerItem, QSizePolicy, QPushButton, QApplication, QStyle
3
4   from ui.main.error import ErrorWidget
5   from ui.main.info import InfoIcon
6   from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9   class Step4(QWidget):
10
11    def __init__(self, parent=None):
12      super(QWidget, self).__init__(parent)
13
14      self.errors = 0
15
16      header = QLabel("Setting of column x0", parent=self)
17      font = header.font()
18      font.setPointSize(15)
19      header.setFont(font)
20
21      info = InfoIcon("Elements of column x0 are expected to be functions\n"
22              "Size: n x 1\n"
23              "Functions must be set in python and SymPy notation")
24
25      self.msg = ErrorWidget("Wrong values in matrix!")
26      self.msg.hide()
27
```

```
28      self.matrix = MatrixWidget(self)
29
30      self.next_btn = QPushButton("Next", self)
31      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33      self.prev_btn = QPushButton("Back", self)
34      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36      header_layout = QHBoxLayout()
37      header_layout.addWidget(info)
38      header_layout.addWidget(header)
39      header_layout.addWidget(self.msg)
40      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42      bottom_bar = QHBoxLayout()
43      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44      bottom_bar.addWidget(self.prev_btn)
45      bottom_bar.addWidget(self.next_btn)
46
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
```

Listing 33: **Step 5 of data input for nonlinear systems**

```
1   import sys
2
3   from PyQt5.QtWidgets import QWidget, QHBoxLayout, QLabel, QLineEdit, \
4     QGridLayout, QVBoxLayout, QSizePolicy, QSpacerItem, QPushButton, \
5     QApplication, QStyle
6
7   from ui.main.error import ErrorWidget
8   from ui.main.info import InfoIcon
9
10
11  class Step5(QWidget):
12
13    def __init__(self, parent=None):
14      super(QWidget, self).__init__(parent)
15
16      self.t0 = sys.float_info.min
17      self.dt = 0
18      self.t1 = sys.float_info.max
19      self.s = 0
20      self.c = 0
21
22      self.t0_is_valid = False
23      self.dt_is_valid = False
24      self.t1_is_valid = False
25      self.s_is_valid = True
```

```
26      self.c_is_valid = False
27
28      self.input_stack = self.parent()
29
30      info_t0 = InfoIcon("Start point of integration interval\n"
31                "Must be in [0, t1) range")
32      info_dt = InfoIcon("Integration step\n"
33                "Must be set in (0, 1) interval")
34      info_t1 = InfoIcon("Final point of integration interval\n"
35                "Must be more then t0")
36      info_s = InfoIcon("This is random generator seed\n"
37               "If You do not want to use specific\n"
38               "seed just leave this field empty")
39      self.info_c = InfoIcon("The constant which defines approximation accuracy")
40
41      label_t0 = QLabel("t0")
42      label_dt = QLabel("dt")
43      label_t1 = QLabel("t1")
44      label_s = QLabel("seed")
45      self.label_c = QLabel("C")
46
47      self.lineedit_t0 = QLineEdit()
48      self.lineedit_dt = QLineEdit()
49      self.lineedit_t1 = QLineEdit()
50      self.lineedit_s = QLineEdit()
51      self.lineedit_c = QLineEdit()
52
53      self.msg_t0 = ErrorWidget("Wrong value!")
54      self.msg_t0.hide()
55
56      self.msg_dt = ErrorWidget("Wrong value!")
57      self.msg_dt.hide()
58
59      self.msg_t1 = ErrorWidget("Wrong value!")
60      self.msg_t1.hide()
61
62      self.msg_s = ErrorWidget("Wrong value!")
63      self.msg_s.hide()
64
65      self.msg_c = ErrorWidget("Wrong value!")
66      self.msg_c.hide()
67
68      header = QLabel("Accuracy settings", parent=self)
69      font = header.font()
70      font.setPointSize(15)
71      header.setFont(font)
72
73      self.prev_btn = QPushButton("Back", self)
74      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
75
76      self.run_btn = QPushButton("Perform modeling", self)
77      self.run_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
78      self.run_btn.setEnabled(False)
79
80      header_layout = QHBoxLayout()
```

```
81      header_layout.addWidget(header)
82      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
83
84      bottom_bar = QHBoxLayout()
85      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
86      bottom_bar.addWidget(self.prev_btn)
87      bottom_bar.addWidget(self.run_btn)
88
89      grid_layout = QGridLayout()
90
91      grid_layout.addWidget(self.msg_t0, 0, 2)
92
93      grid_layout.addWidget(info_t0, 1, 0)
94      grid_layout.addWidget(label_t0, 1, 1)
95      grid_layout.addWidget(self.lineedit_t0, 1, 2)
96
97      grid_layout.addWidget(self.msg_dt, 0, 5)
98
99      grid_layout.addWidget(info_dt, 1, 3)
100     grid_layout.addWidget(label_dt, 1, 4)
101     grid_layout.addWidget(self.lineedit_dt, 1, 5)
102
103     grid_layout.addWidget(self.msg_t1, 0, 8)
104
105     grid_layout.addWidget(info_t1, 1, 6)
106     grid_layout.addWidget(label_t1, 1, 7)
107     grid_layout.addWidget(self.lineedit_t1, 1, 8)
108
109     grid_layout.addWidget(self.msg_s, 2, 2)
110
111     grid_layout.addWidget(info_s, 3, 0)
112     grid_layout.addWidget(label_s, 3, 1)
113     grid_layout.addWidget(self.lineedit_s, 3, 2)
114
115     grid_layout.addWidget(self.msg_c, 2, 5)
116
117     grid_layout.addWidget(self.info_c, 3, 3)
118     grid_layout.addWidget(self.label_c, 3, 4)
119     grid_layout.addWidget(self.lineedit_c, 3, 5)
120
121     column_layout = QVBoxLayout()
122     column_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding)
        )
123     column_layout.addLayout(header_layout)
124     column_layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum))
125     column_layout.addLayout(grid_layout)
126     column_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding)
        )
127
128     control_layout = QHBoxLayout()
129     control_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding
        ))
130     control_layout.addLayout(column_layout)
131     control_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding
        ))
```

```
132
133     layout = QVBoxLayout()
134     layout.addLayout(control_layout)
135     layout.addLayout(bottom_bar)
136
137     self.setLayout(layout)
138
139     self.lineedit_t0.textChanged.connect(self.validate_t0)
140     self.lineedit_t0.textChanged.connect(self.validate_t1)
141     self.lineedit_t0.textChanged.connect(self.validate_dt)
142     self.lineedit_dt.textChanged.connect(self.validate_dt)
143     self.lineedit_t1.textChanged.connect(self.validate_t0)
144     self.lineedit_t1.textChanged.connect(self.validate_t1)
145     self.lineedit_t1.textChanged.connect(self.validate_dt)
146     self.lineedit_s.textChanged.connect(self.validate_s)
147     self.lineedit_c.textChanged.connect(self.validate_c)
148
149   def count_c(self, flag: bool):
150     if flag:
151       self.info_c.show()
152       self.label_c.show()
153       self.lineedit_c.show()
154       self.validate_c()
155     else:
156       self.info_c.hide()
157       self.label_c.hide()
158       self.lineedit_c.hide()
159       self.c_is_valid = True
160
161     self.msg_c.hide()
162
163   def validate_form(self):
164     if self.t0_is_valid \
165         and self.dt_is_valid \
166         and self.t1_is_valid \
167         and self.s_is_valid \
168         and self.c_is_valid:
169       self.run_btn.setEnabled(True)
170     else:
171       self.run_btn.setEnabled(False)
172
173   def validate_t0(self):
174     try:
175       typed_value = float(self.lineedit_t0.text())
176       if typed_value >= self.t1:
177         raise ValueError()
178
179       self.t0_is_valid = True
180       self.t0 = typed_value
181       self.msg_t0.hide()
182
183     except ValueError:
184       self.t0_is_valid = False
185       self.msg_t0.show()
186
```

```
187   def validate_dt(self):
188     try:
189       typed_value = float(self.lineedit_dt.text())
190       if typed_value <= 0 \
191           or (self.t1 - self.t0) / typed_value < 1 \
192           or typed_value >= 1:
193         raise ValueError()
194
195       self.dt_is_valid = True
196       self.dt = typed_value
197       self.msg_dt.hide()
198       self.validate_form()
199
200     except ValueError:
201       self.dt_is_valid = False
202       self.msg_dt.show()
203
204   def validate_t1(self):
205     try:
206       typed_value = float(self.lineedit_t1.text())
207       if typed_value <= self.t0:
208         raise ValueError()
209
210       self.t1_is_valid = True
211       self.t1 = typed_value
212       self.msg_t1.hide()
213
214     except ValueError:
215       self.t1_is_valid = False
216       self.msg_t1.show()
217
218   def validate_s(self):
219     try:
220       if self.lineedit_s.text() == "":
221         self.s_is_valid = True
222         self.s = 0
223       else:
224         typed_value = int(self.lineedit_s.text())
225         if typed_value <= 0:
226           raise ValueError()
227         self.s = typed_value
228
229       self.s_is_valid = True
230       self.msg_s.hide()
231
232     except ValueError:
233       self.s_is_valid = False
234       self.msg_s.show()
235
236     finally:
237       self.validate_form()
238
239   def validate_c(self):
240     try:
241       typed_value = float(self.lineedit_c.text())
```

```
242         if typed_value <= 0:
243           raise ValueError()
244 
245         self.c_is_valid = True
246         self.c = typed_value
247         self.msg_c.hide()
248 
249       except ValueError:
250         self.c_is_valid = False
251         self.msg_c.show()
252 
253       finally:
254         self.validate_form()
```

6.1.4 Source Codes of Input for Linear Systems of Itô SDEs

Listing 34: **Base part of data input for linear systems**

```
1  import logging
2  from time import time
3 
4  import numpy as np
5  from PyQt5.QtCore import QThreadPool, pyqtSignal
6  from PyQt5.QtWidgets import QCheckBox, QPushButton, QStyle, QApplication, \
7    QSizePolicy, QHBoxLayout, QSpacerItem, QVBoxLayout, QStackedWidget, \
8    QWidget, QLabel
9 
10 from mathematics.sde.linear.dindet import dindet
11 from mathematics.sde.linear.distortions import Symbolic, ComplexDistortion
12 from mathematics.sde.linear.integration import Integral
13 from mathematics.sde.linear.stoch import stoch
14 from ui.async_calls.worker import Worker
15 from ui.charts.visuals.line import Line
16 from ui.main.modeling.linear.step1 import Step1
17 from ui.main.modeling.linear.step2 import Step2
18 from ui.main.modeling.linear.step3 import Step3
19 from ui.main.modeling.linear.step4 import Step4
20 from ui.main.modeling.linear.step5 import Step5
21 from ui.main.modeling.linear.step6 import Step6
22 from ui.main.modeling.linear.step7 import Step7
23 from ui.main.modeling.linear.step8 import Step8
24 
25 
26 class LinearModelingWidget(QWidget):
27 
28   show_main_menu = pyqtSignal()
29   start_progress = pyqtSignal(str)
30   stop_progress = pyqtSignal(str)
31   draw_chart = pyqtSignal(list)
32 
33   def __init__(self, parent=None):
```

```
34      super(QWidget, self).__init__(parent)
35
36      self.logger = logging.getLogger(__name__)
37
38      self.stack_widget = QStackedWidget(self)
39
40      self.step1 = Step1()
41      self.step2 = Step2()
42      self.step3 = Step3()
43      self.step4 = Step4()
44      self.step5 = Step5()
45      self.step6 = Step6()
46      self.step7 = Step7()
47      self.step8 = Step8()
48
49      back_btn = QPushButton("Back", self)
50      back_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
51
52      self.charts_check = QCheckBox("Charts window", self)
53
54      self.scheme_name = QLabel("Linear Systems of Ito SDEs")
55
56      bar_layout = QHBoxLayout()
57      bar_layout.addWidget(back_btn)
58      bar_layout.addItem(QSpacerItem(10, 35, QSizePolicy.Minimum, QSizePolicy.Minimum))
59      bar_layout.addWidget(self.scheme_name)
60      bar_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
61      bar_layout.addWidget(self.charts_check)
62
63      self.stack_widget.addWidget(self.step1)
64      self.stack_widget.addWidget(self.step2)
65      self.stack_widget.addWidget(self.step3)
66      self.stack_widget.addWidget(self.step4)
67      self.stack_widget.addWidget(self.step5)
68      self.stack_widget.addWidget(self.step6)
69      self.stack_widget.addWidget(self.step7)
70      self.stack_widget.addWidget(self.step8)
71
72      layout = QVBoxLayout()
73      layout.addLayout(bar_layout)
74      layout.addWidget(self.stack_widget)
75
76      self.setLayout(layout)
77
78      back_btn.clicked.connect(self.show_main_menu.emit)
79      back_btn.clicked.connect(
80        lambda: self.stack_widget.setCurrentWidget(self.step1))
81
82      self.step1.next_btn.clicked.connect(
83        lambda: self.stack_widget.setCurrentWidget(self.step2))
84      self.step2.prev_btn.clicked.connect(
85        lambda: self.stack_widget.setCurrentWidget(self.step1))
86      self.step2.next_btn.clicked.connect(
87        lambda: self.stack_widget.setCurrentWidget(self.step3))
88      self.step3.prev_btn.clicked.connect(
```

```
89          lambda: self.stack_widget.setCurrentWidget(self.step2))
90      self.step3.next_btn.clicked.connect(
91          lambda: self.stack_widget.setCurrentWidget(self.step4))
92      self.step4.prev_btn.clicked.connect(
93          lambda: self.stack_widget.setCurrentWidget(self.step3))
94      self.step4.next_btn.clicked.connect(
95          lambda: self.stack_widget.setCurrentWidget(self.step5))
96      self.step5.prev_btn.clicked.connect(
97          lambda: self.stack_widget.setCurrentWidget(self.step4))
98      self.step5.next_btn.clicked.connect(
99          lambda: self.stack_widget.setCurrentWidget(self.step6))
100     self.step6.prev_btn.clicked.connect(
101         lambda: self.stack_widget.setCurrentWidget(self.step5))
102     self.step6.next_btn.clicked.connect(
103         lambda: self.stack_widget.setCurrentWidget(self.step7))
104     self.step7.prev_btn.clicked.connect(
105         lambda: self.stack_widget.setCurrentWidget(self.step6))
106     self.step7.next_btn.clicked.connect(
107         lambda: self.stack_widget.setCurrentWidget(self.step8))
108     self.step8.prev_btn.clicked.connect(
109         lambda: self.stack_widget.setCurrentWidget(self.step7))
110
111     self.step8.run_btn.clicked.connect(
112         lambda: self.run_modeling())
113     self.step8.run_btn.clicked.connect(
114         lambda: self.stack_widget.setCurrentWidget(self.step1))
115
116     self.step1.n_valid.connect(self.step2.matrix.resize_h)
117     self.step1.n_valid.connect(self.step2.matrix.resize_w)
118
119     self.step1.n_valid.connect(self.step3.matrix.resize_h)
120     self.step1.k_valid.connect(self.step3.matrix.resize_w)
121
122     self.step1.n_valid.connect(self.step4.matrix.resize_h)
123     self.step1.m_valid.connect(self.step4.matrix.resize_w)
124
125     self.step1.k_valid.connect(self.step5.matrix.resize_h)
126
127     self.step1.n_valid.connect(self.step6.matrix.resize_w)
128
129     self.step1.n_valid.connect(self.step7.matrix.resize_h)
130
131 def run_modeling(self):
132     self.start_progress.emit("The modeling is being performed...")
133
134     worker = Worker(self.routine)
135     worker.signals.result.connect(self.on_modeling_finish)
136     worker.signals.error.connect(self.on_modeling_corrupted)
137     QThreadPool.globalInstance().start(worker)
138
139 def routine(self):
140     n = int(self.step1.lineedit_n.text())
141     m = int(self.step1.lineedit_m.text())
142     k = int(self.step1.lineedit_k.text())
143     t0 = float(self.step8.lineedit_t0.text())
```

```
144     dt = float(self.step8.lineedit_dt.text())
145     t1 = float(self.step8.lineedit_t1.text())
146 
147     self.logger.info("Reading input data")
148 
149     integral = Integral(n)
150 
151     integral.k, integral.m, integral.dt, integral.t0, integral.tk = \
152       k, m, dt, t0, t1
153 
154     integral.m_a = np.array([[float(self.step2.matrix.m[i][j])
155                   for j in range(self.step2.matrix.columnCount())]
156                   for i in range(self.step2.matrix.rowCount())])
157 
158     integral.mat_b = np.array([[float(self.step3.matrix.m[i][j])
159                    for j in range(self.step3.matrix.columnCount())]
160                    for i in range(self.step3.matrix.rowCount())])
161 
162     integral.mat_f = np.array([[float(self.step4.matrix.m[i][j])
163                    for j in range(self.step4.matrix.columnCount())]
164                    for i in range(self.step4.matrix.rowCount())])
165 
166     integral.m_h = np.array([[float(self.step6.matrix.m[i][j])
167                   for j in range(self.step6.matrix.columnCount())]
168                   for i in range(self.step6.matrix.rowCount())])
169 
170     integral.m_x0 = np.array([[float(self.step7.matrix.m[i][j])
171                    for j in range(self.step7.matrix.columnCount())]
172                   for i in range(self.step7.matrix.rowCount())])
173 
174     integral.m_mx0 = np.array([[float(self.step7.matrix.m[i][j])
175                    for j in range(self.step7.matrix.columnCount())]
176                    for i in range(self.step7.matrix.rowCount())])
177 
178     integral.m_dx0 = np.zeros((integral.n, integral.n))
179 
180     self.logger.info("Input is correct")
181     self.logger.info("Calculation of Ad and Bd (Algorithm 11.2)")
182 
183     integral.m_ad, integral.m_bd = dindet(
184       integral.n, integral.k, integral.m_a, integral.mat_b, integral.dt)
185 
186     self.logger.info("Calculation of Fd (Algorithm 11.6)")
187 
188     integral.m_fd = stoch(integral.n, integral.m_a, integral.mat_f, integral.dt)
189 
190     mat_u = np.array([[object]] * self.step5.matrix.rowCount())
191     for i in range(self.step5.matrix.rowCount()):
192       mat_u[i][0] = Symbolic(self.step5.matrix.m[i][0])
193 
194     integral.distortion = ComplexDistortion(self.step5.matrix.rowCount(), mat_u)
195 
196     self.logger.info("Starting modeling loop")
197 
198     start_time = time()
```

```
199     integral.integrate()
200
201     self.logger.info(f"Integration took {(time() - start_time):.3f} seconds")
202
203     lines = [Line(f"Linear, x{i + 1}",
204             np.array(integral.v_t).astype(float),
205             np.array(integral.m_xt[i, :]).astype(float),
206             mx=np.array(integral.m_mx[i, :]).astype(float),
207             dx=np.array(integral.m_dx[i, :]).astype(float))
208           for i in range(integral.m_xt.shape[0])]
209
210     name = f"Linear"
211
212     lines.append(Line(f"Linear, y",
213               np.array(integral.v_t).astype(float),
214               np.array(integral.v_yt).astype(float),
215               mx=np.array(integral.v_my).astype(float),
216               dx=np.array(integral.v_dy).astype(float)))
217
218     return lines
219
220   def on_modeling_finish(self, result):
221     self.stop_progress.emit("The modeling has been completed!")
222     self.draw_chart.emit(result)
223
224   def on_modeling_corrupted(self, result):
225     self.logger.error(result[0])
226     self.logger.error(result[1])
227     self.logger.error(result[2])
228
229     self.stop_progress.emit("The modeling failed!")
```

Listing 35: **Step 1 of data input for linear systems**

```
1  from PyQt5.QtCore import pyqtSignal
2  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QGridLayout, QLineEdit, QLabel, \
3    QVBoxLayout, QSpacerItem, QSizePolicy, QPushButton, QApplication, QStyle
4
5  from ui.main.error import ErrorWidget
6  from ui.main.info import InfoIcon
7  from ui.main.svg import SVG
8
9
10 class Step1(QWidget):
11
12   n_valid = pyqtSignal(int)
13   m_valid = pyqtSignal(int)
14   k_valid = pyqtSignal(int)
15
16   def __init__(self, parent=None):
17     super(QWidget, self).__init__(parent)
18
19     self.n_is_valid = False
```

```
20      self.m_is_valid = False
21      self.k_is_valid = False
22
23      self.input_stack = self.parent()
24
25      info_n = InfoIcon("Dimension of linear system of Ito SDEs")
26      info_m = InfoIcon("Dimension of vector Wiener process")
27      info_k = InfoIcon("Dimension of vector function u(t)")
28
29      label_n = QLabel("n")
30      label_m = QLabel("m")
31      label_k = QLabel("k")
32
33      self.lineedit_n = QLineEdit()
34      self.lineedit_m = QLineEdit()
35      self.lineedit_k = QLineEdit()
36
37      self.msg_n = ErrorWidget("Wrong value!")
38      self.msg_n.hide()
39
40      self.msg_m = ErrorWidget("Wrong value!")
41      self.msg_m.hide()
42
43      self.msg_k = ErrorWidget("Wrong value!")
44      self.msg_k.hide()
45
46      grid_layout = QGridLayout()
47      grid_layout.addWidget(self.msg_n, 0, 2)
48      grid_layout.addWidget(self.msg_m, 2, 2)
49      grid_layout.addWidget(self.msg_k, 4, 2)
50      grid_layout.addWidget(info_n, 1, 0)
51      grid_layout.addWidget(info_m, 3, 0)
52      grid_layout.addWidget(info_k, 5, 0)
53      grid_layout.addWidget(label_n, 1, 1)
54      grid_layout.addWidget(label_m, 3, 1)
55      grid_layout.addWidget(label_k, 5, 1)
56      grid_layout.addWidget(self.lineedit_n, 1, 2)
57      grid_layout.addWidget(self.lineedit_m, 3, 2)
58      grid_layout.addWidget(self.lineedit_k, 5, 2)
59
60      header = QLabel("Dimensions settings", parent=self)
61      font = header.font()
62      font.setPointSize(15)
63      header.setFont(font)
64
65      self.next_btn = QPushButton("Next", self)
66      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
67      self.next_btn.setEnabled(False)
68
69      header_layout = QHBoxLayout()
70      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
71      header_layout.addWidget(header)
72      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
73
74      bottom_bar = QHBoxLayout()
```

```
75      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
76      bottom_bar.addWidget(self.next_btn)
77
78      eq1 = QHBoxLayout()
79      eq1.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
80      eq1.addWidget(SVG("equation3.svg", scale_factor=1.))
81      eq1.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
82
83      eq2 = QHBoxLayout()
84      eq2.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
85      eq2.addWidget(SVG("equation4.svg", scale_factor=1.))
86      eq2.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
87
88      equalities_layout = QVBoxLayout()
89      equalities_layout.addLayout(eq1)
90      equalities_layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum
        ))
91      equalities_layout.addLayout(eq2)
92
93      equalities_wrap = QHBoxLayout()
94      equalities_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum)
        )
95      equalities_wrap.addLayout(equalities_layout)
96      equalities_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum)
        )
97
98      grid_wrap = QHBoxLayout()
99      grid_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
100     grid_wrap.addLayout(grid_layout)
101     grid_wrap.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
102
103     layout = QVBoxLayout()
104     layout.addItem(QSpacerItem(0, 0, QSizePolicy.Minimum, QSizePolicy.Expanding))
105     layout.addLayout(equalities_wrap)
106     layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum))
107     layout.addLayout(header_layout)
108     layout.addItem(QSpacerItem(0, 5, QSizePolicy.Minimum, QSizePolicy.Minimum))
109     layout.addLayout(grid_wrap)
110     layout.addItem(QSpacerItem(0, 0, QSizePolicy.Minimum, QSizePolicy.Expanding))
111     layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum))
112     layout.addLayout(bottom_bar)
113
114     self.setLayout(layout)
115
116     self.lineedit_n.textChanged.connect(self.validate_n)
117     self.lineedit_m.textChanged.connect(self.validate_m)
118     self.lineedit_k.textChanged.connect(self.validate_k)
119
120   def validate_form(self):
121     if self.n_is_valid and self.m_is_valid and self.k_is_valid:
122       self.next_btn.setEnabled(True)
123     else:
124       self.next_btn.setEnabled(False)
125
126   def validate_n(self, value):
```

```
127     try:
128       typed_value = int(value)
129       if typed_value <= 0:
130         raise ValueError()
131 
132       self.n_is_valid = True
133       self.n_valid.emit(typed_value)
134       self.msg_n.hide()
135 
136     except ValueError:
137       self.n_is_valid = False
138       self.msg_n.show()
139 
140     finally:
141       self.validate_form()
142 
143   def validate_m(self, value):
144     try:
145       typed_value = int(value)
146       if typed_value <= 0:
147         raise ValueError()
148 
149       self.m_is_valid = True
150       self.m_valid.emit(typed_value)
151       self.msg_m.hide()
152 
153     except ValueError:
154       self.m_is_valid = False
155       self.msg_m.show()
156 
157     finally:
158       self.validate_form()
159 
160   def validate_k(self, value):
161     try:
162       typed_value = int(value)
163       if typed_value <= 0:
164         raise ValueError()
165 
166       self.k_is_valid = True
167       self.k_valid.emit(typed_value)
168       self.msg_k.hide()
169 
170     except ValueError:
171       self.k_is_valid = False
172       self.msg_k.show()
173 
174     finally:
175       self.validate_form()
```

Listing 36: **Step 2 of data input for linear systems**

```
1 from PyQt5.QtWidgets import QWidget, QHBoxLayout, QLabel, QSpacerItem, QSizePolicy, \
```

```
2   QVBoxLayout, QPushButton, QApplication, QStyle
3
4 from ui.main.error import ErrorWidget
5 from ui.main.info import InfoIcon
6 from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9 class Step2(QWidget):
10
11   def __init__(self, parent=None):
12     super(QWidget, self).__init__(parent)
13
14     self.errors = 0
15
16     header = QLabel("Setting of matrix A", parent=self)
17     font = header.font()
18     font.setPointSize(15)
19     header.setFont(font)
20
21     info = InfoIcon("Elements of matrix A are\n"
22             "expected to be real values\n"
23             "Size: n x n")
24
25     self.msg = ErrorWidget("Wrong values in matrix!")
26     self.msg.hide()
27
28     self.matrix = MatrixWidget(self)
29
30     self.next_btn = QPushButton("Next", self)
31     self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33     self.prev_btn = QPushButton("Back", self)
34     self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36     header_layout = QHBoxLayout()
37     header_layout.addWidget(info)
38     header_layout.addWidget(header)
39     header_layout.addWidget(self.msg)
40     header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42     bottom_bar = QHBoxLayout()
43     bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44     bottom_bar.addWidget(self.prev_btn)
45     bottom_bar.addWidget(self.next_btn)
46
47     layout = QVBoxLayout()
48     layout.addLayout(header_layout)
49     layout.addWidget(self.matrix)
50     layout.addLayout(bottom_bar)
51
52     self.setLayout(layout)
53
54     self.matrix.itemChanged.connect(self.validate_item)
55
56   def validate_item(self, item):
```

```
57
58      value = item.text()
59      try:
60        float(value)
61        if not item.valid:
62          item.valid = True
63          self.errors -= 1
64
65      except ValueError:
66        if item.valid:
67          item.valid = False
68          self.errors += 1
69
70      finally:
71        self.validate_form()
72
73    def validate_form(self):
74
75      if self.errors == 0:
76        self.msg.hide()
77        self.next_btn.setEnabled(True)
78      else:
79        self.msg.show()
80        self.next_btn.setEnabled(False)
```

Listing 37: **Step 3 of data input for linear systems**

```
1  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QLabel, QSpacerItem, QSizePolicy, \
2    QVBoxLayout, QPushButton, QApplication, QStyle
3
4  from ui.main.error import ErrorWidget
5  from ui.main.info import InfoIcon
6  from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9  class Step3(QWidget):
10
11   def __init__(self, parent=None):
12     super(QWidget, self).__init__(parent)
13
14     self.errors = 0
15
16     header = QLabel("Setting of matrix B", parent=self)
17     font = header.font()
18     font.setPointSize(15)
19     header.setFont(font)
20
21     info = InfoIcon("Elements of matrix B are\n"
22             "expected to be real values\n"
23             "Size: n x k")
24
25     self.msg = ErrorWidget("Wrong values in matrix!")
26     self.msg.hide()
```

```
27
28      self.matrix = MatrixWidget(self)
29
30      self.next_btn = QPushButton("Next", self)
31      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33      self.prev_btn = QPushButton("Back", self)
34      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36      header_layout = QHBoxLayout()
37      header_layout.addWidget(info)
38      header_layout.addWidget(header)
39      header_layout.addWidget(self.msg)
40      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42      bottom_bar = QHBoxLayout()
43      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44      bottom_bar.addWidget(self.prev_btn)
45      bottom_bar.addWidget(self.next_btn)
46
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
53
54      self.matrix.itemChanged.connect(self.validate_item)
55
56    def validate_item(self, item):
57
58      value = item.text()
59      try:
60        float(value)
61        if not item.valid:
62          item.valid = True
63          self.errors -= 1
64
65      except ValueError:
66        if item.valid:
67          item.valid = False
68          self.errors += 1
69
70      finally:
71        self.validate_form()
72
73    def validate_form(self):
74
75      if self.errors == 0:
76        self.msg.hide()
77        self.next_btn.setEnabled(True)
78      else:
79        self.msg.show()
80        self.next_btn.setEnabled(False)
```

Listing 38: **Step 4 of data input for linear systems**

```
from PyQt5.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QLabel, QSpacerItem, \
  QSizePolicy, QPushButton, QApplication, QStyle

from ui.main.error import ErrorWidget
from ui.main.info import InfoIcon
from ui.main.modeling.matrix_widget import MatrixWidget


class Step4(QWidget):

  def __init__(self, parent=None):
    super(QWidget, self).__init__(parent)

    self.errors = 0

    header = QLabel("Setting of matrix F", parent=self)
    font = header.font()
    font.setPointSize(15)
    header.setFont(font)

    info = InfoIcon("Elements of matrix F are\n"
            "expected to be real values\n"
            "Size: n x m")

    self.msg = ErrorWidget("Wrong values in matrix!")
    self.msg.hide()

    self.matrix = MatrixWidget(self)

    self.next_btn = QPushButton("Next", self)
    self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))

    self.prev_btn = QPushButton("Back", self)
    self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))

    header_layout = QHBoxLayout()
    header_layout.addWidget(info)
    header_layout.addWidget(header)
    header_layout.addWidget(self.msg)
    header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))

    bottom_bar = QHBoxLayout()
    bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
    bottom_bar.addWidget(self.prev_btn)
    bottom_bar.addWidget(self.next_btn)

    layout = QVBoxLayout()
    layout.addLayout(header_layout)
    layout.addWidget(self.matrix)
    layout.addLayout(bottom_bar)

    self.setLayout(layout)

```

```
54      self.matrix.itemChanged.connect(self.validate_item)
55
56    def validate_item(self, item):
57
58      value = item.text()
59      try:
60        float(value)
61        if not item.valid:
62          item.valid = True
63          self.errors -= 1
64
65      except ValueError:
66        if item.valid:
67          item.valid = False
68          self.errors += 1
69
70      finally:
71        self.validate_form()
72
73    def validate_form(self):
74
75      if self.errors == 0:
76        self.msg.hide()
77        self.next_btn.setEnabled(True)
78      else:
79        self.msg.show()
80        self.next_btn.setEnabled(False)
```

Listing 39: **Step 5 of data input for linear systems**

```
1   from PyQt5.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QLabel, QSpacerItem, \
2     QSizePolicy, QPushButton, QApplication, QStyle
3
4   from ui.main.error import ErrorWidget
5   from ui.main.info import InfoIcon
6   from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9   class Step5(QWidget):
10
11    def __init__(self, parent=None):
12      super(QWidget, self).__init__(parent)
13
14      self.errors = 0
15
16      header = QLabel("Setting of vector function u(t)", parent=self)
17      font = header.font()
18      font.setPointSize(15)
19      header.setFont(font)
20
21      info = InfoIcon("Elements of vector u(t) are expected to be functions\n"
22              "Size: k x 1\n"
23              "Functions must be set in python and SymPy notation")
```

```
24
25      self.msg = ErrorWidget("Wrong values in matrix!")
26      self.msg.hide()
27
28      self.matrix = MatrixWidget(self)
29
30      self.next_btn = QPushButton("Next", self)
31      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33      self.prev_btn = QPushButton("Back", self)
34      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36      header_layout = QHBoxLayout()
37      header_layout.addWidget(info)
38      header_layout.addWidget(header)
39      header_layout.addWidget(self.msg)
40      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42      bottom_bar = QHBoxLayout()
43      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44      bottom_bar.addWidget(self.prev_btn)
45      bottom_bar.addWidget(self.next_btn)
46
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
```

Listing 40: **Step 6 of data input for linear systems**

```
1  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QLabel, QSpacerItem, \
2    QSizePolicy, QPushButton, QApplication, QStyle
3
4  from ui.main.error import ErrorWidget
5  from ui.main.info import InfoIcon
6  from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9  class Step6(QWidget):
10
11   def __init__(self, parent=None):
12     super(QWidget, self).__init__(parent)
13
14     self.errors = 0
15
16     header = QLabel("Setting of matrix H", parent=self)
17     font = header.font()
18     font.setPointSize(15)
19     header.setFont(font)
20
21     info = InfoIcon("Elements of matrix H are\n"
```

```
22              "expected to be int values\n"
23              "Size: 1 x n")
24
25      self.msg = ErrorWidget("Wrong values in matrix!")
26      self.msg.hide()
27
28      self.matrix = MatrixWidget(self)
29
30      self.next_btn = QPushButton("Next", self)
31      self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33      self.prev_btn = QPushButton("Back", self)
34      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36      header_layout = QHBoxLayout()
37      header_layout.addWidget(info)
38      header_layout.addWidget(header)
39      header_layout.addWidget(self.msg)
40      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42      bottom_bar = QHBoxLayout()
43      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44      bottom_bar.addWidget(self.prev_btn)
45      bottom_bar.addWidget(self.next_btn)
46
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
53
54      self.matrix.itemChanged.connect(self.validate_item)
55
56  def validate_item(self, item):
57
58      value = item.text()
59      try:
60          float(value)
61          if not item.valid:
62              item.valid = True
63              self.errors -= 1
64
65      except ValueError:
66          if item.valid:
67              item.valid = False
68              self.errors += 1
69
70      finally:
71          self.validate_form()
72
73  def validate_form(self):
74
75      if self.errors == 0:
76          self.msg.hide()
```

```
77          self.next_btn.setEnabled(True)
78        else:
79          self.msg.show()
80          self.next_btn.setEnabled(False)
```

Listing 41: **Step 7 of data input for linear systems**

```
1  from PyQt5.QtWidgets import QWidget, QHBoxLayout, QVBoxLayout, QLabel, QSpacerItem, \
2    QSizePolicy, QPushButton, QApplication, QStyle
3
4  from ui.main.error import ErrorWidget
5  from ui.main.info import InfoIcon
6  from ui.main.modeling.matrix_widget import MatrixWidget
7
8
9  class Step7(QWidget):
10
11   def __init__(self, parent=None):
12     super(QWidget, self).__init__(parent)
13
14     self.errors = 0
15
16     header = QLabel("Setting of column x0", parent=self)
17     font = header.font()
18     font.setPointSize(15)
19     header.setFont(font)
20
21     info = InfoIcon("Elements of column x0 are\n"
22             "expected to be real values\n"
23             "Size: n x 1")
24
25     self.msg = ErrorWidget("Wrong values in matrix!")
26     self.msg.hide()
27
28     self.matrix = MatrixWidget(self)
29
30     self.next_btn = QPushButton("Next", self)
31     self.next_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
32
33     self.prev_btn = QPushButton("Back", self)
34     self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
35
36     header_layout = QHBoxLayout()
37     header_layout.addWidget(info)
38     header_layout.addWidget(header)
39     header_layout.addWidget(self.msg)
40     header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
41
42     bottom_bar = QHBoxLayout()
43     bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
44     bottom_bar.addWidget(self.prev_btn)
45     bottom_bar.addWidget(self.next_btn)
46
```

```
47      layout = QVBoxLayout()
48      layout.addLayout(header_layout)
49      layout.addWidget(self.matrix)
50      layout.addLayout(bottom_bar)
51
52      self.setLayout(layout)
53
54      self.matrix.itemChanged.connect(self.validate_item)
55
56    def validate_item(self, item):
57
58      value = item.text()
59      try:
60        float(value)
61        if not item.valid:
62          item.valid = True
63          self.errors -= 1
64
65      except ValueError:
66        if item.valid:
67          item.valid = False
68          self.errors += 1
69
70      finally:
71        self.validate_form()
72
73    def validate_form(self):
74
75      if self.errors == 0:
76        self.msg.hide()
77        self.next_btn.setEnabled(True)
78      else:
79        self.msg.show()
80        self.next_btn.setEnabled(False)
```

Listing 42: **Step 8 of data input for linear systems**

```
1   import sys
2
3   from PyQt5.QtWidgets import QWidget, QHBoxLayout, QLabel, QLineEdit, QGridLayout, \
4     QVBoxLayout, QSizePolicy, QSpacerItem, QPushButton, QApplication, QStyle
5
6   from ui.main.error import ErrorWidget
7   from ui.main.info import InfoIcon
8
9
10  class Step8(QWidget):
11
12    def __init__(self, parent=None):
13      super(QWidget, self).__init__(parent)
14
15      self.t0 = sys.float_info.min
16      self.dt = 0
```

```
17      self.t1 = sys.float_info.max
18      self.s = 0
19
20      self.t0_is_valid = False
21      self.dt_is_valid = False
22      self.t1_is_valid = False
23      self.s_is_valid = True
24
25      self.input_stack = self.parent()
26
27      info_t0 = InfoIcon("Start point of integration interval\n"
28                "Must be in [0, t1) range")
29      info_dt = InfoIcon("Integration step\n"
30                "Must be set in (0, 1) interval")
31      info_t1 = InfoIcon("Final point of integration interval\n"
32                "Must be more then t0")
33      info_s = InfoIcon("This is random generator seed\n"
34               "If You do not want to use specific\n"
35               "seed just leave this field empty")
36
37      label_t0 = QLabel("t0")
38      label_dt = QLabel("dt")
39      label_t1 = QLabel("t1")
40      label_s = QLabel("seed")
41
42      self.lineedit_t0 = QLineEdit()
43      self.lineedit_dt = QLineEdit()
44      self.lineedit_t1 = QLineEdit()
45      self.lineedit_s = QLineEdit()
46
47      self.msg_t0 = ErrorWidget("Wrong value!")
48      self.msg_t0.hide()
49
50      self.msg_dt = ErrorWidget("Wrong value!")
51      self.msg_dt.hide()
52
53      self.msg_t1 = ErrorWidget("Wrong value!")
54      self.msg_t1.hide()
55
56      self.msg_s = ErrorWidget("Wrong value!")
57      self.msg_s.hide()
58
59      header = QLabel("Accuracy settings", parent=self)
60      font = header.font()
61      font.setPointSize(15)
62      header.setFont(font)
63
64      self.prev_btn = QPushButton("Back", self)
65      self.prev_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowBack))
66
67      self.run_btn = QPushButton("Perform modeling", self)
68      self.run_btn.setIcon(QApplication.style().standardIcon(QStyle.SP_ArrowForward))
69      self.run_btn.setEnabled(False)
70
71      header_layout = QHBoxLayout()
```

```
72      header_layout.addWidget(header)
73      header_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
74
75      bottom_bar = QHBoxLayout()
76      bottom_bar.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Minimum))
77      bottom_bar.addWidget(self.prev_btn)
78      bottom_bar.addWidget(self.run_btn)
79
80      grid_layout = QGridLayout()
81
82      grid_layout.addWidget(self.msg_t0, 0, 2)
83
84      grid_layout.addWidget(info_t0, 1, 0)
85      grid_layout.addWidget(label_t0, 1, 1)
86      grid_layout.addWidget(self.lineedit_t0, 1, 2)
87
88      grid_layout.addWidget(self.msg_dt, 0, 5)
89
90      grid_layout.addWidget(info_dt, 1, 3)
91      grid_layout.addWidget(label_dt, 1, 4)
92      grid_layout.addWidget(self.lineedit_dt, 1, 5)
93
94      grid_layout.addWidget(self.msg_t1, 0, 8)
95
96      grid_layout.addWidget(info_t1, 1, 6)
97      grid_layout.addWidget(label_t1, 1, 7)
98      grid_layout.addWidget(self.lineedit_t1, 1, 8)
99
100     grid_layout.addWidget(self.msg_s, 2, 2)
101
102     grid_layout.addWidget(info_s, 3, 0)
103     grid_layout.addWidget(label_s, 3, 1)
104     grid_layout.addWidget(self.lineedit_s, 3, 2)
105
106     column_layout = QVBoxLayout()
107     column_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding)
        )
108     column_layout.addLayout(header_layout)
109     column_layout.addItem(QSpacerItem(0, 25, QSizePolicy.Minimum, QSizePolicy.Minimum))
110     column_layout.addLayout(grid_layout)
111     column_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding)
        )
112
113     control_layout = QHBoxLayout()
114     control_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding
        ))
115     control_layout.addLayout(column_layout)
116     control_layout.addItem(QSpacerItem(0, 0, QSizePolicy.Expanding, QSizePolicy.Expanding
        ))
117
118     layout = QVBoxLayout()
119     layout.addLayout(control_layout)
120     layout.addLayout(bottom_bar)
121
122     self.setLayout(layout)
```

```
123
124     self.lineedit_t0.textChanged.connect(self.validate_t0)
125     self.lineedit_t0.textChanged.connect(self.validate_t1)
126     self.lineedit_t0.textChanged.connect(self.validate_dt)
127     self.lineedit_dt.textChanged.connect(self.validate_dt)
128     self.lineedit_t1.textChanged.connect(self.validate_t0)
129     self.lineedit_t1.textChanged.connect(self.validate_t1)
130     self.lineedit_t1.textChanged.connect(self.validate_dt)
131     self.lineedit_s.textChanged.connect(self.validate_s)
132
133   def count_c(self, flag: bool):
134     if flag:
135       self.info_c.show()
136       self.label_c.show()
137     else:
138       self.info_c.hide()
139       self.label_c.hide()
140
141     self.msg_c.hide()
142
143   def validate_form(self):
144     if self.t0_is_valid \
145         and self.dt_is_valid \
146         and self.t1_is_valid \
147         and self.s_is_valid:
148       self.run_btn.setEnabled(True)
149     else:
150       self.run_btn.setEnabled(False)
151
152   def validate_t0(self):
153     try:
154       typed_value = float(self.lineedit_t0.text())
155       if typed_value >= self.t1:
156         raise ValueError()
157
158       self.t0_is_valid = True
159       self.t0 = typed_value
160       self.msg_t0.hide()
161
162     except ValueError:
163       self.t0_is_valid = False
164       self.msg_t0.show()
165
166   def validate_dt(self):
167     try:
168       typed_value = float(self.lineedit_dt.text())
169       if typed_value <= 0 \
170           or (self.t1 - self.t0) / typed_value < 1 \
171           or typed_value >= 1:
172         raise ValueError()
173
174       self.dt_is_valid = True
175       self.dt = typed_value
176       self.msg_dt.hide()
177       self.validate_form()
```

```
178
179       except ValueError:
180         self.dt_is_valid = False
181         self.msg_dt.show()
182
183     def validate_t1(self):
184       try:
185         typed_value = float(self.lineedit_t1.text())
186         if typed_value <= self.t0:
187           raise ValueError()
188
189         self.t1_is_valid = True
190         self.t1 = typed_value
191         self.msg_t1.hide()
192
193       except ValueError:
194         self.t1_is_valid = False
195         self.msg_t1.show()
196
197     def validate_s(self):
198       try:
199         if self.lineedit_s.text() == "":
200           self.s_is_valid = True
201           self.s = 0
202         else:
203           typed_value = int(self.lineedit_s.text())
204           if typed_value <= 0:
205             raise ValueError()
206           self.s = typed_value
207
208         self.s_is_valid = True
209         self.msg_s.hide()
210
211       except ValueError:
212         self.s_is_valid = False
213         self.msg_s.show()
214
215       finally:
216         self.validate_form()
```

6.2 Source Codes for Nonlinear Systems of Itô SDEs

6.2.1 Source Codes for Calculation of the Fourier–Legendre Coefficients

Listing 43: **Symbolic function of the Legendre polinomial**

```
1 from sympy import Rational, factorial, diff
2
```

```
3
4  def polynomial(n: int):
5    """
6    Returns the Legendre polynomial in symbolic format
7    Parameters
8    ==========
9    n : int
10     degree of the Legendre polynomial
11   Returns
12   =======
13   sympy.Expr
14   """
15   from sympy.abc import x
16   return Rational(1, 2) ** n / factorial(n) * diff((x ** 2 - 1) ** n, x, n)
```

Listing 44: **Symbolic function of the Fourier–Legendre coefficient calculation**

```
1  from sympy import S, integrate
2
3  from mathematics.sde.nonlinear.legendre_polynomial import polynomial
4
5
6  def get_c(indices: tuple, weights: tuple):
7    """
8    Calculates the Fourier-Legendre coefficient depending on indices and weights
9    Parameters
10   ==========
11   indices : tuple
12     indices of the Fourier-Legendre coefficient
13   weights : tuple
14     weights of the Fourier-Legendre coefficient
15   Returns
16   =======
17   sympy.Rational
18   """
19   from sympy.abc import x, y
20   # multiplicity of iterated integral which is the Fourier-Legendre coefficient
21   n = len(indices)
22   w = list(reversed(weights))
23   c = S.One
24
25   for i in reversed(range(1, n)):
26     c = integrate(polynomial(indices[i]) * (x + 1) ** w[i] * c, (x, -1, y)).subs(y, x)
27   c = integrate(polynomial(indices[0]) * (x + 1) ** w[0] * c, (x, -1, 1))
28
29   if sum(w) % 2 == 0:
30     return c
31   else:
32     return -c
```

Listing 45: **Symbolic function of the Fourier–Legendre coefficient**

```
1  import logging
```

```
2
3  from sympy import sympify, Function
4
5  import tools.database as db
6  from mathematics.sde.nonlinear.c import get_c
7
8
9  class C(Function):
10   """
11   Gives the Fourier-Legendre coefficient with requested indices and weights
12   """
13   _preloaded = dict()
14
15   def __new__(cls, indices: tuple, weights: tuple, to_float=True, **kwargs):
16     """
17     Creates the Fourier-Legendre coefficient object with needed indices and weights
18     Parameters
19     ==========
20     indices: tuple
21       requested indices
22     weights: tuple
23       requested weights
24     Returns
25     =======
26     symbolic.Rational or C
27       calculated value or symbolic expression
28     """
29     if not len(indices) == len(weights):
30       return super(C, cls).__new__(cls, indices, weights, **kwargs)
31
32     index = f"{':'.join([str(i) for i in indices])}_{':'.join([str(i) for i in weights])}
       "
33     try:
34       return cls._value(index, to_float)
35
36     except KeyError:
37       respond = cls._download_one(index)
38       if len(respond) != 0:
39         cls._preloaded[respond[0][0]] = respond[0][1]
40         return cls._value(index, to_float)
41       else:
42         new_c = cls._calculate(index, indices, weights)
43         cls._upload_one(new_c)
44         cls._preloaded[new_c[0]] = new_c[1]
45         return cls._value(index, to_float)
46
47   @classmethod
48   def _calculate(cls, index, indices, weights):
49     new_c = get_c(indices, weights)
50     return index, (new_c, sympify(new_c).evalf())
51
52   @classmethod
53   def _upload_one(cls, c):
54     logging.info(f"C: ADDING NEW C_{c[0]} = {c[1][0]}")
55     db.execute(f"INSERT INTO `C` (`index`, `value`, `value_f`) VALUES ('{c[0]}', '{c
```

```
    [1][0]}', {c[1][1]})")
56
57  @classmethod
58  def _unpack(cls, rows):
59    return [(rows[i][0], (rows[i][1], rows[i][2])) for i in range(len(rows))]
60
61  @classmethod
62  def _value(cls, index, to_float):
63    c = cls._preloaded[index]
64    if to_float:
65      return c[1]
66    else:
67      return sympify(c[0])
68
69  @classmethod
70  def _download_one(cls, index):
71    logging.info(f"C: MISSING PRELOADED VERSION OF C_{index}")
72    respond = db.execute(
73      f"SELECT `index`, `value`, `value_f` FROM `C`"
74      f"WHERE REGEXP(`index`, '^{index}$')"
75    )
76    return cls._unpack(respond)
77
78  @classmethod
79  def preload(cls, *args):
80    """
81    Updates dictionary of the preloaded Fourier-Legendre coefficients
82    Note: weights are not accepted, such coefficients are loaded
83    with all available weights
84    Parameters
85    ==========
86    args
87      Indices for the Fourier-Legendre coefficients
88      to download them from database
89    """
90    logging.info(f"C: PRELOADING COEFFICIENTS {args}")
91
92    query = []
93    for q in range(len(args)):
94      numbers = [int(char) for char in str(args[q] + 1)]
95      pattern = []
96
97      for i in range(1, len(numbers)):
98        pattern.append("[0-9]" * i)
99
100     for i in range(len(numbers)):
101       p = []
102       for j in range(len(numbers)):
103         if j < i:
104           p.append(str(numbers[j]))
105         elif i == j:
106           p.append(f"[0-{numbers[j] - 1}]")
107         elif j > i:
108           p.append("[0-9]")
109       pattern.append("".join(p))
```

```
110
111       regex = f"^{':'.join(['|'.join(pattern) for _ in range(q + 2)])}_.*$"
112       query.append(
113         f"SELECT `index`, `value`, `value_f` FROM `C`"
114         f"WHERE REGEXP(`index`, '{regex}')"
115       )
116
117     result = db.execute("\nUNION\n".join(query))
118     cls._preloaded.update(cls._unpack(result))
119
120   def doit(self, **hints):
121     """
122     Tries to expand or calculate function
123     Returns
124     =======
125     C
126     """
127     return C(*self.args, **hints)
```

Listing 46: **Calculation of the Fourier–Legendre coefficients** $C^{000}_{j_3 j_2 j_1}$

```
1  from math import sqrt
2
3  from sympy import sympify, Number, Function
4
5  from mathematics.sde.nonlinear.symbolic.coefficients.c import C
6
7
8  class C000(Function):
9    """
10   Gives the Fourier-Legendre coefficient with requested indices and weights
11   """
12   nargs = 4
13
14   def __new__(cls, *args, **kwargs):
15     """
16     Creates the Fourier-Legendre coefficient object with needed
17     indices and weights and calculates it
18     Parameters
19     ==========
20     indices: tuple
21       requested indices
22     weights: tuple
23       requested weights
24     Returns
25     =======
26     symbolic.Rational or C000
27       calculated value or symbolic expression
28     """
29     j3, j2, j1, dt = sympify(args)
30
31     if not (isinstance(j1, Number) and
32         isinstance(j2, Number) and
```

```
        isinstance(j3, Number) and
        isinstance(dt, Number)):
      return super(C000, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1)) * \
        dt ** 1.5 * \
        C((j3, j2, j1), (0, 0, 0)) / 8

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C000
    """
    return C000(*self.args, **hints)
```

Listing 47: **Calculation of the Fourier–Legendre coefficients** $C^{10}_{j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C10(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C10
      calculated value or symbolic expression
    """
    j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
```

```
        isinstance(j2, Number) and
        isinstance(dt, Number)):
      return super(C10, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1)) * \
        dt ** 2 * \
        C((j2, j1), (1, 0)) / 8

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C10
    """
    return C10(*self.args, **hints)
```

Listing 48: **Calculation of the Fourier–Legendre coefficients** $C^{01}_{j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C01(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C01
      calculated value or symbolic expression
    """
    j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
```

```
        isinstance(dt, Number)):
      return super(C01, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1)) * \
        dt ** 2 * \
        C((j2, j1), (0, 1)) / 8

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C01
    """
    return C01(*self.args, **hints)
```

Listing 49: **Calculation of the Fourier–Legendre coefficients** $C^{0000}_{j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Number, Function

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C0000(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C0000
      calculated value or symbolic expression
    """
    j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
```

```
        isinstance(j4, Number) and
        isinstance(dt, Number)):
      return super(C0000, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1)) * \
        dt ** 2 * \
       C((j4, j3, j2, j1), (0, 0, 0, 0)) / 16

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C0000
    """
    return C0000(*self.args, **hints)
```

Listing 50: **Calculation of the Fourier–Legendre coefficients** $C^{100}_{j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C100(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 4

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C100
      calculated value or symbolic expression
    """
    j3, j2, j1, dt = sympify(args)

```

```
    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(dt, Number)):
      return super(C100, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1)) * \
        dt ** 2.5 * \
       C((j3, j2, j1), (1, 0, 0)) / 16

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C100
    """
    return C100(*self.args, **hints)
```

Listing 51: **Calculation of the Fourier–Legendre coefficients** $C^{010}_{j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C010(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 4

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C010
      calculated value or symbolic expression
    """
    j3, j2, j1, dt = sympify(args)
```

```
30
31      if not (isinstance(j1, Number) and
32          isinstance(j2, Number) and
33          isinstance(j3, Number) and
34          isinstance(dt, Number)):
35        return super(C010, cls).__new__(cls, *args, **kwargs)
36
37      return sqrt(
38        (j1 * 2 + 1) *
39        (j2 * 2 + 1) *
40        (j3 * 2 + 1)) * \
41          dt ** 2.5 * \
42          C((j3, j2, j1), (0, 1, 0)) / 16
43
44    def doit(self, **hints):
45      """
46      Tries to expand or calculate function
47      Returns
48      =======
49      C010
50      """
51      return C010(*self.args, **hints)
```

Listing 52: **Calculation of the Fourier–Legendre coefficients** $C^{001}_{j_3 j_2 j_1}$

```
1  from math import sqrt
2
3  from sympy import sympify, Function, Number
4
5  from mathematics.sde.nonlinear.symbolic.coefficients.c import C
6
7
8  class C001(Function):
9    """
10   Gives the Fourier-Legendre coefficient with requested indices and weights
11   """
12   nargs = 4
13
14   def __new__(cls, *args, **kwargs):
15     """
16     Creates the Fourier-Legendre coefficient object with needed
17     indices and weights and calculates it
18     Parameters
19     ==========
20     indices: tuple
21       requested indices
22     weights: tuple
23       requested weights
24     Returns
25     =======
26     symbolic.Rational or C001
27       calculated value or symbolic expression
28     """
```

```
    j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(dt, Number)):
      return super(C001, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1)) * \
        dt ** 2.5 * \
        C((j3, j2, j1), (0, 0, 1)) / 16

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C001
    """
    return C001(*self.args, **hints)
```

Listing 53: **Calculation of the Fourier–Legendre coefficients** $C^{00000}_{j_5 j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Number, Function

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C00000(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C00000
      calculated value or symbolic expression
```

```
    """
    j5, j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(j4, Number) and
        isinstance(j5, Number) and
        isinstance(dt, Number)):
      return super(C00000, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1) *
      (j5 * 2 + 1)) * \
        dt ** 2.5 * \
        C((j5, j4, j3, j2, j1), (0, 0, 0, 0, 0)) / 32

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C00000
    """
    return C00000(*self.args, **hints)
```

Listing 54: **Calculation of the Fourier–Legendre coefficients** $C^{20}_{j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C20(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
```

```
      requested weights
    Returns
    =======
    symbolic.Rational or C20
      calculated value or symbolic expression
    """
    j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(dt, Number)):
      return super(C20, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1)) * \
        dt ** 3 * \
        C((j2, j1), (2, 0)) / 16

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C20
    """
    return C20(*self.args, **hints)
```

Listing 55: **Calculation of the Fourier–Legendre coefficients** $C^{11}_{j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C11(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
```

```
    Returns
    =======
    symbolic.Rational or C11
      calculated value or symbolic expression
    """
    j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(dt, Number)):
      return super(C11, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1)) * \
        dt ** 3 * \
        C((j2, j1), (1, 1)) / 16

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C11
    """
    return C11(*self.args, **hints)
```

Listing 56: **Calculation of the Fourier–Legendre coefficients** $C^{02}_{j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C02(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
```

```
    =======
    symbolic.Rational or C02
      calculated value or symbolic expression
    """
    j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(dt, Number)):
      return super(C02, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1)) * \
        dt ** 3 * \
        C((j2, j1), (0, 2)) / 16

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C02
    """
    return C02(*self.args, **hints)
```

Listing 57: **Calculation of the Fourier–Legendre coefficients** $C^{1000}_{j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Number, Function

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C1000(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
```

```
    symbolic.Rational or C1000
      calculated value or symbolic expression
    """
    j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(j4, Number) and
        isinstance(dt, Number)):
      return super(C1000, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1)) * \
        dt ** 3 * \
        C((j4, j3, j2, j1), (1, 0, 0, 0)) / 32

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C1000
    """
    return C1000(*self.args, **hints)
```

Listing 58: **Calculation of the Fourier–Legendre coefficients** $C^{0100}_{j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C0100(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
```

```
      requested weights
    Returns
    =======
    symbolic.Rational or C0100
      calculated value or symbolic expression
    """
    j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(j4, Number) and
        isinstance(dt, Number)):
      return super(C0100, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1)) * \
        dt ** 3 * \
        C((j4, j3, j2, j1), (0, 1, 0, 0)) / 32

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C0100
    """
    return C0100(*self.args, **hints)
```

Listing 59: **Calculation of the Fourier–Legendre coefficients** $C^{0010}_{j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C0010(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
```

```
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C0010
      calculated value or symbolic expression
    """
    j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(j4, Number) and
        isinstance(dt, Number)):
      return super(C0010, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1)) * \
        dt ** 3 * \
        C((j4, j3, j2, j1), (0, 0, 1, 0)) / 32

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C0010
    """
    return C0010(*self.args, **hints)
```

Listing 60: **Calculation of the Fourier–Legendre coefficients** $C^{0001}_{j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Function, Number

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C0001(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier-Legendre coefficient object with needed
```

```
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C0001
      calculated value or symbolic expression
    """
    j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(j4, Number) and
        isinstance(dt, Number)):
      return super(C0001, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1)) * \
        dt ** 3 * \
        C((j4, j3, j2, j1), (0, 0, 0, 1)) / 32

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C0001
    """
    return C0001(*self.args, **hints)
```

Listing 61: **Calculation of the Fourier–Legendre coefficients** $C^{000000}_{j_6 j_5 j_4 j_3 j_2 j_1}$

```
from math import sqrt

from sympy import sympify, Number, Function

from mathematics.sde.nonlinear.symbolic.coefficients.c import C


class C000000(Function):
  """
  Gives the Fourier-Legendre coefficient with requested indices and weights
  """
  nargs = 7

```

```
  def __new__(cls, *args, **kwargs):
    """
    Creates the Fourier–Legendre coefficient object with needed
    indices and weights and calculates it
    Parameters
    ==========
    indices: tuple
      requested indices
    weights: tuple
      requested weights
    Returns
    =======
    symbolic.Rational or C000000
      calculated value or symbolic expression
    """
    j6, j5, j4, j3, j2, j1, dt = sympify(args)

    if not (isinstance(j1, Number) and
        isinstance(j2, Number) and
        isinstance(j3, Number) and
        isinstance(j4, Number) and
        isinstance(j5, Number) and
        isinstance(dt, Number)):
      return super(C000000, cls).__new__(cls, *args, **kwargs)

    return sqrt(
      (j1 * 2 + 1) *
      (j2 * 2 + 1) *
      (j3 * 2 + 1) *
      (j4 * 2 + 1) *
      (j5 * 2 + 1) *
      (j6 * 2 + 1)) * \
        dt ** 3 * \
        C((j6, j5, j4, j3, j2, j1), (0, 0, 0, 0, 0, 0)) / 64

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    C000000
    """
    return C000000(*self.args, **hints)
```

Listing 62: **Program entry for generation of new Fourier–Legendre coefficients**

```
#!/usr/bin/env python
import logging
import os
from datetime import datetime
from multiprocessing import cpu_count, Pool
from pprint import pprint

```

```
8  from config import csv, new_c_portion_size
9  from mathematics.sde.nonlinear.new_c import thread_c, split_task
10
11
12 def main():
13   logging.basicConfig(
14     level=logging.INFO,
15     format="%(asctime)s - %(levelname)s - %(message)s",
16     datefmt="%H:%M:%S"
17   )
18   logger = logging.getLogger(__name__)
19
20   filename = os.path.join(csv, f"c_{datetime.now().strftime('%d-%m-%Y_%H-%M-%S')}.csv")
21   logging.info(f"Writing to file {filename}")
22
23   tasks = [
24     (((57, 58), (57, 58), (57, 58)), (0, 0, 0)),
25     # (((0, 56), (0, 56), (0, 56)), (0, 0, 0)),
26     #
27     # (((0, 15), (0, 15)), (0, 1)),
28     # (((0, 15), (0, 15)), (1, 0)),
29     # (((0, 15), (0, 15), (0, 15)), (0, 0, 0, 0)),
30     #
31     # (((0, 6), (0, 6), (0, 6)), (0, 0, 1)),
32     # (((0, 6), (0, 6), (0, 6)), (0, 1, 0)),
33     # (((0, 6), (0, 6), (0, 6)), (1, 0, 0)),
34     # (((0, 6), (0, 6), (0, 6), (0, 6)), (0, 0, 0, 0, 0)),
35     #
36     # (((0, 2), (0, 2)), (0, 2)),
37     # (((0, 2), (0, 2)), (1, 1)),
38     # (((0, 2), (0, 2)), (2, 0)),
39     # (((0, 2), (0, 2), (0, 2), (0, 2)), (0, 0, 0, 1)),
40     # (((0, 2), (0, 2), (0, 2), (0, 2)), (0, 0, 1, 0)),
41     # (((0, 2), (0, 2), (0, 2), (0, 2)), (0, 1, 0, 0)),
42     # (((0, 2), (0, 2), (0, 2), (0, 2)), (1, 0, 0, 0)),
43     # (((0, 2), (0, 2), (0, 2), (0, 2), (0, 2)), (0, 0, 0, 0, 0)),
44     # (((0, 2), (0, 2), (0, 2), (0, 2), (0, 2), (0, 2)), (0, 0, 0, 0, 0, 0)),
45   ]
46
47   with Pool(cpu_count()) as p:
48
49     for t in tasks:
50
51       logger.info(f"Running task {t}")
52
53       for chunk in chunks(split_task(t), new_c_portion_size):
54         c = p.map(thread_c, chunk)
55         c.append("")
56
57         with open(filename, "a") as f:
58           f.write("\n".join(c))
59           f.close()
60
61         logger.info(f"The portion of C has been written {chunk[0]}-{chunk[-1]}")
62
```

```
63    logger.info("Generation has been done")
64
65
66  def chunks(lst, n):
67    for i in range(0, len(lst), n):
68      yield lst[i:i + n]
69
70
71  if __name__ == "__main__":
72    main()
```

Listing 63: **Module for the Fourier–Legendre coefficients generation**

```
1   from mathematics.sde.nonlinear.c import get_c
2
3
4   def split_task(tasks):
5     c = len(tasks[1])
6
7     if c == 2:
8       return split_2(tasks)
9
10    if c == 3:
11      return split_3(tasks)
12
13    if c == 4:
14      return split_4(tasks)
15
16    if c == 5:
17      return split_5(tasks)
18
19    if c == 6:
20      return split_6(tasks)
21
22
23  def split_2(ranges):
24    return [((i, j), ranges[1])
25        for j in range(ranges[0][1][1])
26        for i in range(ranges[0][0][1])
27        if i >= ranges[0][1][0]
28        or j >= ranges[0][0][1]]
29
30
31  def split_3(ranges):
32    return [((i, j, k), ranges[1])
33        for k in range(ranges[0][2][1])
34        for j in range(ranges[0][1][1])
35        for i in range(ranges[0][0][1])
36        if k >= ranges[0][2][0]
37        or j >= ranges[0][1][0]
38        or i >= ranges[0][0][0]]
39
40
```

```
41 def split_4(ranges):
42   return [((i, j, k, l), ranges[1])
43       for j in range(*ranges[0][3])
44       for i in range(*ranges[0][2])
45       for k in range(*ranges[0][1])
46       for l in range(*ranges[0][0])
47       if l >= ranges[0][3][0]
48       or k >= ranges[0][2][0]
49       or j >= ranges[0][1][0]
50       or i >= ranges[0][0][0]]
51
52
53 def split_5(ranges):
54   return [((i, j, k, l, m), ranges[1])
55       for j in range(*ranges[0][4])
56       for i in range(*ranges[0][3])
57       for k in range(*ranges[0][2])
58       for l in range(*ranges[0][1])
59       for m in range(*ranges[0][0])
60       if m >= ranges[0][4][0]
61       if l >= ranges[0][3][0]
62       or k >= ranges[0][2][0]
63       or j >= ranges[0][1][0]
64       or i >= ranges[0][0][0]]
65
66
67 def split_6(ranges):
68   return [((i, j, k, l, m, n), ranges[1])
69       for j in range(*ranges[0][5])
70       for i in range(*ranges[0][4])
71       for k in range(*ranges[0][3])
72       for l in range(*ranges[0][2])
73       for m in range(*ranges[0][1])
74       for n in range(*ranges[0][0])
75       if n >= ranges[0][5][0]
76       if m >= ranges[0][4][0]
77       if l >= ranges[0][3][0]
78       or k >= ranges[0][2][0]
79       or j >= ranges[0][1][0]
80       or i >= ranges[0][0][0]]
81
82
83 def thread_c(ranges):
84   c = len(ranges[1])
85
86   if c == 2:
87     return gen_2(*ranges[0], ranges[1])
88
89   if c == 3:
90     return gen_3(*ranges[0], ranges[1])
91
92   if c == 4:
93     return gen_4(*ranges[0], ranges[1])
94
95   if c == 5:
```

```
96      return gen_5(*ranges[0], ranges[1])
97
98    if c == 6:
99      return gen_6(*ranges[0], ranges[1])
100
101
102 def gen_2(i, j, w):
103   return f"\"{i}:{j}_{w[0]}:{w[1]}\";\"{get_c((i, j), w)}\""
104
105
106 def gen_3(i, j, k, w):
107   return f"\"{i}:{j}:{k}_{w[0]}:{w[1]}:{w[2]}\";\"{get_c((i, j, k), w)}\""
108
109
110 def gen_4(i, j, k, l, w):
111   return f"\"{i}:{j}:{k}:{l}_{w[0]}:{w[1]}:{w[2]}:{w[3]}\";\"{get_c((i, j, k, l), w)}\""
112
113
114 def gen_5(i, j, k, l, m, w):
115   return f"\"{i}:{j}:{k}:{l}:{m}_{w[0]}:{w[1]}:{w[2]}:{w[3]}:{w[4]}\";\"{get_c((i, j, k,
        l, m), w)}\""
116
117
118 def gen_6(i, j, k, l, m, n, w):
119   return f"\"{i}:{j}:{k}:{l}:{m}_{w[0]}:{w[1]}:{w[2]}:{w[3]}:{w[4]}:{w[5]}\";\"{get_c((i,
        j, k, l, m, n), w)}\""
```

6.2.2 Source Codes for Supplementary Differential Operators and Functions

Listing 64: **Implementation of the differential operator** L

```
1  from sympy import Add, sympify, Number, diff
2
3  from mathematics.sde.nonlinear.symbolic.operator import Operator
4
5
6  class L(Operator):
7    nargs = 4
8
9    def __new__(cls, *args, **kwargs):
10     """
11     Creates new L object with given args
12     Parameters
13     ==========
14     args
15       bunch of necessary arguments
16     Returns
17     =======
18     sympy.Expr
```

```
      formula to simplify and substitute
    """
    a, b, f, dxs = sympify(args)

    if not ((isinstance(f, Number) or f.has(*dxs)) and
        not f.has(Operator) and a.is_Matrix):
      return super(L, cls).__new__(cls, *args, **kwargs)

    n = b.shape[0]
    m = b.shape[1]
    from sympy.abc import t

    return Add(
      diff(f, t),
      *[a[i, 0] * diff(f, dxs[i]) for i in range(n)],
      *[0.5 * b[i, j] * b[k, j] * diff(f, dxs[i], dxs[k])
       for j in range(m)
       for i in range(n)
       for k in range(n)]
    )

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    L
    """
    return L(*self.args, **hints)
```

Listing 65: **Implementation of the differential operator $G_0^{(i)}$**

```
from sympy import Add, sympify, Number, diff

from mathematics.sde.nonlinear.symbolic.operator import Operator


class G(Operator):
  nargs = 3

  _dict = dict()

  def __new__(cls, *args, **kwargs):
    """
    Creates new G object with given args
    Parameters
    ==========
    args
      bunch of necessary arguments
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
```

```
    """
    c, f, dxs = sympify(args)

    if not ((isinstance(f, Number) or f.has(*dxs)) and
        not f.has(Operator)):
      return super(G, cls).__new__(cls, *args, **kwargs)

    return Add(*[c[i, 0] * diff(f, dxs[i])
           for i in range(len(dxs))])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    G
    """
    return G(*self.args, **hints)
```

Listing 66: **Implementation of the function $\bar{\mathbf{a}}(\mathbf{x}, t)$**

```
from sympy import sympify, Matrix, Add

from mathematics.sde.nonlinear.symbolic.g import G
from mathematics.sde.nonlinear.symbolic.operator import Operator


class Aj(Operator):
  nargs = 4

  def __new__(cls, *args, **kwargs):
    """
    Creates new Aj object with given args
    Parameters
    ==========
    args
      bunch of necessary arguments
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i, a, b, dxs = sympify(args)
    n = b.shape[0]
    m = b.shape[1]

    return Matrix([a[i, 0] -
             (Add(*[0.5 * G(b[:, j], b[i, j], dxs)
                for j in range(m)]))
             for i in range(n)])
```

Listing 67: **Implementation of the differential operator $\bar{L}$**

```
from sympy import sympify, Number, diff, Add
```

```
2
3  from mathematics.sde.nonlinear.symbolic.operator import Operator
4
5
6  class Lj(Operator):
7    nargs = 3
8
9    def __new__(cls, *args, **kwargs):
10     """
11     Creates new Lj object with given args
12     Parameters
13     ==========
14     args
15       bunch of necessary arguments
16     Returns
17     =======
18     sympy.Expr
19       formula to simplify and substitute
20     """
21     a, f, dxs = sympify(args)
22
23     if not (isinstance(f, Number) or (f.has(*dxs)) and
24         not f.has(Operator)):
25       return super(Lj, cls).__new__(cls, *args, **kwargs)
26
27     n = a.shape[0]
28     from sympy.abc import t
29     return Add(diff(f, t), *[a[i, 0] * diff(f, dxs[i])
30                   for i in range(n)])
31
32   def doit(self, **hints):
33     """
34     Tries to expand or calculate function
35     Returns
36     =======
37     Lj
38     """
39     return Lj(*self.args, **hints)
```

Listing 68: **Implementation of the indicator function $\mathbf{1}_{\{i_1=i_2\}}$**

```
1  from sympy import sympify, Number, Function
2
3
4  class Ind(Function):
5    """
6    Indicator function
7    """
8    nargs = 2
9
10   def __new__(cls, *args, **kwargs):
11     """
12     Creates new Ind object with given args
```

```
    Parameters
    ==========
    args
      bunch of necessary arguments
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2 = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number)):
      return super(Ind, cls).__new__(cls, *args, **kwargs)

    if i1 == i2:
      return 1
    else:
      return 0

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    Ind
    """
    return Ind(*self.args, **hints)
```

6.2.3 Source Codes for Iterated Itô Stochastic Integrals Approximations Subprograms

Listing 69: **Approximation of Itô stochastic integral** $I_{(0)\tau_{p+1},\tau_p}^{(i_1)}$

```
from math import sqrt

from sympy import Function, sympify, Number


class I0(Function):
  """
  Ito stochastic integral
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates new I0 object with given args
    Parameters
    ==========
```

```
    i1 : int
      integral index
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, dt, ksi = sympify(args)

    if not isinstance(i1, Number):
      return super(I0, cls).__new__(cls, *args, **kwargs)

    return ksi[0, i1] * sqrt(dt)

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I0
    """
    return I0(*self.args, **hints)
```

Listing 70: **Approximation of iterated Itô stochastic integral** $I_{(00)\tau_{p+1},\tau_p}^{(i_1 i_2)}$

```
from math import sqrt

from sympy import sympify, Number, Function, Add

from mathematics.sde.nonlinear.symbolic.ind import Ind


class I00(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates new I00 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    q : int
      amount of terms in approximation of
```

```
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(q, Number)):
      return super(I00, cls).__new__(cls, *args, **kwargs)

    return \
      (ksi[0, i1] * ksi[0, i2] +
       Add(*[
         (ksi[j1 - 1, i1] * ksi[j1, i2] -
         ksi[j1, i1] * ksi[j1 - 1, i2]) /
         sqrt(j1 ** 2 * 4 - 1)
         for j1 in range(1, q + 1)]) -
       Ind(i1, i2)) * dt / 2

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I00
    """
    return I00(*self.args, **hints)
```

Listing 71: **Approximation of Itô stochastic integral** $I^{(i_1)}_{(1)\tau_{p+1},\tau_p}$

```
from math import sqrt

from sympy import Function, sympify, Number


class I1(Function):
  """
  Ito stochastic integral
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates new I1 object with given args
    Parameters
    ==========
```

```
    i1 : int
      integral index
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, dt, ksi = sympify(args)

    if not isinstance(i1, Number):
      return super(I1, cls).__new__(cls, *args, **kwargs)

    return -(ksi[0, i1] + ksi[1, i1] / sqrt(3)) * dt ** 1.5 / 2

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I1
    """
    return I1(*self.args, **hints)
```

Listing 72: **Approximation of iterated Itô stochastic integral** $I_{(000)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c000 import C000
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I000(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates new I000 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    q : int
```

```
25        amount of terms in approximation of
26        iterated Ito stochastic integral
27      dt : float
28        delta time
29      ksi : numpy.ndarray
30        matrix of Gaussian random variables
31      Returns
32      =======
33      sympy.Expr
34        formula to simplify and substitute
35      """
36      i1, i2, i3, q, dt, ksi = sympify(args)
37
38      if not (isinstance(i1, Number) and
39          isinstance(i2, Number) and
40          isinstance(i3, Number) and
41          isinstance(q, Number)):
42        return super(I000, cls).__new__(cls, *args, **kwargs)
43
44      return Add(*[
45        C000(j3, j2, j1, dt) *
46        (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] -
47         Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] -
48         Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] -
49         Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1])
50        for j3 in range(q + 1)
51        for j2 in range(q + 1)
52        for j1 in range(q + 1)])
53
54    def doit(self, **hints):
55      """
56      Tries to expand or calculate function
57      Returns
58      =======
59      I000
60      """
61      return I000(*self.args, **hints)
```

Listing 73: **Approximation of iterated Itô stochastic integral** $I_{(10)\tau_{p+1},\tau_p}^{(i_1 i_2)}$

```
1  from sympy import Function, sympify, Number, Add
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c10 import C10
4  from mathematics.sde.nonlinear.symbolic.ind import Ind
5
6
7  class I10(Function):
8    """
9    Iterated Ito stochastic integral
10   """
11   nargs = 5
12
13   def __new__(cls, *args, **kwargs):
```

```
    """
    Creates new I10 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(q, Number)):
      return super(I10, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C10(j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] -
       Ind(i1, i2) * Ind(j1, j2))
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I10
    """
    return I10(*self.args, **hints)
```

Listing 74: **Approximation of iterated Itô stochastic integral** $I_{(01)\tau_{p+1},\tau_p}^{(i_1 i_2)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c01 import C01
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I01(Function):
  """
```

```
    Iterated Ito stochastic integral
    """
    nargs = 5

    def __new__(cls, *args, **kwargs):
        """
        Creates new I01 object with given args
        Parameters
        ==========
        i1 : int
            integral index
        i2 : int
            integral index
        q : int
            amount of terms in approximation of
            iterated Ito stochastic integral
        dt : float
            delta time
        ksi : numpy.ndarray
            matrix of Gaussian random variables
        Returns
        =======
        sympy.Expr
            formula to simplify and substitute
        """
        i1, i2, q, dt, ksi = sympify(args)

        if not (isinstance(i1, Number) and
                isinstance(i2, Number) and
                isinstance(q, Number)):
            return super(I01, cls).__new__(cls, *args, **kwargs)

        return Add(*[
            C01(j2, j1, dt) *
            (ksi[j1, i1] * ksi[j2, i2] -
             Ind(i1, i2) * Ind(j1, j2))
            for j2 in range(q + 1)
            for j1 in range(q + 1)])

    def doit(self, **hints):
        """
        Tries to expand or calculate function
        Returns
        =======
        I01
        """
        return I01(*self.args, **hints)
```

Listing 75: **Approximation of iterated Itô stochastic integral** $I_{(0000)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c0000 import C0000
```

```
4  from mathematics.sde.nonlinear.symbolic.ind import Ind
5
6
7  class I0000(Function):
8    """
9    Iterated Ito stochastic integral
10   """
11   nargs = 7
12
13   def __new__(cls, *args, **kwargs):
14     """
15     Creates new I0000 object with given args
16     Parameters
17     ==========
18     i1 : int
19       integral index
20     i2 : int
21       integral index
22     i3 : int
23       integral index
24     i4 : int
25       integral index
26     q : int
27       amount of terms in approximation of
28       iterated Ito stochastic integral
29     dt : float
30       delta time
31     ksi : numpy.ndarray
32       matrix of Gaussian random variables
33     Returns
34     =======
35     sympy.Expr
36       formula to simplify and substitute
37     """
38     i1, i2, i3, i4, q, dt, ksi = sympify(args)
39
40     if not (isinstance(i1, Number) and
41         isinstance(i2, Number) and
42         isinstance(i3, Number) and
43         isinstance(i4, Number) and
44         isinstance(q, Number)):
45       return super(I0000, cls).__new__(cls, *args, **kwargs)
46
47     return Add(*[
48       C0000(j4, j3, j2, j1, dt) *
49       (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] -
50        Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] * ksi[j4, i4] -
51        Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] * ksi[j4, i4] -
52        Ind(i1, i4) * Ind(j1, j4) * ksi[j2, i2] * ksi[j3, i3] -
53        Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1] * ksi[j4, i4] -
54        Ind(i2, i4) * Ind(j2, j4) * ksi[j1, i1] * ksi[j3, i3] -
55        Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1] * ksi[j2, i2] +
56        Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i4) * Ind(j3, j4) +
57        Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i4) * Ind(j2, j4) +
58        Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i3) * Ind(j2, j3))
```

```
      for j4 in range(q + 1)
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I0000
    """
    return I0000(*self.args, **hints)
```

Listing 76: **Approximation of Itô stochastic integral** $I^{(i_1)}_{(2)\tau_{p+1},\tau_p}$

```
from math import sqrt

from sympy import sympify, Number, Function


class I2(Function):
  """
  Ito stochastic integral
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates new I2 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, dt, ksi = sympify(args)

    if not isinstance(i1, Number):
      return super(I2, cls).__new__(cls, *args, **kwargs)

    return (ksi[0, i1] + ksi[1, i1] * sqrt(3) / 2 +
        ksi[2, i1] / sqrt(5) / 2) * dt ** 2.5 / 3

  def doit(self, **hints):
    """
```

```
    Tries to expand or calculate function
    Returns
    =======
    I2
    """
    return I2(*self.args, **hints)
```

Listing 77: **Approximation of iterated Itô stochastic integral** $I_{(100)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c100 import C100
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I100(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates new I100 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(q, Number)):
      return super(I100, cls).__new__(cls, *args, **kwargs)

    return Add(*[
```

```
      C100(j3, j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] -
       Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] -
       Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] -
       Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1])
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I100
    """
    return I100(*self.args, **hints)
```

Listing 78: **Approximation of iterated Itô stochastic integral** $I_{(010)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c010 import C010
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I010(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates new I010 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
```

```
34        formula to simplify and substitute
35      """
36      i1, i2, i3, q, dt, ksi = sympify(args)
37
38      if not (isinstance(i1, Number) and
39          isinstance(i2, Number) and
40          isinstance(i3, Number) and
41          isinstance(q, Number)):
42        return super(I010, cls).__new__(cls, *args, **kwargs)
43
44      return Add(*[
45        C010(j3, j2, j1, dt) *
46        (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] -
47         Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] -
48         Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] -
49         Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1])
50        for j3 in range(q + 1)
51        for j2 in range(q + 1)
52        for j1 in range(q + 1)])
53
54    def doit(self, **hints):
55      """
56      Tries to expand or calculate function
57      Returns
58      =======
59      I010
60      """
61      return I010(*self.args, **hints)
```

Listing 79: **Approximation of iterated Itô stochastic integral** $I_{(001)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3)}$

```
1  from sympy import Function, sympify, Number, Add
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c001 import C001
4  from mathematics.sde.nonlinear.symbolic.ind import Ind
5
6
7  class I001(Function):
8    """
9    Iterated Ito stochastic integral
10   """
11   nargs = 6
12
13   def __new__(cls, *args, **kwargs):
14     """
15     Creates new I001 object with given args
16     Parameters
17     ==========
18     i1 : int
19       integral index
20     i2 : int
21       integral index
22     i3 : int
```

```
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(q, Number)):
      return super(I001, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C001(j3, j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] -
       Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] -
       Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] -
       Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1])
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I001
    """
    return I001(*self.args, **hints)
```

Listing 80: **Approximation of iterated Itô stochastic integral** $I_{(00000)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4 i_5)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c00000 import C00000
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I00000(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 8
```

```
12
13    def __new__(cls, *args, **kwargs):
14        """
15        Creates new I00000 object with given args
16        Parameters
17        ==========
18        i1 : int
19            integral index
20        i2 : int
21            integral index
22        i3 : int
23            integral index
24        i4 : int
25            integral index
26        i5 : int
27            integral index
28        q : int
29            amount of terms in approximation of
30            iterated Ito stochastic integral
31        dt : float
32            delta time
33        ksi : numpy.ndarray
34            matrix of Gaussian random variables
35        Returns
36        =======
37        sympy.Expr
38            formula to simplify and substitute
39        """
40        i1, i2, i3, i4, i5, q, dt, ksi = sympify(args)
41
42        if not (isinstance(i1, Number) and
43                isinstance(i2, Number) and
44                isinstance(i3, Number) and
45                isinstance(i4, Number) and
46                isinstance(i5, Number) and
47                isinstance(q, Number) and
48                isinstance(dt, Number)):
49            return super(I00000, cls).__new__(cls, *args, **kwargs)
50
51        return Add(*[
52            C00000(j5, j4, j3, j2, j1, dt) *
53            (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] * ksi[j5, i5] -
54             Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] * ksi[j4, i4] * ksi[j5, i5] -
55             Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] * ksi[j4, i4] * ksi[j5, i5] -
56             Ind(i1, i4) * Ind(j1, j4) * ksi[j2, i2] * ksi[j3, i3] * ksi[j5, i5] -
57             Ind(i1, i5) * Ind(j1, j5) * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] -
58             Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1] * ksi[j4, i4] * ksi[j5, i5] -
59             Ind(i2, i4) * Ind(j2, j4) * ksi[j1, i1] * ksi[j3, i3] * ksi[j5, i5] -
60             Ind(i2, i5) * Ind(j2, j5) * ksi[j1, i1] * ksi[j3, i3] * ksi[j4, i4] -
61             Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1] * ksi[j2, i2] * ksi[j5, i5] -
62             Ind(i3, i5) * Ind(j3, j5) * ksi[j1, i1] * ksi[j2, i2] * ksi[j4, i4] -
63             Ind(i4, i5) * Ind(j4, j5) * ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] +
64             Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i4) * Ind(j3, j4) * ksi[j5, i5] +
65             Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i5) * Ind(j3, j5) * ksi[j4, i4] +
66             Ind(i1, i2) * Ind(j1, j2) * Ind(i4, i5) * Ind(j4, j5) * ksi[j3, i3] +
```

```
        Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i4) * Ind(j2, j4) * ksi[j5, i5] +
        Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i5) * Ind(j2, j5) * ksi[j4, i4] +
        Ind(i1, i3) * Ind(j1, j3) * Ind(i4, i5) * Ind(j4, j5) * ksi[j2, i2] +
        Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i3) * Ind(j2, j3) * ksi[j5, i5] +
        Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i5) * Ind(j2, j5) * ksi[j3, i3] +
        Ind(i1, i4) * Ind(j1, j4) * Ind(i3, i5) * Ind(j3, j5) * ksi[j2, i2] +
        Ind(i1, i5) * Ind(j1, j5) * Ind(i2, i3) * Ind(j2, j3) * ksi[j4, i4] +
        Ind(i1, i5) * Ind(j1, j5) * Ind(i2, i4) * Ind(j2, j4) * ksi[j3, i3] +
        Ind(i1, i5) * Ind(j1, j5) * Ind(i3, i4) * Ind(j3, j4) * ksi[j2, i2] +
        Ind(i2, i3) * Ind(j2, j3) * Ind(i4, i5) * Ind(j4, j5) * ksi[j1, i1] +
        Ind(i2, i4) * Ind(j2, j4) * Ind(i3, i5) * Ind(j3, j5) * ksi[j1, i1] +
        Ind(i2, i5) * Ind(j2, j5) * Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1])
      for j5 in range(q + 1)
      for j4 in range(q + 1)
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I00000
    """
    return I00000(*self.args, **hints)
```

Listing 81: **Approximation of iterated Itô stochastic integral** $I_{(20)\tau_{p+1},\tau_p}^{(i_1 i_2)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c20 import C20
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I20(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates new I20 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
```

```
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number)):
      return super(I20, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C20(j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] - Ind(i1, i2) * Ind(j1, j2))
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I20
    """
    return I20(*self.args, **hints)
```

Listing 82: **Approximation of iterated Itô stochastic integral** $I_{(02)\tau_{p+1},\tau_p}^{(i_1 i_2)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c02 import C02
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I02(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates new I02 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
```

```
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number)):
      return super(I02, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C02(j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] - Ind(i1, i2) * Ind(j1, j2))
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I02
    """
    return I02(*self.args, **hints)
```

Listing 83: **Approximation of iterated Itô stochastic integral** $I_{(11)\tau_{p+1},\tau_p}^{(i_1 i_2)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c11 import C11
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I11(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates new I11 object with given args
    Parameters
    ==========
    i1 : int
```

```
      integral index
    i2 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number)):
      return super(I11, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C11(j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] - Ind(i1, i2) * Ind(j1, j2))
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I11
    """
    return I11(*self.args, **hints)
```

Listing 84: **Approximation of iterated Itô stochastic integral** $I_{(1000)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c1000 import C1000
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I1000(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 7

  def __new__(cls, *args, **kwargs):
    """
    Creates new I1000 object with given args
```

```
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    i4 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, i4, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(i4, Number) and
        isinstance(q, Number)):
      return super(I1000, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C1000(j4, j3, j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] -
       Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] * ksi[j4, i4] -
       Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] * ksi[j4, i4] -
       Ind(i1, i4) * Ind(j1, j4) * ksi[j2, i2] * ksi[j3, i3] -
       Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1] * ksi[j4, i4] -
       Ind(i2, i4) * Ind(j2, j4) * ksi[j1, i1] * ksi[j3, i3] -
       Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1] * ksi[j2, i2] +
       Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i4) * Ind(j3, j4) +
       Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i4) * Ind(j2, j4) +
       Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i3) * Ind(j2, j3))
      for j4 in range(q + 1)
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I1000
    """
```

```
71      return I1000(*self.args, **hints)
```

Listing 85: **Approximation of iterated Itô stochastic integral** $I_{(0100)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4)}$

```
1   from sympy import Function, Number, sympify, Add
2
3   from mathematics.sde.nonlinear.symbolic.coefficients.c0100 import C0100
4   from mathematics.sde.nonlinear.symbolic.ind import Ind
5
6
7   class I0100(Function):
8     """
9     Iterated Ito stochastic integral
10    """
11    nargs = 7
12
13    def __new__(cls, *args, **kwargs):
14      """
15      Creates new I0100 object with given args
16      Parameters
17      ==========
18      i1 : int
19        integral index
20      i2 : int
21        integral index
22      i3 : int
23        integral index
24      i4 : int
25        integral index
26      q : int
27        amount of terms in approximation of
28        iterated Ito stochastic integral
29      dt : float
30        delta time
31      ksi : numpy.ndarray
32        matrix of Gaussian random variables
33      Returns
34      =======
35      sympy.Expr
36        formula to simplify and substitute
37      """
38      i1, i2, i3, i4, q, dt, ksi = sympify(args)
39
40      if not (isinstance(i1, Number) and
41          isinstance(i2, Number) and
42          isinstance(i3, Number) and
43          isinstance(i4, Number) and
44          isinstance(q, Number)):
45        return super(I0100, cls).__new__(cls, *args, **kwargs)
46
47      return Add(*[
48        C0100(j4, j3, j2, j1, dt) *
49        (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] -
```

```
        Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] * ksi[j4, i4] -
        Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] * ksi[j4, i4] -
        Ind(i1, i4) * Ind(j1, j4) * ksi[j2, i2] * ksi[j3, i3] -
        Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1] * ksi[j4, i4] -
        Ind(i2, i4) * Ind(j2, j4) * ksi[j1, i1] * ksi[j3, i3] -
        Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1] * ksi[j2, i2] +
        Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i4) * Ind(j3, j4) +
        Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i4) * Ind(j2, j4) +
        Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i3) * Ind(j2, j3))
      for j4 in range(q + 1)
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I0100
    """
    return I0100(*self.args, **hints)
```

Listing 86: **Approximation of iterated Itô stochastic integral** $I_{(0010)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c0010 import C0010
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I0010(Function):
  """
  Iterated Ito stochastic integral
  """
  nargs = 7

  def __new__(cls, *args, **kwargs):
    """
    Creates new I0010 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    i3 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Ito stochastic integral
```

```
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, i4, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(i4, Number) and
        isinstance(q, Number)):
      return super(I0010, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C0010(j4, j3, j2, j1, dt) *
      (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] -
       Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] * ksi[j4, i4] -
       Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] * ksi[j4, i4] -
       Ind(i1, i4) * Ind(j1, j4) * ksi[j2, i2] * ksi[j3, i3] -
       Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1] * ksi[j4, i4] -
       Ind(i2, i4) * Ind(j2, j4) * ksi[j1, i1] * ksi[j3, i3] -
       Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1] * ksi[j2, i2] +
       Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i4) * Ind(j3, j4) +
       Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i4) * Ind(j2, j4) +
       Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i3) * Ind(j2, j3))
      for j4 in range(q + 1)
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    I0010
    """
    return I0010(*self.args, **hints)
```

Listing 87: **Approximation of iterated Itô stochastic integral** $I_{(0001)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c0001 import C0001
from mathematics.sde.nonlinear.symbolic.ind import Ind


class I0001(Function):
```

```
8   """
9   Iterated Ito stochastic integral
10  """
11  nargs = 7
12
13  def __new__(cls, *args, **kwargs):
14    """
15    Creates new I0001 object with given args
16    Parameters
17    ==========
18    i1 : int
19      integral index
20    i2 : int
21      integral index
22    i3 : int
23      integral index
24    i4 : int
25      integral index
26    q : int
27      amount of terms in approximation of
28      iterated Ito stochastic integral
29    dt : float
30      delta time
31    ksi : numpy.ndarray
32      matrix of Gaussian random variables
33    Returns
34    =======
35    sympy.Expr
36      formula to simplify and substitute
37    """
38    i1, i2, i3, i4, q, dt, ksi = sympify(args)
39
40    if not (isinstance(i1, Number) and
41        isinstance(i2, Number) and
42        isinstance(i3, Number) and
43        isinstance(i4, Number) and
44        isinstance(q, Number)):
45      return super(I0001, cls).__new__(cls, *args, **kwargs)
46
47    return Add(*[
48      C0001(j4, j3, j2, j1, dt) *
49      (ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] -
50       Ind(i1, i2) * Ind(j1, j2) * ksi[j3, i3] * ksi[j4, i4] -
51       Ind(i1, i3) * Ind(j1, j3) * ksi[j2, i2] * ksi[j4, i4] -
52       Ind(i1, i4) * Ind(j1, j4) * ksi[j2, i2] * ksi[j3, i3] -
53       Ind(i2, i3) * Ind(j2, j3) * ksi[j1, i1] * ksi[j4, i4] -
54       Ind(i2, i4) * Ind(j2, j4) * ksi[j1, i1] * ksi[j3, i3] -
55       Ind(i3, i4) * Ind(j3, j4) * ksi[j1, i1] * ksi[j2, i2] +
56       Ind(i1, i2) * Ind(j1, j2) * Ind(i3, i4) * Ind(j3, j4) +
57       Ind(i1, i3) * Ind(j1, j3) * Ind(i2, i4) * Ind(j2, j4) +
58       Ind(i1, i4) * Ind(j1, j4) * Ind(i2, i3) * Ind(j2, j3))
59      for j4 in range(q + 1)
60      for j3 in range(q + 1)
61      for j2 in range(q + 1)
62      for j1 in range(q + 1)])
```

```
63
64    def doit(self, **hints):
65        """
66        Tries to expand or calculate function
67        Returns
68        =======
69        I0001
70        """
71        return I0001(*self.args, **hints)
```

Listing 88: **Approximation of iterated Itô stochastic integral** $I_{(000000)\tau_{p+1},\tau_p}^{(i_1 i_2 i_3 i_4 i_5 i_6)}$

```
1  from sympy import Function, Number, sympify, Add
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c000000 import C000000
4  from mathematics.sde.nonlinear.symbolic.ind import Ind
5
6
7  class I000000(Function):
8      """
9      Iterated Ito stochastic integral
10     """
11     nargs = 9
12
13     def __new__(cls, *args, **kwargs):
14         """
15         Creates new I000000 object with given args
16         Parameters
17         ==========
18         i1 : int
19             integral index
20         i2 : int
21             integral index
22         i3 : int
23             integral index
24         i4 : int
25             integral index
26         i5 : int
27             integral index
28         i6 : int
29             integral index
30         q : int
31             amount of terms in approximation of
32             iterated Ito stochastic integral
33         dt : float
34             delta time
35         ksi : numpy.ndarray
36             matrix of Gaussian random variables
37         Returns
38         =======
39         sympy.Expr
40             formula to simplify and substitute
41         """
```

```
    i1, i2, i3, i4, i5, i6, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(i4, Number) and
        isinstance(i5, Number) and
        isinstance(i6, Number) and
        isinstance(q, Number) and
        isinstance(dt, Number)):
      return super(I000000, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C000000(j6, j5, j4, j3, j2, j1, dt) *
      (ksi[j1,i1] * ksi[j2,i2] * ksi[j3,i3] *
       ksi[j4,i4] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j1,j6) * Ind(i1,i6) * ksi[j2,i2] * ksi[j3,i3] * ksi[j4,i4] * ksi[j5,i5] -
       Ind(j2,j6) * Ind(i2,i6) * ksi[j1,i1] * ksi[j3,i3] * ksi[j4,i4] * ksi[j5,i5] -
       Ind(j3,j6) * Ind(i3,i6) * ksi[j1,i1] * ksi[j2,i2] * ksi[j4,i4] * ksi[j5,i5] -
       Ind(j4,j6) * Ind(i4,i6) * ksi[j1,i1] * ksi[j2,i2] * ksi[j3,i3] * ksi[j5,i5] -
       Ind(j5,j6) * Ind(i5,i6) * ksi[j1,i1] * ksi[j2,i2] * ksi[j3,i3] * ksi[j4,i4] -
       Ind(j1,j2) * Ind(i1,i2) * ksi[j3,i3] * ksi[j4,i4] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j1,j3) * Ind(i1,i3) * ksi[j2,i2] * ksi[j4,i4] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j1,j4) * Ind(i1,i4) * ksi[j2,i2] * ksi[j3,i3] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j1,j5) * Ind(i1,i5) * ksi[j2,i2] * ksi[j4,i4] * ksi[j3,i3] * ksi[j6,i6] -
       Ind(j2,j3) * Ind(i2,i3) * ksi[j1,i1] * ksi[j4,i4] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j2,j4) * Ind(i2,i4) * ksi[j1,i1] * ksi[j3,i3] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j2,j5) * Ind(i2,i5) * ksi[j1,i1] * ksi[j3,i3] * ksi[j4,i4] * ksi[j6,i6] -
       Ind(j3,j4) * Ind(i3,i4) * ksi[j1,i1] * ksi[j2,i2] * ksi[j5,i5] * ksi[j6,i6] -
       Ind(j3,j5) * Ind(i3,i5) * ksi[j1,i1] * ksi[j2,i2] * ksi[j4,i4] * ksi[j6,i6] -
       Ind(j4,j5) * Ind(i4,i5) * ksi[j1,i1] * ksi[j2,i2] * ksi[j3,i3] * ksi[j6,i6] +
       Ind(j1,j2) * Ind(i1,i2) * Ind(j3,j4) * Ind(i3,i4) * ksi[j5,i5] * ksi[j6,i6] +
       Ind(j1,j2) * Ind(i1,i2) * Ind(j3,j5) * Ind(i3,i5) * ksi[j4,i4] * ksi[j6,i6] +
       Ind(j1,j2) * Ind(i1,i2) * Ind(j4,j5) * Ind(i4,i5) * ksi[j3,i3] * ksi[j6,i6] +
       Ind(j1,j3) * Ind(i1,i3) * Ind(j2,j4) * Ind(i2,i4) * ksi[j5,i5] * ksi[j6,i6] +
       Ind(j1,j3) * Ind(i1,i3) * Ind(j2,j5) * Ind(i2,i5) * ksi[j4,i4] * ksi[j6,i6] +
       Ind(j1,j3) * Ind(i1,i3) * Ind(j4,j5) * Ind(i4,i5) * ksi[j2,i2] * ksi[j6,i6] +
       Ind(j1,j4) * Ind(i1,i4) * Ind(j2,j3) * Ind(i2,i3) * ksi[j5,i5] * ksi[j6,i6] +
       Ind(j1,j4) * Ind(i1,i4) * Ind(j2,j5) * Ind(i2,i5) * ksi[j3,i3] * ksi[j6,i6] +
       Ind(j1,j4) * Ind(i1,i4) * Ind(j3,j5) * Ind(i3,i5) * ksi[j2,i2] * ksi[j6,i6] +
       Ind(j1,j5) * Ind(i1,i5) * Ind(j2,j3) * Ind(i2,i3) * ksi[j4,i4] * ksi[j6,i6] +
       Ind(j1,j5) * Ind(i1,i5) * Ind(j2,j4) * Ind(i2,i4) * ksi[j3,i3] * ksi[j6,i6] +
       Ind(j1,j5) * Ind(i1,i5) * Ind(j3,j4) * Ind(i3,i4) * ksi[j2,i2] * ksi[j6,i6] +
       Ind(j2,j3) * Ind(i2,i3) * Ind(j4,j5) * Ind(i4,i5) * ksi[j1,i1] * ksi[j6,i6] +
       Ind(j2,j4) * Ind(i2,i4) * Ind(j3,j5) * Ind(i3,i5) * ksi[j1,i1] * ksi[j6,i6] +
       Ind(j2,j5) * Ind(i2,i5) * Ind(j3,j4) * Ind(i3,i4) * ksi[j1,i1] * ksi[j6,i6] +
       Ind(j6,j1) * Ind(i6,i1) * Ind(j3,j4) * Ind(i3,i4) * ksi[j2,i2] * ksi[j5,i5] +
       Ind(j6,j1) * Ind(i6,i1) * Ind(j3,j5) * Ind(i3,i5) * ksi[j2,i2] * ksi[j4,i4] +
       Ind(j6,j1) * Ind(i6,i1) * Ind(j2,j5) * Ind(i2,i5) * ksi[j3,i3] * ksi[j4,i4] +
       Ind(j6,j1) * Ind(i6,i1) * Ind(j2,j4) * Ind(i2,i4) * ksi[j3,i3] * ksi[j5,i5] +
       Ind(j6,j1) * Ind(i6,i1) * Ind(j4,j5) * Ind(i4,i5) * ksi[j2,i2] * ksi[j3,i3] +
       Ind(j6,j1) * Ind(i6,i1) * Ind(j2,j3) * Ind(i2,i3) * ksi[j4,i4] * ksi[j5,i5] +
       Ind(j6,j2) * Ind(i6,i2) * Ind(j3,j5) * Ind(i3,i5) * ksi[j1,i1] * ksi[j4,i4] +
       Ind(j6,j2) * Ind(i6,i2) * Ind(j4,j5) * Ind(i4,i5) * ksi[j1,i1] * ksi[j3,i3] +
       Ind(j6,j2) * Ind(i6,i2) * Ind(j3,j4) * Ind(i3,i4) * ksi[j1,i1] * ksi[j5,i5] +
```

```
97        Ind(j6,j2) * Ind(i6,i2) * Ind(j1,j5) * Ind(i1,i5) * ksi[j3,i3] * ksi[j4,i4] +
98        Ind(j6,j2) * Ind(i6,i2) * Ind(j1,j4) * Ind(i1,i4) * ksi[j3,i3] * ksi[j5,i5] +
99        Ind(j6,j2) * Ind(i6,i2) * Ind(j1,j3) * Ind(i1,i3) * ksi[j4,i4] * ksi[j5,i5] +
100       Ind(j6,j3) * Ind(i6,i3) * Ind(j2,j5) * Ind(i2,i5) * ksi[j1,i1] * ksi[j4,i4] +
101       Ind(j6,j3) * Ind(i6,i3) * Ind(j4,j5) * Ind(i4,i5) * ksi[j1,i1] * ksi[j2,i2] +
102       Ind(j6,j3) * Ind(i6,i3) * Ind(j2,j4) * Ind(i2,i4) * ksi[j1,i1] * ksi[j5,i5] +
103       Ind(j6,j3) * Ind(i6,i3) * Ind(j1,j5) * Ind(i1,i5) * ksi[j2,i2] * ksi[j4,i4] +
104       Ind(j6,j3) * Ind(i6,i3) * Ind(j1,j4) * Ind(i1,i4) * ksi[j2,i2] * ksi[j5,i5] +
105       Ind(j6,j3) * Ind(i6,i3) * Ind(j1,j2) * Ind(i1,i2) * ksi[j4,i4] * ksi[j5,i5] +
106       Ind(j6,j4) * Ind(i6,i4) * Ind(j3,j5) * Ind(i3,i5) * ksi[j1,i1] * ksi[j2,i2] +
107       Ind(j6,j4) * Ind(i6,i4) * Ind(j2,j5) * Ind(i2,i5) * ksi[j1,i1] * ksi[j3,i3] +
108       Ind(j6,j4) * Ind(i6,i4) * Ind(j2,j3) * Ind(i2,i3) * ksi[j1,i1] * ksi[j5,i5] +
109       Ind(j6,j4) * Ind(i6,i4) * Ind(j1,j5) * Ind(i1,i5) * ksi[j2,i2] * ksi[j3,i3] +
110       Ind(j6,j4) * Ind(i6,i4) * Ind(j1,j3) * Ind(i1,i3) * ksi[j2,i2] * ksi[j5,i5] +
111       Ind(j6,j4) * Ind(i6,i4) * Ind(j1,j2) * Ind(i1,i2) * ksi[j3,i3] * ksi[j5,i5] +
112       Ind(j6,j5) * Ind(i6,i5) * Ind(j3,j4) * Ind(i3,i4) * ksi[j1,i1] * ksi[j2,i2] +
113       Ind(j6,j5) * Ind(i6,i5) * Ind(j2,j4) * Ind(i2,i4) * ksi[j1,i1] * ksi[j3,i3] +
114       Ind(j6,j5) * Ind(i6,i5) * Ind(j2,j3) * Ind(i2,i3) * ksi[j1,i4] * ksi[j1,i4] +
115       Ind(j6,j5) * Ind(i6,i5) * Ind(j1,j4) * Ind(i1,i4) * ksi[j2,i2] * ksi[j3,i3] +
116       Ind(j6,j5) * Ind(i6,i5) * Ind(j1,j3) * Ind(i1,i3) * ksi[j2,i2] * ksi[j4,i4] +
117       Ind(j6,j5) * Ind(i6,i5) * Ind(j1,j2) * Ind(i1,i2) * ksi[j3,i3] * ksi[j4,i4] -
118       Ind(j6,j1) * Ind(i6,i1) * Ind(j2,j5) * Ind(i2,i5) * Ind(j3,j4) * Ind(i3,i4) -
119       Ind(j6,j1) * Ind(i6,i1) * Ind(j2,j4) * Ind(i2,i4) * Ind(j3,j5) * Ind(i3,i5) -
120       Ind(j6,j1) * Ind(i6,i1) * Ind(j2,j3) * Ind(i2,i3) * Ind(j4,j5) * Ind(i4,i5) -
121       Ind(j6,j2) * Ind(i6,i2) * Ind(j1,j5) * Ind(i1,i5) * Ind(j3,j4) * Ind(i3,i4) -
122       Ind(j6,j2) * Ind(i6,i2) * Ind(j1,j4) * Ind(i1,i4) * Ind(j3,j5) * Ind(i3,i5) -
123       Ind(j6,j2) * Ind(i6,i2) * Ind(j1,j3) * Ind(i1,i3) * Ind(j4,j5) * Ind(i4,i5) -
124       Ind(j6,j3) * Ind(i6,i3) * Ind(j1,j5) * Ind(i1,i5) * Ind(j2,j4) * Ind(i2,i4) -
125       Ind(j6,j3) * Ind(i6,i3) * Ind(j1,j4) * Ind(i1,i4) * Ind(j2,j5) * Ind(i2,i5) -
126       Ind(j3,j6) * Ind(i3,i6) * Ind(j1,j2) * Ind(i1,i2) * Ind(j4,j5) * Ind(i4,i5) -
127       Ind(j6,j4) * Ind(i6,i4) * Ind(j1,j5) * Ind(i1,i5) * Ind(j2,j3) * Ind(i2,i3) -
128       Ind(j6,j4) * Ind(i6,i4) * Ind(j1,j3) * Ind(i1,i3) * Ind(j2,j5) * Ind(i2,i5) -
129       Ind(j6,j4) * Ind(i6,i4) * Ind(j1,j2) * Ind(i1,i2) * Ind(j3,j5) * Ind(i3,i5) -
130       Ind(j6,j5) * Ind(i6,i5) * Ind(j1,j4) * Ind(i1,i4) * Ind(j2,j3) * Ind(i2,i3) -
131       Ind(j6,j5) * Ind(i6,i5) * Ind(j1,j2) * Ind(i1,i2) * Ind(j3,j4) * Ind(i3,i4) -
132       Ind(j6,j5) * Ind(i6,i5) * Ind(j1,j3) * Ind(i1,i3) * Ind(j2,j4) * Ind(i2,i4))
133     for j6 in range(q + 1)
134     for j5 in range(q + 1)
135     for j4 in range(q + 1)
136     for j3 in range(q + 1)
137     for j2 in range(q + 1)
138     for j1 in range(q + 1)])
139
140 def doit(self, **hints):
141   """
142   Tries to expand or calculate function
143   Returns
144   =======
145   I000000
146   """
147   return I000000(*self.args, **hints)
```

6.2.4 Source Codes for Iterated Stratonovich Stochastic Integrals Approximations Subprograms

Listing 89: **Approximation of Stratonovich stochastic integral** $I^{*(i_1)}_{(0)\tau_{p+1},\tau_p}$

```
from math import sqrt

from sympy import Function, sympify, Number


class J0(Function):
  """
  Stratonovich stochastic integral
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates new J0 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, dt, ksi = sympify(args)

    if not isinstance(i1, Number):
      return super(J0, cls).__new__(cls, *args, **kwargs)

    return ksi[0, i1] * sqrt(dt)

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J0
    """
    return J0(*self.args, **hints)
```

Listing 90: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2)}_{(00)\tau_{p+1},\tau_p}$

```
from math import sqrt
```

```
2
3  from sympy import sympify, Number, Function, Add
4
5
6  class J00(Function):
7    """
8    Iterated Stratonovich stochastic integral
9    """
10   nargs = 5
11
12   def __new__(cls, *args, **kwargs):
13     """
14     Creates new J00 object with given args
15     Parameters
16     ==========
17     i1 : int
18       integral index
19     i2 : int
20       integral index
21     q : int
22       amount of terms in approximation of
23       iterated Stratonovich stochastic integral
24     dt : float
25       delta time
26     ksi : numpy.ndarray
27       matrix of Gaussian random variables
28     Returns
29     =======
30     sympy.Expr
31       formula to simplify and substitute
32     """
33     i1, i2, q, dt, ksi = sympify(args)
34
35     if not (isinstance(i1, Number) and
36         isinstance(i2, Number) and
37         isinstance(q, Number)):
38       return super(J00, cls).__new__(cls, *args, **kwargs)
39
40     return \
41       (ksi[0, i1] * ksi[0, i2] +
42        Add(*[
43          (ksi[j1 - 1, i1] * ksi[j1, i2] -
44          ksi[j1, i1] * ksi[j1 - 1, i2]) /
45          sqrt(j1 ** 2 * 4 - 1)
46          for j1 in range(1, q + 1)])) * dt / 2
47
48   def doit(self, **hints):
49     """
50     Tries to expand or calculate function
51     Returns
52     =======
53     J00
54     """
55     return J00(*self.args, **hints)
```

Listing 91: **Approximation of Stratonovich stochastic integral** $I^{*(i_1)}_{(1)T,t}$

```
from math import sqrt

from sympy import Function, sympify, Number


class J1(Function):
  """
  Stratonovich stochastic integral
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates new J1 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, dt, ksi = sympify(args)

    if not isinstance(i1, Number):
      return super(J1, cls).__new__(cls, *args, **kwargs)

    return -(ksi[0, i1] + ksi[1, i1] / sqrt(3)) * dt ** 1.5 / 2

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J1
    """
    return J1(*self.args, **hints)
```

Listing 92: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3)}_{(000)\tau_{p+1},\tau_p}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c000 import C000


class J000(Function):
  """
```

```
    Iterated Stratonovich stochastic integral
    """
    nargs = 6

    def __new__(cls, *args, **kwargs):
        """
        Creates new J000 object with given args
        Parameters
        ==========
        i1 : int
            integral index
        i2 : int
            integral index
        i3 : int
            integral index
        q : int
            amount of terms in approximation of
            iterated Stratonovich stochastic integral
        dt : float
            delta time
        ksi : numpy.ndarray
            matrix of Gaussian random variables
        Returns
        =======
        sympy.Expr
            formula to simplify and substitute
        """
        i1, i2, i3, q, dt, ksi = sympify(args)

        if not (isinstance(i1, Number) and
                isinstance(i2, Number) and
                isinstance(i3, Number) and
                isinstance(q, Number)):
            return super(J000, cls).__new__(cls, *args, **kwargs)

        return Add(*[
            C000(j3, j2, j1, dt) *
            ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3]
            for j3 in range(q + 1)
            for j2 in range(q + 1)
            for j1 in range(q + 1)])

    def doit(self, **hints):
        """
        Tries to expand or calculate function
        Returns
        =======
        J000
        """
        return J000(*self.args, **hints)
```

Listing 93: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2)}_{(10)\tau_{p+1},\tau_p}$

```
from sympy import Function, sympify, Number, Add
```

```
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c10 import C10
4
5
6  class J10(Function):
7    """
8    Iterated Stratonovich stochastic integral
9    """
10   nargs = 5
11
12   def __new__(cls, *args, **kwargs):
13     """
14     Creates new J10 object with given args
15     Parameters
16     ==========
17     i1 : int
18       integral index
19     i2 : int
20       integral index
21     q : int
22       amount of terms in approximation of
23       iterated Stratonovich stochastic integral
24     dt : float
25       delta time
26     ksi : numpy.ndarray
27       matrix of Gaussian random variables
28     Returns
29     =======
30     sympy.Expr
31       formula to simplify and substitute
32     """
33     i1, i2, q, dt, ksi = sympify(args)
34
35     if not (isinstance(i1, Number) and
36         isinstance(i2, Number) and
37         isinstance(q, Number)):
38       return super(J10, cls).__new__(cls, *args, **kwargs)
39
40     return Add(*[
41       C10(j2, j1, dt) *
42       ksi[j1, i1] * ksi[j2, i2]
43       for j2 in range(q + 1)
44       for j1 in range(q + 1)])
45
46   def doit(self, **hints):
47     """
48     Tries to expand or calculate function
49     Returns
50     =======
51     J10
52     """
53     return J10(*self.args, **hints)
```

Listing 94: **Approximation of iterated Stratonovich stochastic integral** $I_{(01)\tau_{p+1},\tau_p}^{*(i_1 i_2)}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c01 import C01


class J01(Function):
  """
  Iterated Stratonovich stochastic integral
  """

  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates new J01 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(q, Number)):
      return super(J01, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C01(j2, j1, dt) *
      ksi[j1, i1] * ksi[j2, i2]
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J01
    """
```

```
54      return J01(*self.args, **hints)
```

Listing 95: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3 i_4)}_{(0000)\tau_{p+1},\tau_p}$

```
1   from sympy import Function, sympify, Number, Add
2
3   from mathematics.sde.nonlinear.symbolic.coefficients.c0000 import C0000
4
5
6   class J0000(Function):
7     """
8     Iterated Stratonovich stochastic integral
9     """
10    nargs = 7
11
12    def __new__(cls, *args, **kwargs):
13      """
14      Creates new J0000 object with given args
15      Parameters
16      ==========
17      i1 : int
18        integral index
19      i2 : int
20        integral index
21      i3 : int
22        integral index
23      i4 : int
24        integral index
25      q : int
26        amount of terms in approximation of
27        iterated Stratonovich stochastic integral
28      dt : float
29        delta time
30      ksi : numpy.ndarray
31        matrix of Gaussian random variables
32      Returns
33      =======
34      sympy.Expr
35        formula to simplify and substitute
36      """
37      i1, i2, i3, i4, q, dt, ksi = sympify(args)
38
39      if not (isinstance(i1, Number) and
40          isinstance(i2, Number) and
41          isinstance(i3, Number) and
42          isinstance(i4, Number) and
43          isinstance(q, Number)):
44        return super(J0000, cls).__new__(cls, *args, **kwargs)
45
46      return Add(*[
47        C0000(j4, j3, j2, j1, dt) *
48        ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4]
49        for j4 in range(q + 1)
```

```
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J0000
    """
    return J0000(*self.args, **hints)
```

Listing 96: **Approximation of Stratonovich stochastic integral** $I^{*(i_1)}_{(2)\tau_{p+1},\tau_p}$

```
from math import sqrt

from sympy import sympify, Number, Function


class J2(Function):
  """
  Stratonovich stochastic integral
  """
  nargs = 3

  def __new__(cls, *args, **kwargs):
    """
    Creates new J2 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, dt, ksi = sympify(args)

    if not isinstance(i1, Number):
      return super(J2, cls).__new__(cls, *args, **kwargs)

    return (ksi[0, i1] + ksi[1, i1] * sqrt(3) / 2 +
        ksi[2, i1] / sqrt(5) / 2) * dt ** 2.5 / 3

  def doit(self, **hints):
    """
    Tries to expand or calculate function
```

```
    Returns
    =======
    J2
    """
    return J2(*self.args, **hints)
```

Listing 97: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3)}_{(100)\tau_{p+1},\tau_p}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c100 import C100


class J100(Function):
  """
  Iterated Stratonovich stochastic integral
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates new J100 object with given args

    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(q, Number)):
      return super(J100, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C100(j3, j2, j1, dt) *
```

```
      ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3]
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J100
    """
    return J100(*self.args, **hints)
```

Listing 98: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3)}_{(010)\tau_{p+1},\tau_p}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c010 import C010


class J010(Function):
  """
  Iterated Stratonovich stochastic integral
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates new J010 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
```

```
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(q, Number)):
      return super(J010, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C010(j3, j2, j1, dt) *
      ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3]
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J010
    """
    return J010(*self.args, **hints)
```

Listing 99: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3)}_{(001)\tau_{p+1},\tau_p}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c001 import C001


class J001(Function):
  """
  Iterated Stratonovich stochastic integral
  """
  nargs = 6

  def __new__(cls, *args, **kwargs):
    """
    Creates new J001 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
```

```
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(q, Number)):
      return super(J001, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C001(j3, j2, j1, dt) *
      ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3]
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J001
    """
    return J001(*self.args, **hints)
```

Listing 100: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3 i_4 i_5)}_{(00000)\tau_{p+1},\tau_p}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c00000 import C00000


class J00000(Function):
  """
  Iterated Stratonovich stochastic integral
  """
  nargs = 8

  def __new__(cls, *args, **kwargs):
    """
    Creates new J00000 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    i4 : int
```

```
      integral index
    i5 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, i4, i5, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(i4, Number) and
        isinstance(i5, Number) and
        isinstance(q, Number) and
        isinstance(dt, Number)):
      return super(J00000, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C00000(j5, j4, j3, j2, j1, dt) *
      ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4] * ksi[j5, i5]
      for j5 in range(q + 1)
      for j4 in range(q + 1)
      for j3 in range(q + 1)
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J00000
    """
    return J00000(*self.args, **hints)
```

Listing 101: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2)}_{(20)\tau_{p+1},\tau_p}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c20 import C20


class J20(Function):
  """
```

```
    Iterated Stratonovich stochastic integral
    """
    nargs = 5

    def __new__(cls, *args, **kwargs):
        """
        Creates new J20 object with given args
        Parameters
        ==========
        i1 : int
            integral index
        i2 : int
            integral index
        q : int
            amount of terms in approximation of
            iterated Stratonovich stochastic integral
        dt : float
            delta time
        ksi : numpy.ndarray
            matrix of Gaussian random variables
        Returns
        =======
        sympy.Expr
            formula to simplify and substitute
        """
        i1, i2, q, dt, ksi = sympify(args)

        if not (isinstance(i1, Number) and
                isinstance(i2, Number)):
            return super(J20, cls).__new__(cls, *args, **kwargs)

        return Add(*[
            C20(j2, j1, dt) *
            ksi[j1, i1] * ksi[j2, i2]
            for j2 in range(q + 1)
            for j1 in range(q + 1)])

    def doit(self, **hints):
        """
        Tries to expand or calculate function
        Returns
        =======
        J20
        """
        return J20(*self.args, **hints)
```

Listing 102: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2)}_{(02)\tau_{p+1},\tau_p}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c02 import C02


```

```
class J02(Function):
  """
  Iterated Stratonovich stochastic integral
  """
  nargs = 5

  def __new__(cls, *args, **kwargs):
    """
    Creates new J02 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number)):
      return super(J02, cls).__new__(cls, *args, **kwargs)

    return Add(*[
      C02(j2, j1, dt) *
      ksi[j1, i1] * ksi[j2, i2]
      for j2 in range(q + 1)
      for j1 in range(q + 1)])

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    J02
    """
    return J02(*self.args, **hints)
```

Listing 103: **Approximation of iterated Stratonovich stochastic integral** $I_{(11)\tau_{p+1},\tau_p}^{*(i_1 i_2)}$

```
from sympy import Function, sympify, Number, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c11 import C11
```

```
4
5
6  class J11(Function):
7    """
8    Iterated Stratonovich stochastic integral
9    """
10   nargs = 5
11
12   def __new__(cls, *args, **kwargs):
13     """
14     Creates new J11 object with given args
15     Parameters
16     ==========
17     i1 : int
18       integral index
19     i2 : int
20       integral index
21     q : int
22       amount of terms in approximation of
23       iterated Stratonovich stochastic integral
24     dt : float
25       delta time
26     ksi : numpy.ndarray
27       matrix of Gaussian random variables
28     Returns
29     =======
30     sympy.Expr
31       formula to simplify and substitute
32     """
33     i1, i2, q, dt, ksi = sympify(args)
34
35     if not (isinstance(i1, Number) and
36         isinstance(i2, Number)):
37       return super(J11, cls).__new__(cls, *args, **kwargs)
38
39     return Add(*[
40       C11(j2, j1, dt) *
41       ksi[j1, i1] * ksi[j2, i2]
42       for j2 in range(q + 1)
43       for j1 in range(q + 1)])
44
45   def doit(self, **hints):
46     """
47     Tries to expand or calculate function
48     Returns
49     =======
50     J11
51     """
52     return J11(*self.args, **hints)
```

Listing 104: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3 i_4)}_{(1000)\tau_{p+1},\tau_p}$

```
1  from sympy import Function, Number, sympify, Add
```

```
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c1000 import C1000
4
5
6  class J1000(Function):
7    """
8    Iterated Stratonovich stochastic integral
9    """
10   nargs = 7
11
12   def __new__(cls, *args, **kwargs):
13     """
14     Creates new J1000 object with given args
15     Parameters
16     ==========
17     i1 : int
18       integral index
19     i2 : int
20       integral index
21     i3 : int
22       integral index
23     i4 : int
24       integral index
25     q : int
26       amount of terms in approximation of
27       iterated Stratonovich stochastic integral
28     dt : float
29       delta time
30     ksi : numpy.ndarray
31       matrix of Gaussian random variables
32     Returns
33     =======
34     sympy.Expr
35       formula to simplify and substitute
36     """
37     i1, i2, i3, i4, q, dt, ksi = sympify(args)
38
39     if not (isinstance(i1, Number) and
40         isinstance(i2, Number) and
41         isinstance(i3, Number) and
42         isinstance(i4, Number) and
43         isinstance(q, Number)):
44       return super(J1000, cls).__new__(cls, *args, **kwargs)
45
46     return Add(*[
47       C1000(j4, j3, j2, j1, dt) *
48       ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4]
49       for j4 in range(q + 1)
50       for j3 in range(q + 1)
51       for j2 in range(q + 1)
52       for j1 in range(q + 1)])
53
54   def doit(self, **hints):
55     """
56     Tries to expand or calculate function
```

```
    Returns
    =======
    J1000
    """
    return J1000(*self.args, **hints)
```

Listing 105: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3 i_4)}_{(0100)\tau_{p+1},\tau_p}$

```
from sympy import Function, Number, sympify, Add

from mathematics.sde.nonlinear.symbolic.coefficients.c0100 import C0100


class J0100(Function):
  """
  Iterated Stratonovich stochastic integral
  """
  nargs = 7

  def __new__(cls, *args, **kwargs):
    """
    Creates new J0100 object with given args
    Parameters
    ==========
    i1 : int
      integral index
    i2 : int
      integral index
    i3 : int
      integral index
    i4 : int
      integral index
    q : int
      amount of terms in approximation of
      iterated Stratonovich stochastic integral
    dt : float
      delta time
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i1, i2, i3, i4, q, dt, ksi = sympify(args)

    if not (isinstance(i1, Number) and
        isinstance(i2, Number) and
        isinstance(i3, Number) and
        isinstance(i4, Number) and
        isinstance(q, Number)):
      return super(J0100, cls).__new__(cls, *args, **kwargs)

```

```
46      return Add(*[
47        C0100(j4, j3, j2, j1, dt) *
48        ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4]
49        for j4 in range(q + 1)
50        for j3 in range(q + 1)
51        for j2 in range(q + 1)
52        for j1 in range(q + 1)])
53
54    def doit(self, **hints):
55      """
56      Tries to expand or calculate function
57      Returns
58      =======
59      J0100
60      """
61      return J0100(*self.args, **hints)
```

Listing 106: **Approximation of iterated Stratonovich stochastic integral** $I_{(0010)\tau_{p+1},\tau_p}^{*(i_1 i_2 i_3 i_4)}$

```
1   from sympy import Function, sympify, Number, Add
2
3   from mathematics.sde.nonlinear.symbolic.coefficients.c0010 import C0010
4
5
6   class J0010(Function):
7     """
8     Iterated Stratonovich stochastic integral
9     """
10    nargs = 7
11
12    def __new__(cls, *args, **kwargs):
13      """
14      Creates new J0010 object with given args
15      Parameters
16      ==========
17      i1 : int
18        integral index
19      i2 : int
20        integral index
21      i3 : int
22        integral index
23      i3 : int
24        integral index
25      q : int
26        amount of terms in approximation of
27        iterated Stratonovich stochastic integral
28      dt : float
29        delta time
30      ksi : numpy.ndarray
31        matrix of Gaussian random variables
32      Returns
33      =======
34      sympy.Expr
```

```
35        formula to simplify and substitute
36      """
37      i1, i2, i3, i4, q, dt, ksi = sympify(args)
38
39      if not (isinstance(i1, Number) and
40          isinstance(i2, Number) and
41          isinstance(i3, Number) and
42          isinstance(i4, Number) and
43          isinstance(q, Number)):
44        return super(J0010, cls).__new__(cls, *args, **kwargs)
45
46      return Add(*[
47        C0010(j4, j3, j2, j1, dt) *
48        ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4]
49        for j4 in range(q + 1)
50        for j3 in range(q + 1)
51        for j2 in range(q + 1)
52        for j1 in range(q + 1)])
53
54    def doit(self, **hints):
55      """
56      Tries to expand or calculate function
57      Returns
58      =======
59      J0010
60      """
61      return J0010(*self.args, **hints)
```

Listing 107: **Approximation of iterated Stratonovich stochastic integral** $I^{*(i_1 i_2 i_3 i_4)}_{(0001)\tau_{p+1},\tau_p}$

```
1  from sympy import Function, Number, sympify, Add
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c0001 import C0001
4
5
6  class J0001(Function):
7    """
8    Iterated Stratonovich stochastic integral
9    """
10   nargs = 7
11
12   def __new__(cls, *args, **kwargs):
13     """
14     Creates new J0001 object with given args
15     Parameters
16     ==========
17     i1 : int
18       integral index
19     i2 : int
20       integral index
21     i3 : int
22       integral index
23     i4 : int
```

```
24        integral index
25      q : int
26        amount of terms in approximation of
27        iterated Stratonovich stochastic integral
28      dt : float
29        delta time
30      ksi : numpy.ndarray
31        matrix of Gaussian random variables
32      Returns
33      =======
34      sympy.Expr
35        formula to simplify and substitute
36      """
37      i1, i2, i3, i4, q, dt, ksi = sympify(args)
38
39      if not (isinstance(i1, Number) and
40          isinstance(i2, Number) and
41          isinstance(i3, Number) and
42          isinstance(i4, Number) and
43          isinstance(q, Number)):
44        return super(J0001, cls).__new__(cls, *args, **kwargs)
45
46      return Add(*[
47        C0001(j4, j3, j2, j1, dt) *
48        ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] * ksi[j4, i4]
49        for j4 in range(q + 1)
50        for j3 in range(q + 1)
51        for j2 in range(q + 1)
52        for j1 in range(q + 1)])
53
54    def doit(self, **hints):
55      """
56      Tries to expand or calculate function
57      Returns
58      =======
59      J0001
60      """
61      return J0001(*self.args, **hints)
```

Listing 108: **Approximation of iterated Stratonovich stochastic integral** $I_{(000000)\tau_{p+1},\tau_p}^{*(i_1 i_2 i_3 i_4 i_5 i_6)}$

```
1  from sympy import Function, Number, sympify, Add
2
3  from mathematics.sde.nonlinear.symbolic.coefficients.c000000 import C000000
4
5
6  class J000000(Function):
7    """
8    Iterated Stratonovich stochastic integral
9    """
10   nargs = 9
11
12   def __new__(cls, *args, **kwargs):
```

```
13      """
14      Creates new J000000 object with given args
15      Parameters
16      ==========
17      i1 : int
18        integral index
19      i2 : int
20        integral index
21      i3 : int
22        integral index
23      i4 : int
24        integral index
25      i5 : int
26        integral index
27      i6 : int
28        integral index
29      q : int
30        amount of terms in approximation of
31        iterated Stratonovich stochastic integral
32      dt : float
33        delta time
34      ksi : numpy.ndarray
35        matrix of Gaussian random variables
36      Returns
37      =======
38      sympy.Expr
39        formula to simplify and substitute
40      """
41      i1, i2, i3, i4, i5, i6, q, dt, ksi = sympify(args)
42
43      if not (isinstance(i1, Number) and
44          isinstance(i2, Number) and
45          isinstance(i3, Number) and
46          isinstance(i4, Number) and
47          isinstance(i5, Number) and
48          isinstance(i6, Number) and
49          isinstance(q, Number) and
50          isinstance(dt, Number)):
51        return super(J000000, cls).__new__(cls, *args, **kwargs)
52
53      return Add(*[
54        C000000(j6, j5, j4, j3, j2, j1, dt) *
55        ksi[j1, i1] * ksi[j2, i2] * ksi[j3, i3] *
56        ksi[j4, i4] * ksi[j5, i5] * ksi[j6, i6]
57        for j6 in range(q + 1)
58        for j5 in range(q + 1)
59        for j4 in range(q + 1)
60        for j3 in range(q + 1)
61        for j2 in range(q + 1)
62        for j1 in range(q + 1)])
63
64    def doit(self, **hints):
65      """
66      Tries to expand or calculate function
67      Returns
```

```
    =======
    J000000
    """
    return J000000(*self.args, **hints)
```

6.2.5 Source Codes for Calculation of the Numbers q, q_1,..., q_{15}

Listing 109: **Calculation of the numbers** q, q_1,..., q_{15}

```
from mathematics.sde.nonlinear.symbolic.coefficients.c import C


def solve_q(i):
  """
  Calculates value for iteration of loop
  Parameters
  ==========
  i : int
    amount of members
  Returns
  =======
  values : float
    value for iteration of loop that calculates amount of q
  """
  return 1 / 4 - 1 / 2 * sum([
    1 / (4 * j ** 2 - 1)
    for j in range(1, i + 1)
  ])


def solve_q1(i):
  """
  Calculates value for iteration of loop
  Parameters
  ==========
  i : int
    amount of members
  Returns
  =======
  values : float
    value for iteration of loop that calculates amount of q
  """
  return 1 / 6 - 1 / 64 * sum([
    (2 * j1 + 1) *
    (2 * j2 + 1) *
    (2 * j3 + 1) *
    C((j1, j2, j3), (0, 0, 0)) ** 2
    for j1 in range(i + 1)
    for j2 in range(i + 1)
    for j3 in range(i + 1)
  ])
```

```
43
44
45 def solve_q2(i):
46     """
47     Calculates value for iteration of loop
48     Parameters
49     ==========
50     i : int
51       amount of members
52     Returns
53     =======
54     values : float
55       value for iteration of loop that calculates amount of q
56     """
57     return 1 / 4 - 1 / 64 * sum([
58       (2 * j1 + 1) *
59       (2 * j2 + 1) *
60       (C((j1, j2), (0, 1)) ** 2)
61       for j1 in range(i + 1)
62       for j2 in range(i + 1)
63     ])
64
65
66 def solve_q2_optional(i):
67     """
68     Calculates value for iteration of loop
69     Parameters
70     ==========
71     i : int
72       amount of members
73     Returns
74     =======
75     values : float
76       value for iteration of loop that calculates amount of q
77     """
78     return 1 / 12 - 1 / 64 * sum([
79       (2 * j1 + 1) *
80       (2 * j2 + 1) *
81       (C((j1, j2), (1, 0)) ** 2)
82       for j1 in range(i + 1)
83       for j2 in range(i + 1)
84     ])
85
86
87 def solve_q3(i):
88     """
89     Calculates value for iteration of loop
90     Parameters
91     ==========
92     i : int
93       amount of members
94     Returns
95     =======
96     values : float
97       value for iteration of loop that calculates amount of q
```

```
98    """
99    return 1 / 24 - 1 / 256 * sum([
100     (2 * j1 + 1) *
101     (2 * j2 + 1) *
102     (2 * j3 + 1) *
103     (2 * j4 + 1) *
104     (C((j1, j2, j3, j4), (0, 0, 0, 0)) ** 2)
105     for j1 in range(i + 1)
106     for j2 in range(i + 1)
107     for j3 in range(i + 1)
108     for j4 in range(i + 1)
109   ])
110
111
112 def solve_q4(i):
113   """
114   Calculates value for iteration of loop
115   Parameters
116   ==========
117   i : int
118     amount of members
119   Returns
120   =======
121   values : float
122     value for iteration of loop that calculates amount of q
123   """
124   return 1 / 120 - 1 / (32 ** 2) * sum([
125     (2 * j1 + 1) *
126     (2 * j2 + 1) *
127     (2 * j3 + 1) *
128     (2 * j4 + 1) *
129     (2 * j5 + 1) *
130     (C((j1, j2, j3, j4, j5), (0, 0, 0, 0, 0)) ** 2)
131     for j1 in range(i + 1)
132     for j2 in range(i + 1)
133     for j3 in range(i + 1)
134     for j4 in range(i + 1)
135     for j5 in range(i + 1)
136   ])
137
138
139 def solve_q5(i):
140   """
141   Calculates value for iteration of loop
142   Parameters
143   ==========
144   i : int
145     amount of members
146   Returns
147   =======
148   values : float
149     value for iteration of loop that calculates amount of q
150   """
151   return 1 / 60 - 1 / 256 * sum([
152     (2 * j1 + 1) *
```

```
153       (2 * j2 + 1) *
154       (C((j1, j2), (2, 0)) ** 2)
155       for j1 in range(i + 1)
156       for j2 in range(i + 1)
157     ])
158
159
160   def solve_q6(i):
161     """
162     Calculates value for iteration of loop
163     Parameters
164     ==========
165     i : int
166       amount of members
167     Returns
168     =======
169     values : float
170       value for iteration of loop that calculates amount of q
171     """
172     return 1 / 18 - 1 / 256 * sum([
173       (2 * j1 + 1) *
174       (2 * j2 + 1) *
175       (C((j1, j2), (1, 1)) ** 2)
176       for j1 in range(i + 1)
177       for j2 in range(i + 1)
178     ])
179
180
181   def solve_q7(i):
182     """
183     Calculates value for iteration of loop
184     Parameters
185     ==========
186     i : int
187       amount of members
188     Returns
189     =======
190     values : float
191       value for iteration of loop that calculates amount of q
192     """
193     return 1 / 6 - 1 / 256 * sum([
194       (2 * j1 + 1) *
195       (2 * j2 + 1) *
196       (C((j1, j2), (0, 2)) ** 2)
197       for j1 in range(i + 1)
198       for j2 in range(i + 1)
199     ])
200
201
202   def solve_q8(i):
203     """
204     Calculates value for iteration of loop
205     Parameters
206     ==========
207     i : int
```

```
208     amount of members
209   Returns
210   =======
211   values : float
212     value for iteration of loop that calculates amount of q
213   """
214   return 1 / 10 - 1 / 256 * sum([
215     (2 * j1 + 1) *
216     (2 * j2 + 1) *
217     (2 * j3 + 1) *
218     (C((j1, j2, j3), (0, 0, 1)) ** 2)
219     for j1 in range(i + 1)
220     for j2 in range(i + 1)
221     for j3 in range(i + 1)
222   ])
223
224
225 def solve_q9(i):
226   """
227   Calculates value for iteration of loop
228   Parameters
229   ==========
230   i : int
231     amount of members
232   Returns
233   =======
234   values : float
235     value for iteration of loop that calculates amount of q
236   """
237   return 1 / 20 - 1 / 256 * sum([
238     (2 * j1 + 1) *
239     (2 * j2 + 1) *
240     (2 * j3 + 1) *
241     (C((j1, j2, j3), (0, 1, 0)) ** 2)
242     for j1 in range(i + 1)
243     for j2 in range(i + 1)
244     for j3 in range(i + 1)
245   ])
246
247
248 def solve_q10(i):
249   """
250   Calculates value for iteration of loop
251   Parameters
252   ==========
253   i : int
254     amount of members
255   Returns
256   =======
257   values : float
258     value for iteration of loop that calculates amount of q
259   """
260   return 1 / 60 - 1 / 256 * sum([
261     (2 * j1 + 1) *
262     (2 * j2 + 1) *
```

```
263       (2 * j3 + 1) *
264       (C((j1, j2, j3), (1, 0, 0)) ** 2)
265       for j1 in range(i + 1)
266       for j2 in range(i + 1)
267       for j3 in range(i + 1)
268     ])
269
270
271   def solve_q11(i):
272     """
273     Calculates value for iteration of loop
274     Parameters
275     ==========
276     i : int
277       amount of members
278     Returns
279     =======
280     values : float
281       value for iteration of loop that calculates amount of q
282     """
283     return 1 / 36 - 1 / (32 ** 2) * sum([
284       (2 * j1 + 1) *
285       (2 * j2 + 1) *
286       (2 * j3 + 1) *
287       (2 * j4 + 1) *
288       (C((j1, j2, j3, j4), (0, 0, 0, 1)) ** 2)
289       for j1 in range(i + 1)
290       for j2 in range(i + 1)
291       for j3 in range(i + 1)
292       for j4 in range(i + 1)
293     ])
294
295
296   def solve_q12(i):
297     """
298     Calculates value for iteration of loop
299     Parameters
300     ==========
301     i : int
302       amount of members
303     Returns
304     =======
305     values : float
306       value for iteration of loop that calculates amount of q
307     """
308     return 1 / 60 - 1 / (32 ** 2) * sum([
309       (2 * j1 + 1) *
310       (2 * j2 + 1) *
311       (2 * j3 + 1) *
312       (2 * j4 + 1) *
313       (C((j1, j2, j3, j4), (0, 0, 1, 0)) ** 2)
314       for j1 in range(i + 1)
315       for j2 in range(i + 1)
316       for j3 in range(i + 1)
317       for j4 in range(i + 1)
```

```
318   ])
319
320
321 def solve_q13(i):
322   """
323   Calculates value for iteration of loop
324   Parameters
325   ==========
326   i : int
327     amount of members
328   Returns
329   =======
330   values : float
331     value for iteration of loop that calculates amount of q
332   """
333   return 1 / 120 - 1 / (32 ** 2) * sum([
334     (2 * j1 + 1) *
335     (2 * j2 + 1) *
336     (2 * j3 + 1) *
337     (2 * j4 + 1) *
338     (C((j1, j2, j3, j4), (0, 1, 0, 0)) ** 2)
339     for j1 in range(i + 1)
340     for j2 in range(i + 1)
341     for j3 in range(i + 1)
342     for j4 in range(i + 1)
343   ])
344
345
346 def solve_q14(i):
347   """
348   Calculates value for iteration of loop
349   Parameters
350   ==========
351   i : int
352     amount of members
353   Returns
354   =======
355   values : float
356     value for iteration of loop that calculates amount of q
357   """
358   return 1 / 360 - 1 / (32 ** 2) * sum([
359     (2 * j1 + 1) *
360     (2 * j2 + 1) *
361     (2 * j3 + 1) *
362     (2 * j4 + 1) *
363     (C((j1, j2, j3, j4), (1, 0, 0, 0)) ** 2)
364     for j1 in range(i + 1)
365     for j2 in range(i + 1)
366     for j3 in range(i + 1)
367     for j4 in range(i + 1)
368   ])
369
370
371 def solve_q15(i):
372   """
```

```
373     Calculates value for iteration of loop
374     Parameters
375     ==========
376     i : int
377       amount of members
378     Returns
379     =======
380     values : float
381       value for iteration of loop that calculates amount of q
382     """
383     return 1 / 720 - 1 / (64 ** 2) * sum([
384       (2 * j1 + 1) *
385       (2 * j2 + 1) *
386       (2 * j3 + 1) *
387       (2 * j4 + 1) *
388       (2 * j5 + 1) *
389       (2 * j6 + 1) *
390       (C((j1, j2, j3, j4, j5, j6), (0, 0, 0, 0, 0, 0)) ** 2)
391       for j1 in range(i + 1)
392       for j2 in range(i + 1)
393       for j3 in range(i + 1)
394       for j4 in range(i + 1)
395       for j5 in range(i + 1)
396       for j6 in range(i + 1)
397     ])
398
399
400 solvers = [
401     solve_q, solve_q1, solve_q2, solve_q3, solve_q8,
402     solve_q9, solve_q10, solve_q4, solve_q7, solve_q6,
403     solve_q5, solve_q11, solve_q12, solve_q13,
404     solve_q14, solve_q15,
405 ]
406
407 dt_degrees = [
408     [1],
409     [2, 1],
410     [3, 2, 1, 1],
411     [4, 3, 2, 2, 1, 1, 1, 1],
412     [5, 4, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1],
413 ]
414
415 q_ranges = [
416     1, 2, 4, 8, 16
417 ]
418
419
420 def loop(dt: float, k: float, degree: int, solver):
421     """
422     Loop that chooses amount of q that provides necessary accuracy
423     Parameters
424     ==========
425     dt : float
426       delta time
427     k : float
```

```
428       user chosen coefficient of accuracy
429     degree : int
430       degree of dt depending on q
431     solver : function
432       function that
433     Returns
434     =======
435     i : int
436       amount of q
437     """
438     i = 0
439     while True:
440       if solver(i) <= k * dt ** degree:
441         break
442       i += 1
443     return i
444
445
446 def get_q(dt: float, k: float, r: float):
447     """
448     Iterates solvers and get q values necessary to
449     achieve given accuracy
450     Parameters
451     ==========
452     dt: float
453       integration step
454     k: float
455       user chosen coefficient of accuracy
456     r: float
457       strong numerical scheme order
458     Returns
459     =======
460     qs_result: tuple
461       q values
462     """
463     qs_result = []
464
465     degree = int(r * 2)
466     range_id = degree - 2
467
468     for q_id in range(q_ranges[range_id]):
469       qs_result.append(loop(dt, k, dt_degrees[range_id][q_id], solvers[q_id]))
470
471     return tuple(qs_result)
```

6.2.6 Source Codes for Strong Taylor–Itô Numerical Schemes with Convergence Orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs

Listing 110: **Euler scheme modeling subprogram**

```
1 import logging
```

```
2  from time import time
3
4  import numpy as np
5  from sympy import lambdify, Matrix, symbols, MatrixSymbol, Symbol
6
7  from mathematics.sde.nonlinear.symbolic.schemes.euler import Euler
8
9
10 def euler(y0: np.array, a: Matrix, b: Matrix, times: tuple):
11     """
12     Performs modeling of Euler scheme
13     Parameters
14     ==========
15     y0 : numpy.ndarray
16         initial conditions
17     a : numpy.ndarray
18         vector function a
19     b : numpy.ndarray
20         matrix function b
21     times : tuple
22         integration limits and step
23     Returns
24     =======
25     y : numpy.ndarray
26         vector of solution
27     t : list
28         list of time moments
29     """
30     start_time = time()
31
32     logger = logging.getLogger(__name__)
33
34     logger.info(f"[{(time() - start_time):.3f} seconds] Euler start")
35
36     # Ranges
37     n = b.shape[0]
38     m = b.shape[1]
39     t1 = times[0]
40     dt = times[1]
41     t2 = times[2]
42
43     # Defining context
44     args = symbols(f"x1:{n + 1}")
45     ticks = int((t2 - t1) / dt)
46
47     # Symbols
48     sym_i, sym_t = Symbol("i"), Symbol("t")
49     sym_ksi = MatrixSymbol("ksi", 1, m)
50     sym_y = Euler(sym_i, Matrix(args), a, b, dt, sym_ksi)
51
52     args_extended = list()
53     args_extended.extend(args)
54     args_extended.extend([sym_t, sym_ksi])
55
56     # Compilation of formulas
```

```
57    y_compiled = list()
58    for tr in range(n):
59      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
60
61    logger.info(f"[{(time() - start_time):.3f} seconds] Euler subs are finished")
62
63    # Substitution values
64    t = [t1 + i * dt for i in range(ticks)]
65    y = np.zeros((n, ticks))
66    y[:, 0] = y0[:, 0]
67
68    # Dynamic substitutions with integration
69    for p in range(ticks - 1):
70      values = [*y[:, p], t[p], np.random.randn(1, m)]
71      for tr in range(n):
72        y[tr, p + 1] = y_compiled[tr](*values)
73
74    logger.info(f"[{(time() - start_time):.3f} seconds] Euler calculations are finished")
75
76    return y, t
```

Listing 111: **Euler scheme**

```
1   from sympy import Function, sympify, Add
2
3   from mathematics.sde.nonlinear.symbolic.ito.i0 import I0
4
5
6   class Euler(Function):
7     """
8     Euler scheme
9     """
10    nargs = 6
11
12    def __new__(cls, *args, **kwargs):
13      """
14      Creates new Euler object with given args
15      Parameters
16      ==========
17      i : int
18        component of stochastic process
19      yp : numpy.ndarray
20        initial conditions
21      a : numpy.ndarray
22        algebraic, given in the variables x and t
23      b : numpy.ndarray
24        algebraic, given in the variables x and t
25      dt : float
26        integration step
27      ksi : numpy.ndarray
28        matrix of Gaussian random variables
29      Returns
30      =======
```

```
31      sympy.Expr
32        formula to simplify and substitute
33      """
34      i, yp, a, b, dt, ksi = sympify(args)
35      n, m = b.shape[0], b.shape[1]
36
37      return Add(
38
39        yp[i, 0], a[i, 0] * dt,
40
41        *[b[i, i1] * I0(i1, dt, ksi)
42         for i1 in range(m)]
43
44      )
45
46    def doit(self, **hints):
47      """
48      Tries to expand or calculate function
49      Returns
50      =======
51      sympy.Expr
52      """
53      return Euler(*self.args, **hints)
```

Listing 112: **Milstein scheme modeling subprogram**

```
1   import logging
2   from time import time
3
4   import numpy as np
5   from sympy import symbols, MatrixSymbol, Matrix, lambdify
6
7   from mathematics.sde.nonlinear.q import get_q
8   from mathematics.sde.nonlinear.symbolic.schemes.milstein import Milstein
9
10
11  def milstein(y0: np.array, a: Matrix, b: Matrix, k: float, times: tuple):
12    """
13    Performs modeling of Milstein scheme
14    Parameters
15    ==========
16    y0 : numpy.ndarray
17      initial conditions
18    a : numpy.ndarray
19      vector function a
20    b : numpy.ndarray
21      matrix function b
22    q : tuple
23      amount of independent random variables
24    times : tuple
25      integration limits and step
26    Returns
27    =======
```

```
    y : numpy.ndarray
      vector of solution
    t : list
      list of time moments
    """
    start_time = time()

    logger = logging.getLogger(__name__)

    logger.info(f"[{(time() - start_time):.3f} seconds] Milstein start")

    # Ranges
    n = b.shape[0]
    m = b.shape[1]
    t1 = times[0]
    dt = times[1]
    t2 = times[2]

    # Defining context
    args = symbols(f"x1:{n + 1}")
    ticks = int((t2 - t1) / dt)
    q = get_q(dt, k, 1)
    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")

    # Symbols
    sym_i, sym_t = symbols("i t")
    sym_ksi = MatrixSymbol("ksi", q[0] + 2, m)
    sym_y = Milstein(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)

    args_extended = list()
    args_extended.extend(args)
    args_extended.extend([sym_t, sym_ksi])

    # Compilation of formulas
    y_compiled = list()
    for tr in range(n):
      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))

    logger.info(f"[{(time() - start_time):.3f} seconds] Milstein subs are finished")

    # Substitution values
    t = [t1 + i * dt for i in range(ticks)]
    y = np.zeros((n, ticks))
    y[:, 0] = y0[:, 0]

    # Dynamic substitutions with integration
    for p in range(ticks - 1):
      values = [*y[:, p], t[p], np.random.randn(q[0] + 2, m)]
      for tr in range(n):
        y[tr, p + 1] = y_compiled[tr](*values)

    logger.info(f"[{(time() - start_time):.3f} seconds] Milstein calculations are finished"
      )
```

```
82
83    return y, t
```

Listing 113: **Milstein scheme**

```
1   from sympy import Function, sympify, Add
2
3   from mathematics.sde.nonlinear.symbolic.g import G
4   from mathematics.sde.nonlinear.symbolic.ito.i0 import I0
5   from mathematics.sde.nonlinear.symbolic.ito.i00 import I00
6
7
8   class Milstein(Function):
9     """
10    Milstein scheme
11    """
12    nargs = 8
13
14    def __new__(cls, *args, **kwargs):
15      """
16      Creates new Milstein object with given args
17      Parameters
18      ==========
19      i : int
20        component of stochastic process
21      yp : numpy.ndarray
22        initial conditions
23      a : numpy.ndarray
24        algebraic, given in the variables x and t
25      b : numpy.ndarray
26        algebraic, given in the variables x and t
27      dt : float
28        integration step
29      ksi : numpy.ndarray
30        matrix of Gaussian random variables
31      q : tuple
32        amounts of q for integrals approximations
33      Returns
34      =======
35      sympy.Expr
36        formula to simplify and substitute
37      """
38      i, yp, a, b, dt, ksi, dxs, q = sympify(args)
39      n, m = b.shape[0], b.shape[1]
40
41      return Add(
42
43        yp[i, 0], a[i, 0] * dt,
44
45        *[b[i, i1] * I0(i1, dt, ksi)
46         for i1 in range(m)],
47
48        *[G(b[:, i1], b[i, i2], dxs) *
```

```
49          I00(i1, i2, q[0], dt, ksi)
50          for i2 in range(m)
51          for i1 in range(m)]
52
53      )
54
55    def doit(self, **hints):
56      """
57      Tries to expand or calculate function
58      Returns
59      =======
60      sympy.Expr
61      """
62      return Milstein(*self.args, **hints)
```

Listing 114: **Strong Taylor–Itô scheme with convergence order** 1.5 **modeling subprogram**

```
1  import logging
2  from time import time
3
4  import numpy as np
5  from sympy import Matrix, symbols, MatrixSymbol, lambdify
6
7  from mathematics.sde.nonlinear.q import get_q
8  from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_ito_1p5 import
       StrongTaylorIto1p5
9
10
11 def strong_taylor_ito_1p5(y0: np.array, a: Matrix, b: Matrix, k: float, times: tuple):
12   """
13   Performs modeling of Strong Taylor-Ito scheme with convergence order 1.5
14   Parameters
15   ==========
16   y0 : numpy.ndarray
17     initial conditions
18   a : numpy.ndarray
19     vector function a
20   b : numpy.ndarray
21     matrix function b
22   k : float
23     precision constant
24   times : tuple
25     integration limits and step
26   Returns
27   =======
28   y : numpy.ndarray
29     vector of solution
30   t : list
31     list of time moments
32   """
33   start_time = time()
34
35   logger = logging.getLogger(__name__)
```

```
36
37    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 1.5 start")
38
39    # Ranges
40    n = b.shape[0]
41    m = b.shape[1]
42    t1 = times[0]
43    dt = times[1]
44    t2 = times[2]
45
46    # Defining context
47    args = symbols(f"x1:{n + 1}")
48    ticks = int((t2 - t1) / dt)
49    q = get_q(dt, k, 1.5)
50    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 2, m)
57    sym_y = StrongTaylorIto1p5(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 1.5 subs are
        finished")
69
70    # Substitution values
71    t = [t1 + i * dt for i in range(ticks)]
72    y = np.zeros((n, ticks))
73    y[:, 0] = y0[:, 0]
74
75    # Dynamic substitutions with integration
76    for p in range(ticks - 1):
77      values = [*y[:, p], t[p], np.random.randn(q[0] + 2, m)]
78      for tr in range(n):
79        y[tr, p + 1] = y_compiled[tr](*values)
80
81    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 1.5 calculations
        are finished")
82
83    return y, t
```

Listing 115: **Strong Taylor–Itô scheme with convergence order** 1.5

```
1  from sympy import Function, sympify, Add
```

```
2
3  from mathematics.sde.nonlinear.symbolic.g import G
4  from mathematics.sde.nonlinear.symbolic.ito.i0 import I0
5  from mathematics.sde.nonlinear.symbolic.ito.i00 import I00
6  from mathematics.sde.nonlinear.symbolic.ito.i000 import I000
7  from mathematics.sde.nonlinear.symbolic.ito.i1 import I1
8  from mathematics.sde.nonlinear.symbolic.l import L
9
10
11 class StrongTaylorIto1p5(Function):
12   """
13   Strong Taylor-Ito scheme with convergence order 1.5
14   """
15   nargs = 8
16
17   def __new__(cls, *args, **kwargs):
18     """
19     Creates new StrongTaylorIto1p5 object with given args
20     Parameters
21     ==========
22     i : int
23       component of stochastic process
24     yp : numpy.ndarray
25       initial conditions
26     a : numpy.ndarray
27       algebraic, given in the variables x and t
28     b : numpy.ndarray
29       algebraic, given in the variables x and t
30     dt : float
31       integration step
32     ksi : numpy.ndarray
33       matrix of Gaussian random variables
34     q : tuple
35       amounts of q for stochastic integrals approximations
36     Returns
37     =======
38     sympy.Expr
39       formula to simplify and substitute
40     """
41     i, yp, a, b, dt, ksi, dxs, q = sympify(args)
42     n, m = b.shape[0], b.shape[1]
43
44     return Add(
45
46       yp[i, 0], a[i, 0] * dt,
47
48       *[b[i, i1] * I0(i1, dt, ksi)
49        for i1 in range(m)],
50
51       *[G(b[:, i1], b[i, i2], dxs) *
52        I00(i1, i2, q[0], dt, ksi)
53        for i2 in range(m)
54        for i1 in range(m)],
55
56       *[G(b[:, i1], a[i, 0], dxs) *
```

```
        (dt * I0(i1, dt, ksi) + I1(i1, dt, ksi)) -
        L(a, b, b[i, i1], dxs) *
        I1(i1, dt, ksi)
        for i1 in range(m)],

       *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
        I000(i1, i2, i3, q[1], dt, ksi)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       dt ** 2 / 2 * L(a, b, a[i, 0], dxs)

    )

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    sympy.Expr
    """
    return StrongTaylorIto1p5(*self.args, **hints)
```

Listing 116: **Strong Taylor–Itô scheme with convergence order** 2.0 **modeling subprogram**

```
import logging
from time import time

import numpy as np
from sympy import symbols, Matrix, MatrixSymbol, lambdify

from mathematics.sde.nonlinear.q import get_q
from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_ito_2p0 import
    StrongTaylorIto2p0


def strong_taylor_ito_2p0(y0: np.array, a: Matrix, b: Matrix, k: float, times: tuple):
  """
  Performs modeling of Strong Taylor-Ito scheme with convergence order 2.0
  Parameters
  ==========
  y0 : numpy.ndarray
    initial conditions
  a : numpy.ndarray
    vector function a
  b : numpy.ndarray
    matrix function b
  k : float
    precision constant
  times : tuple
    integration limits and step
```

```
    Returns
    =======
    y : numpy.ndarray
      vector of solution
    t : list
      list of time moments
    """
    start_time = time()

    logger = logging.getLogger(__name__)

    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 2.0 start")

    # Ranges
    n = b.shape[0]
    m = b.shape[1]
    t1 = times[0]
    dt = times[1]
    t2 = times[2]

    # Defining context
    args = symbols(f"x1:{n + 1}")
    ticks = int((t2 - t1) / dt)
    q = get_q(dt, k, 2)
    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")

    # Symbols
    sym_i, sym_t = symbols("i t")
    sym_ksi = MatrixSymbol("ksi", q[0] + 2, m)
    sym_y = StrongTaylorIto2p0(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)

    args_extended = list()
    args_extended.extend(args)
    args_extended.extend([sym_t, sym_ksi])

    # Compilation of formulas
    y_compiled = list()
    for tr in range(n):
      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))

    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 2.0 subs are
      finished")

    # Substitution values
    t = [t1 + i * dt for i in range(ticks)]
    y = np.zeros((n, ticks))
    y[:, 0] = y0[:, 0]

    # Dynamic substitutions with integration
    for p in range(ticks - 1):
      values = [*y[:, p], t[p], np.random.randn(q[0] + 2, m)]
      for tr in range(n):
        y[tr, p + 1] = y_compiled[tr](*values)
```

```
80
81    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 2.0 calculations
        are finished")
82
83    return y, t
```

Listing 117: **Strong Taylor–Itô scheme with convergence order** 2.0

```
1   from sympy import Function, sympify, Add
2
3   from mathematics.sde.nonlinear.symbolic.g import G
4   from mathematics.sde.nonlinear.symbolic.ito.i0 import I0
5   from mathematics.sde.nonlinear.symbolic.ito.i00 import I00
6   from mathematics.sde.nonlinear.symbolic.ito.i000 import I000
7   from mathematics.sde.nonlinear.symbolic.ito.i0000 import I0000
8   from mathematics.sde.nonlinear.symbolic.ito.i01 import I01
9   from mathematics.sde.nonlinear.symbolic.ito.i1 import I1
10  from mathematics.sde.nonlinear.symbolic.ito.i10 import I10
11  from mathematics.sde.nonlinear.symbolic.l import L
12
13
14  class StrongTaylorIto2p0(Function):
15    """
16    Strong Taylor-Ito scheme with convergence order 2.0
17    """
18    nargs = 8
19
20    def __new__(cls, *args, **kwargs):
21      """
22      Creates new StrongTaylorIto2p0 object with given args
23      Parameters
24      ==========
25      i : int
26        component of stochastic process
27      yp : numpy.ndarray
28        initial conditions
29      a : numpy.ndarray
30        algebraic, given in the variables x and t
31      b : numpy.ndarray
32        algebraic, given in the variables x and t
33      dt : float
34        integration step
35      ksi : numpy.ndarray
36        matrix of Gaussian random variables
37      q : tuple
38        amounts of q for stochastic integrals approximations
39      Returns
40      =======
41      sympy.Expr
42        formula to simplify and substitute
43      """
44      i, yp, a, b, dt, ksi, dxs, q = sympify(args)
45      n, m = b.shape[0], b.shape[1]
```

```
46
47      return Add(
48
49        yp[i, 0], a[i, 0] * dt,
50
51        *[b[i, i1] * I0(i1, dt, ksi)
52         for i1 in range(m)],
53
54        *[G(b[:, i1], b[i, i2], dxs) *
55         I00(i1, i2, q[0], dt, ksi)
56         for i2 in range(m)
57         for i1 in range(m)],
58
59        *[G(b[:, i1], a[i, 0], dxs) *
60         (dt * I0(i1, dt, ksi) + I1(i1, dt, ksi)) -
61         L(a, b, b[i, i1], dxs) *
62         I1(i1, dt, ksi)
63         for i1 in range(m)],
64
65        *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
66         I000(i1, i2, i3, q[1], dt, ksi)
67         for i3 in range(m)
68         for i2 in range(m)
69         for i1 in range(m)],
70
71        dt ** 2 / 2 * L(a, b, a[i, 0], dxs),
72
73        *[G(b[:, i1], L(a, b, b[i, i2], dxs), dxs) *
74         (I10(i1, i2, q[2], dt, ksi) - I01(i1, i2, q[2], dt, ksi)) -
75         L(a, b, G(b[:, i1], b[i, i2], dxs), dxs) * I10(i1, i2, q[2], dt, ksi) +
76         G(b[:, i1], G(b[:, i2], a[i, 0], dxs), dxs) *
77         (I01(i1, i2, q[2], dt, ksi) + dt * I00(i1, i2, q[0], dt, ksi))
78         for i2 in range(m)
79         for i1 in range(m)],
80
81        *[G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs) *
82         I0000(i1, i2, i3, i4, q[3], dt, ksi)
83         for i4 in range(m)
84         for i3 in range(m)
85         for i2 in range(m)
86         for i1 in range(m)]
87
88      )
89
90    def doit(self, **hints):
91      """
92      Tries to expand or calculate function
93      Returns
94      =======
95      sympy.Expr
96      """
97      return StrongTaylorIto2p0(*self.args, **hints)
```

Listing 118: **Strong Taylor–Itô scheme with convergence order** 2.5 **modeling subprogram**

```
import logging
from time import time

import numpy as np
from sympy import symbols, Matrix, MatrixSymbol, lambdify

from mathematics.sde.nonlinear.q import get_q
from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_ito_2p5 import
    StrongTaylorIto2p5


def strong_taylor_ito_2p5(y0: np.array, a: Matrix, b: Matrix, k: float, times: tuple):
  """
  Performs modeling of Strong Taylor-Ito scheme with convergence order 2.5
  Parameters
  ==========
  y0 : numpy.ndarray
    initial conditions
  a : numpy.ndarray
    vector function a
  b : numpy.ndarray
    matrix function b
  k : float
    precision constant
  times : tuple
    integration limits and step
  Returns
  =======
  y : numpy.ndarray
    vector of solution
  t : list
    list of time moments
  """
  start_time = time()

  logger = logging.getLogger(__name__)

  logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 2.5 start")

  # Ranges
  n = b.shape[0]
  m = b.shape[1]
  t1 = times[0]
  dt = times[1]
  t2 = times[2]

  # Defining context
  args = symbols(f"x1:{n + 1}")
  ticks = int((t2 - t1) / dt)
  q = get_q(dt, k, 2.5)
  logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
```

```
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 3, m)
57    sym_y = StrongTaylorIto2p5(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 2.5 subs are
        finished")
69
70    # Substitution values
71    t = [t1 + i * dt for i in range(ticks)]
72    y = np.zeros((n, ticks))
73    y[:, 0] = y0[:, 0]
74
75    # Dynamic substitutions with integration
76    for p in range(ticks - 1):
77      values = [*y[:, p], t[p], np.random.randn(q[0] + 3, m)]
78      for tr in range(n):
79        y[tr, p + 1] = y_compiled[tr](*values)
80
81    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 2.5 calculations
        are finished")
82
83    return y, t
```

Listing 119: **Strong Taylor–Itô scheme with convergence order** 2.5

```
1  from sympy import Function, sympify, Add
2
3  from mathematics.sde.nonlinear.symbolic.g import G
4  from mathematics.sde.nonlinear.symbolic.ito.i0 import I0
5  from mathematics.sde.nonlinear.symbolic.ito.i00 import I00
6  from mathematics.sde.nonlinear.symbolic.ito.i000 import I000
7  from mathematics.sde.nonlinear.symbolic.ito.i0000 import I0000
8  from mathematics.sde.nonlinear.symbolic.ito.i00000 import I00000
9  from mathematics.sde.nonlinear.symbolic.ito.i001 import I001
10 from mathematics.sde.nonlinear.symbolic.ito.i01 import I01
11 from mathematics.sde.nonlinear.symbolic.ito.i010 import I010
12 from mathematics.sde.nonlinear.symbolic.ito.i1 import I1
13 from mathematics.sde.nonlinear.symbolic.ito.i10 import I10
14 from mathematics.sde.nonlinear.symbolic.ito.i100 import I100
15 from mathematics.sde.nonlinear.symbolic.ito.i2 import I2
```

```
16 from mathematics.sde.nonlinear.symbolic.l import L
17
18
19 class StrongTaylorIto2p5(Function):
20   """
21   Strong Taylor-Ito scheme with convergence order 2.5
22   """
23   nargs = 8
24
25   def __new__(cls, *args, **kwargs):
26     """
27     Creates new StrongTaylorIto2p5 object with given args
28     Parameters
29     ==========
30     i : int
31       component of stochastic process
32     yp : numpy.ndarray
33       initial conditions
34     a : numpy.ndarray
35       algebraic, given in the variables x and t
36     b : numpy.ndarray
37       algebraic, given in the variables x and t
38     dt : float
39       integration step
40     ksi : numpy.ndarray
41       matrix of Gaussian random variables
42     q : tuple
43       amounts of q for stochastic integrals approximations
44     Returns
45     =======
46     sympy.Expr
47       formula to simplify and substitute
48     """
49     i, yp, a, b, dt, ksi, dxs, q = sympify(args)
50     n, m = b.shape[0], b.shape[1]
51
52     return Add(
53
54       yp[i, 0], a[i, 0] * dt,
55
56       *[b[i, i1] * I0(i1, dt, ksi)
57        for i1 in range(m)],
58
59       *[G(b[:, i1], b[i, i2], dxs) *
60        I00(i1, i2, q[0], dt, ksi)
61        for i2 in range(m)
62        for i1 in range(m)],
63
64       *[G(b[:, i1], a[i, 0], dxs) *
65        (dt * I0(i1, dt, ksi) + I1(i1, dt, ksi)) -
66        L(a, b, b[i, i1], dxs) *
67        I1(i1, dt, ksi)
68        for i1 in range(m)],
69
70       *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
```

```
        I000(i1, i2, i3, q[1], dt, ksi)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       dt ** 2 / 2 * L(a, b, a[i, 0], dxs),
       *[G(b[:, i1], L(a, b, b[i, i2], dxs), dxs) *
        (I10(i1, i2, q[2], dt, ksi) - I01(i1, i2, q[2], dt, ksi)) -
        L(a, b, G(b[:, i1], b[i, i2], dxs), dxs) * I10(i1, i2, q[2], dt, ksi) +
        G(b[:, i1], G(b[:, i2], a[i, 0], dxs), dxs) *
        (I01(i1, i2, q[2], dt, ksi) + dt * I00(i1, i2, q[0], dt, ksi))
        for i2 in range(m)
        for i1 in range(m)],

       *[G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs) *
        I0000(i1, i2, i3, i4, q[3], dt, ksi)
        for i4 in range(m)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       *[G(b[:, i1], L(a, b, a[i, 0], dxs), dxs) *
        (I2(i1, dt, ksi) / 2 + dt * I1(i1, dt, ksi) + dt ** 2 / 2 * I0(i1, dt, ksi)) +
        L(a, b, L(a, b, b[i, i1], dxs), dxs) * I2(i1, dt, ksi) / 2 -
        L(a, b, G(b[:, i1], a[i, 0], dxs), dxs) * (I2(i1, dt, ksi) + dt * I1(i1, dt, ksi))
        for i1 in range(m)],

       *[G(b[:, i1], L(a, b, G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
        (I100(i1, i2, i3, q[6], dt, ksi) - I010(i1, i2, i3, q[5], dt, ksi)) +
        G(b[:, i1], G(b[:, i2], L(a, b, b[i, i3], dxs), dxs), dxs) *
        (I010(i1, i2, i3, q[5], dt, ksi) - I001(i1, i2, i3, q[4], dt, ksi)) +
        G(b[:, i1], G(b[:, i2], G(b[:, i3], a[i, 0], dxs), dxs), dxs) *
        (dt * I000(i1, i2, i3, q[1], dt, ksi) + I001(i1, i2, i3, q[4], dt, ksi)) -
        L(a, b, G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
        I100(i1, i2, i3, q[6], dt, ksi)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(
        b[:, i4], b[i, i5], dxs), dxs), dxs), dxs) *
        I00000(i1, i2, i3, i4, i5, q[7], dt, ksi)
        for i5 in range(m)
        for i4 in range(m)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       dt ** 3 / 6 * L(a, b, L(a, b, a[i, 0], dxs), dxs)

    )

  def doit(self, **hints):
    """
    Tries to expand or calculate function
```

```
126         Returns
127         =======
128         sympy.Expr
129         """
130         return StrongTaylorIto2p5(*self.args, **hints)
```

Listing 120: **Strong Taylor–Itô scheme with convergence order** 3.0 **modeling subprogram**

```
1   import logging
2   from time import time
3
4   import numpy as np
5   from sympy import Matrix, symbols, MatrixSymbol, lambdify
6
7   from mathematics.sde.nonlinear.q import get_q
8   from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_ito_3p0 import
        StrongTaylorIto3p0
9
10
11  def strong_taylor_ito_3p0(y0: np.array, a: Matrix, b: Matrix, k: float, times: tuple):
12      """
13      Performs modeling of Strong Taylor-Ito scheme with convergence order 3.0
14      Parameters
15      ==========
16      y0 : numpy.ndarray
17          initial conditions
18      a : numpy.ndarray
19          vector function a
20      b : numpy.ndarray
21          matrix function b
22      k : float
23          precision constant
24      times : tuple
25          integration limits and step
26      Returns
27      =======
28      y : numpy.ndarray
29          vector of solution
30      t : list
31          list of time moments
32      """
33      start_time = time()
34
35      logger = logging.getLogger(__name__)
36
37      logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 3.0 start")
38
39      # Ranges
40      n = b.shape[0]
41      m = b.shape[1]
42      t1 = times[0]
43      dt = times[1]
```

```
44    t2 = times[2]
45
46    # Defining context
47    args = symbols(f"x1:{n + 1}")
48    ticks = int((t2 - t1) / dt)
49    q = get_q(dt, k, 3)
50    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 3, m)
57    sym_y = StrongTaylorIto3p0(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 3.0 subs are
        finished")
69
70    # Substitution values
71    t = [t1 + i * dt for i in range(ticks)]
72    y = np.zeros((n, ticks))
73    y[:, 0] = y0[:, 0]
74
75    # Dynamic substitutions with integration
76    for p in range(ticks - 1):
77      values = [*y[:, p], t[p], np.random.randn(q[0] + 3, m)]
78      for tr in range(n):
79        y[tr, p + 1] = y_compiled[tr](*values)
80
81    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Ito 3.0 calculations
        are finished")
82
83    return y, t
```

Listing 121: **Strong Taylor–Itô scheme with convergence order** 3.0

```
1  from sympy import Function, sympify, Add
2
3  from mathematics.sde.nonlinear.symbolic.g import G
4  from mathematics.sde.nonlinear.symbolic.ito.i0 import I0
5  from mathematics.sde.nonlinear.symbolic.ito.i00 import I00
6  from mathematics.sde.nonlinear.symbolic.ito.i000 import I000
7  from mathematics.sde.nonlinear.symbolic.ito.i0000 import I0000
8  from mathematics.sde.nonlinear.symbolic.ito.i00000 import I00000
```

```
9  from mathematics.sde.nonlinear.symbolic.ito.i000000 import I000000
10 from mathematics.sde.nonlinear.symbolic.ito.i0001 import I0001
11 from mathematics.sde.nonlinear.symbolic.ito.i001 import I001
12 from mathematics.sde.nonlinear.symbolic.ito.i0010 import I0010
13 from mathematics.sde.nonlinear.symbolic.ito.i01 import I01
14 from mathematics.sde.nonlinear.symbolic.ito.i010 import I010
15 from mathematics.sde.nonlinear.symbolic.ito.i0100 import I0100
16 from mathematics.sde.nonlinear.symbolic.ito.i02 import I02
17 from mathematics.sde.nonlinear.symbolic.ito.i1 import I1
18 from mathematics.sde.nonlinear.symbolic.ito.i10 import I10
19 from mathematics.sde.nonlinear.symbolic.ito.i100 import I100
20 from mathematics.sde.nonlinear.symbolic.ito.i1000 import I1000
21 from mathematics.sde.nonlinear.symbolic.ito.i11 import I11
22 from mathematics.sde.nonlinear.symbolic.ito.i2 import I2
23 from mathematics.sde.nonlinear.symbolic.ito.i20 import I20
24 from mathematics.sde.nonlinear.symbolic.l import L
25
26
27 class StrongTaylorIto3p0(Function):
28   """
29   Strong Taylor-Ito scheme with convergence order 3.0
30   """
31   nargs = 8
32
33   def __new__(cls, *args, **kwargs):
34     """
35     Creates new StrongTaylorIto3p0 object with given args
36     Parameters
37     ==========
38     i : int
39       component of stochastic process
40     yp : numpy.ndarray
41       initial conditions
42     a : numpy.ndarray
43       algebraic, given in the variables x and t
44     b : numpy.ndarray
45       algebraic, given in the variables x and t
46     dt : float
47       integration step
48     ksi : numpy.ndarray
49       matrix of Gaussian random variables
50     q : tuple
51       amounts of q for stochastic integrals approximations
52     Returns
53     =======
54     sympy.Expr
55       formula to simplify and substitute
56     """
57     i, yp, a, b, dt, ksi, dxs, q = sympify(args)
58     n, m = b.shape[0], b.shape[1]
59
60     return Add(
61
62       yp[i, 0], a[i, 0] * dt,
63
```

```
64        *[b[i, i1] * I0(i1, dt, ksi)
65         for i1 in range(m)],
66
67        *[G(b[:, i1], b[i, i2], dxs) *
68         I00(i1, i2, q[0], dt, ksi)
69         for i2 in range(m)
70         for i1 in range(m)],
71
72        *[G(b[:, i1], a[i, 0], dxs) *
73         (dt * I0(i1, dt, ksi) + I1(i1, dt, ksi)) -
74         L(a, b, b[i, i1], dxs) *
75         I1(i1, dt, ksi)
76         for i1 in range(m)],
77
78        *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
79         I000(i1, i2, i3, q[1], dt, ksi)
80         for i3 in range(m)
81         for i2 in range(m)
82         for i1 in range(m)],
83
84        dt ** 2 / 2 * L(a, b, a[i, 0], dxs),
85
86        *[G(b[:, i1], L(a, b, b[i, i2], dxs), dxs) *
87         (I10(i1, i2, q[2], dt, ksi) - I01(i1, i2, q[2], dt, ksi)) -
88         L(a, b, G(b[:, i1], b[i, i2], dxs), dxs) * I10(i1, i2, q[2], dt, ksi) +
89         G(b[:, i1], G(b[:, i2], a[i, 0], dxs), dxs) *
90         (I01(i1, i2, q[2], dt, ksi) + dt * I00(i1, i2, q[0], dt, ksi))
91         for i2 in range(m)
92         for i1 in range(m)],
93
94        *[G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs) *
95         I0000(i1, i2, i3, i4, q[3], dt, ksi)
96         for i4 in range(m)
97         for i3 in range(m)
98         for i2 in range(m)
99         for i1 in range(m)],
100
101       *[G(b[:, i1], L(a, b, a[i, 0], dxs), dxs) *
102        (I2(i1, dt, ksi) / 2 + dt * I1(i1, dt, ksi) + dt ** 2 / 2 * I0(i1, dt, ksi)) +
103        L(a, b, L(a, b, b[i, i1], dxs), dxs) * I2(i1, dt, ksi) / 2 -
104        L(a, b, G(b[:, i1], a[i, 0], dxs), dxs) * (I2(i1, dt, ksi) + dt * I1(i1, dt, ksi))
105        for i1 in range(m)],
106
107       *[G(b[:, i1], L(a, b, G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
108        (I100(i1, i2, i3, q[6], dt, ksi) - I010(i1, i2, i3, q[5], dt, ksi)) +
109        G(b[:, i1], G(b[:, i2], L(a, b, b[i, i3], dxs), dxs), dxs) *
110        (I010(i1, i2, i3, q[5], dt, ksi) - I001(i1, i2, i3, q[4], dt, ksi)) +
111        G(b[:, i1], G(b[:, i2], G(b[:, i3], a[i, 0], dxs), dxs), dxs) *
112        (dt * I000(i1, i2, i3, q[1], dt, ksi) + I001(i1, i2, i3, q[4], dt, ksi)) -
113        L(a, b, G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
114        I100(i1, i2, i3, q[6], dt, ksi)
115        for i3 in range(m)
116        for i2 in range(m)
117        for i1 in range(m)],
118
```

```
        *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(
         b[:, i4], b[i, i5], dxs), dxs), dxs), dxs) *
         I00000(i1, i2, i3, i4, i5, q[7], dt, ksi)
         for i5 in range(m)
         for i4 in range(m)
         for i3 in range(m)
         for i2 in range(m)
         for i1 in range(m)],

        dt ** 3 / 6 * L(a, b, L(a, b, a[i, 0], dxs), dxs),

        *[G(b[:, i1], G(b[:, i2], L(a, b, a[i, 0], dxs), dxs), dxs) *
         (I02(i1, i2, q[6], dt, ksi) / 2 + dt * I01(i1, i2, q[2], dt, ksi) +
          dt ** 2 / 2 * I00(i1, i2, q[2], dt, ksi)) +
         L(a, b, L(a, b, G(b[:, i1], b[i, i2], dxs), dxs), dxs) / 2 *
         I20(i1, i2, q[10], dt, ksi) +
         G(b[:, i1], L(a, b, G(b[:, i2], a[i, 0], dxs), dxs), dxs) *
         (I11(i1, i2, q[9], dt, ksi) - I02(i1, i2, q[8], dt, ksi) +
          dt * (I10(i1, i2, q[2], dt, ksi) - I01(i1, i2, q[2], dt, ksi))) +
         L(a, b, G(b[:, i1], L(a, b, b[i, i2], dxs), dxs), dxs) *
         (I11(i1, i2, q[9], dt, ksi) - I20(i1, i2, q[10], dt, ksi)) +
         G(b[:, i1], L(a, b, L(a, b, b[i, i2], dxs), dxs), dxs) *
         (I02(i1, i2, q[8], dt, ksi) / 2 + I20(i1, i2, q[10], dt, ksi) / 2 -
          I11(i1, i2, q[9], dt, ksi)) -
         L(a, b, G(b[:, i1], G(b[:, i2], a[i, 0], dxs), dxs), dxs) *
         (dt * I10(i1, i2, q[2], dt, ksi) + I11(i1, i2, q[9], dt, ksi))
         for i2 in range(m)
         for i1 in range(m)],

        *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(b[:, i4], a[i, 0], dxs), dxs), dxs), dxs) *
         (dt * I0000(i1, i2, i3, i4, q[3], dt, ksi) +
         I0001(i1, i2, i3, i4, q[11], dt, ksi)) +
         G(b[:, i1], G(b[:, i2], L(a, b, G(b[:, i3], b[i, i4], dxs), dxs), dxs), dxs) *
         (I0100(i1, i2, i3, i4, q[13], dt, ksi) - I0010(i1, i2, i3, i4, q[12], dt, ksi)) -
         L(a, b, G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs), dxs) *
         I1000(i1, i2, i3, i4, q[14], dt, ksi) +
         G(b[:, i1], L(a, b, G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs), dxs) *
         (I1000(i1, i2, i3, i4, q[14], dt, ksi) - I0100(i1, i2, i3, i4, q[13], dt, ksi)) +
         G(b[:, i1], G(b[:, i2], G(b[:, i3], L(a, b, b[i, i4], dxs), dxs), dxs), dxs) *
         (I0010(i1, i2, i3, i4, q[12], dt, ksi) - I0001(i1, i2, i3, i4, q[11], dt, ksi))
         for i4 in range(m)
         for i3 in range(m)
         for i2 in range(m)
         for i1 in range(m)],

        *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(b[:, i4], G(
         b[:, i5], b[i, i6], dxs), dxs), dxs), dxs), dxs) *
         I000000(i1, i2, i3, i4, i5, i6, q[15], dt, ksi)
         for i6 in range(m)
         for i5 in range(m)
         for i4 in range(m)
         for i3 in range(m)
         for i2 in range(m)
         for i1 in range(m)]
```

```
174         )
175
176     def doit(self, **hints):
177         """
178         Tries to expand or calculate function
179         Returns
180         =======
181         sympy.Expr
182         """
183         return StrongTaylorIto3p0(*self.args, **hints)
```

6.2.7 Source Codes for Strong Taylor–Stratonovich Numerical Schemes with Convergence Orders 1.0, 1.5, 2.0, 2.5, and 3.0 for Itô SDEs

Listing 122: **Strong Taylor–Stratonovich scheme with convergence order** 1.0 **modeling subprogram**

```
1  import logging
2  from time import time
3
4  import numpy as np
5  from sympy import Matrix, symbols, MatrixSymbol, lambdify
6
7  from mathematics.sde.nonlinear.q import get_q
8  from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_stratonovich_1p0 import
       StrongTaylorStratonovich1p0
9
10
11 def strong_taylor_stratonovich_1p0(y0: np.array, a: Matrix, b: Matrix, k: float, times:
       tuple):
12     """
13     Performs modeling of Strong Taylor-Stratonovich scheme with convergence order 1.0
14     Parameters
15     ==========
16     y0 : numpy.ndarray
17         initial conditions
18     a : numpy.ndarray
19         vector function a
20     b : numpy.ndarray
21         matrix function b
22     k : float
23         precision constant
24     times : tuple
25         integration limits and step
26     Returns
27     =======
28     y : numpy.ndarray
29         vector of solution
```

```
    t : list
      list of time moments
    """
    start_time = time()

    logger = logging.getLogger(__name__)

    logger.info(f"[{(time() - start_time):.3f} seconds] Taylor-Stratonovich 1.0 start")

    # Ranges
    n = b.shape[0]
    m = b.shape[1]
    t1 = times[0]
    dt = times[1]
    t2 = times[2]

    # Defining context
    args = symbols(f"x1:{n + 1}")
    ticks = int((t2 - t1) / dt)
    q = get_q(dt, k, 1)
    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")

    # Symbols
    sym_i, sym_t = symbols("i t")
    sym_ksi = MatrixSymbol("ksi", q[0] + 2, m)
    sym_y = StrongTaylorStratonovich1p0(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)

    args_extended = list()
    args_extended.extend(args)
    args_extended.extend([sym_t, sym_ksi])

    # Compilation of formulas
    y_compiled = list()
    for tr in range(n):
      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))

    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
          f"Taylor-Stratonovich 1.0 subs are finished")

    # Substitution values
    t = [t1 + i * dt for i in range(ticks)]
    y = np.zeros((n, ticks))
    y[:, 0] = y0[:, 0]

    # Dynamic substitutions with integration
    for p in range(ticks - 1):
      values = [*y[:, p], t[p], np.random.randn(q[0] + 2, m)]
      for tr in range(n):
        y[tr, p + 1] = y_compiled[tr](*values)

    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
          f"Taylor-Stratonovich 1.0 calculations are finished")

```

```
85    return y, t
```

Listing 123: **Strong Taylor–Stratonovich scheme with convergence order** 1.0

```
1   from sympy import Function, sympify, Add
2
3   from mathematics.sde.nonlinear.symbolic.aj import Aj
4   from mathematics.sde.nonlinear.symbolic.g import G
5   from mathematics.sde.nonlinear.symbolic.stratonovich.j0 import J0
6   from mathematics.sde.nonlinear.symbolic.stratonovich.j00 import J00
7
8
9   class StrongTaylorStratonovich1p0(Function):
10    """
11    Strong Taylor-Stratonovich scheme with convergence order 1.0
12    """
13    nargs = 8
14
15    def __new__(cls, *args, **kwargs):
16      """
17      Creates new StrongTaylorStratonovich1p0 object with given args
18      Parameters
19      ==========
20      i : int
21        component of stochastic process
22      yp : numpy.ndarray
23        initial conditions
24      a : numpy.ndarray
25        algebraic, given in the variables x and t
26      b : numpy.ndarray
27        algebraic, given in the variables x and t
28      dt : float
29        integration step
30      ksi : numpy.ndarray
31        matrix of Gaussian random variables
32      q : tuple
33        amounts of q for stochastic integrals approximations
34      Returns
35      =======
36      sympy.Expr
37        formula to simplify and substitute
38      """
39      i, yp, a, b, dt, ksi, dxs, q = sympify(args)
40      n, m = b.shape[0], b.shape[1]
41
42      aj = Aj(i, a, b, dxs)
43
44      return Add(
45
46        yp[i, 0], aj[i, 0] * dt,
47
48        *[b[i, i1] * J0(i1, dt, ksi)
49          for i1 in range(m)],
```

```
50
51          *[G(b[:, i1], b[i, i2], dxs) *
52           J00(i1, i2, q[0], dt, ksi)
53           for i2 in range(m)
54           for i1 in range(m)]
55
56      )
57
58    def doit(self, **hints):
59      """
60      Tries to expand or calculate function
61      Returns
62      =======
63      sympy.Expr
64      """
65      return StrongTaylorStratonovich1p0(*self.args, **hints)
```

Listing 124: **Strong Taylor–Stratonovich scheme with convergence order** 1.5 **modeling subprogram**

```
1  import logging
2  from time import time
3
4  import numpy as np
5  from sympy import Matrix, MatrixSymbol, symbols, lambdify
6
7  from mathematics.sde.nonlinear.q import get_q
8  from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_stratonovich_1p5 import
       StrongTaylorStratonovich1p5
9
10
11 def strong_taylor_stratonovich_1p5(y0: np.array, a: Matrix, b: Matrix, k: float, times:
       tuple):
12   """
13   Performs modeling of Strong Taylor-Stratonovich scheme with convergence order 1.5
14   Parameters
15   ==========
16   y0 : numpy.ndarray
17     initial conditions
18   a : numpy.ndarray
19     vector function a
20   b : numpy.ndarray
21     matrix function b
22   k : float
23     precision constant
24   times : tuple
25     integration limits and step
26   Returns
27   =======
28   y : numpy.ndarray
29     vector of solution
30   t : list
31     list of time moments
```

```
32    """
33    start_time = time()
34
35    logger = logging.getLogger(__name__)
36
37    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Stratonovich 1.5
        start")
38
39    # Ranges
40    n = b.shape[0]
41    m = b.shape[1]
42    t1 = times[0]
43    dt = times[1]
44    t2 = times[2]
45
46    # Defining context
47    args = symbols(f"x1:{n + 1}")
48    ticks = int((t2 - t1) / dt)
49    q = get_q(dt, k, 1.5)
50    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 2, m)
57    sym_y = StrongTaylorStratonovich1p5(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
69          f"Taylor-Stratonovich 1.5 subs are finished")
70
71    # Substitution values
72    t = [t1 + i * dt for i in range(ticks)]
73    y = np.zeros((n, ticks))
74    y[:, 0] = y0[:, 0]
75
76    # Dynamic substitutions with integration
77    for p in range(ticks - 1):
78      values = [*y[:, p], t[p], np.random.randn(q[0] + 2, m)]
79      for tr in range(n):
80        y[tr, p + 1] = y_compiled[tr](*values)
81
82    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
83          f"Taylor-Stratonovich 1.5 calculations are finished")
84
85    return y, t
```

Listing 125: **Strong Taylor–Stratonovich scheme with convergence order** 1.5

```
from sympy import Function, sympify, Add

from mathematics.sde.nonlinear.symbolic.aj import Aj
from mathematics.sde.nonlinear.symbolic.g import G
from mathematics.sde.nonlinear.symbolic.l import L
from mathematics.sde.nonlinear.symbolic.lj import Lj
from mathematics.sde.nonlinear.symbolic.stratonovich.j0 import J0
from mathematics.sde.nonlinear.symbolic.stratonovich.j00 import J00
from mathematics.sde.nonlinear.symbolic.stratonovich.j000 import J000
from mathematics.sde.nonlinear.symbolic.stratonovich.j1 import J1


class StrongTaylorStratonovich1p5(Function):
  """
  Strong Taylor-Stratonovich scheme with convergence order 1.5
  """
  nargs = 8

  def __new__(cls, *args, **kwargs):
    """
    Creates new StrongTaylorStratonovich1p5 object with given args
    Parameters
    ==========
    i : int
      component of stochastic process
    yp : numpy.ndarray
      initial conditions
    a : numpy.ndarray
      algebraic, given in the variables x and t
    b : numpy.ndarray
      algebraic, given in the variables x and t
    dt : float
      integration step
    ksi : numpy.ndarray
      matrix of Gaussian random variables
    q : tuple
      amounts of q for stochastic integrals approximations
    Returns
    =======
    sympy.Expr
      formula to simplify and substitute
    """
    i, yp, a, b, dt, ksi, dxs, q = sympify(args)
    n, m = b.shape[0], b.shape[1]

    aj = Aj(i, a, b, dxs)

    return Add(

      yp[i, 0], aj[i, 0] * dt,

      *[b[i, i1] * J0(i1, dt, ksi)
       for i1 in range(m)],
```

```
54
55          *[G(b[:, i1], b[i, i2], dxs) *
56           J00(i1, i2, q[0], dt, ksi)
57           for i2 in range(m)
58           for i1 in range(m)],
59
60          *[G(b[:, i1], aj[i, 0], dxs) *
61           (dt * J0(i1, dt, ksi) + J1(i1, dt, ksi)) -
62           Lj(a, b[i, i1], dxs) *
63           J1(i1, dt, ksi)
64           for i1 in range(m)],
65
66          *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
67           J000(i1, i2, i3, q[1], dt, ksi)
68           for i3 in range(m)
69           for i2 in range(m)
70           for i1 in range(m)],
71
72          dt ** 2 / 2 * L(a, b, a[i, 0], dxs)
73
74      )
75
76    def doit(self, **hints):
77      """
78      Tries to expand or calculate function
79      Returns
80      =======
81      sympy.Expr
82      """
83      return StrongTaylorStratonovich1p5(*self.args, **hints)
```

Listing 126: **Strong Taylor–Stratonovich scheme with convergence order** 2.0 **modeling subprogram**

```
1  import logging
2  from time import time
3
4  import numpy as np
5  from sympy import symbols, Matrix, MatrixSymbol, lambdify
6
7  from mathematics.sde.nonlinear.q import get_q
8  from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_stratonovich_2p0 import
       StrongTaylorStratonovich2p0
9
10
11 def strong_taylor_stratonovich_2p0(y0: np.array, a: Matrix, b: Matrix, k: float, times:
       tuple):
12   """
13   Performs modeling of Strong Taylor-Stratonovich scheme with convergence order 2.0
14   Parameters
15   ==========
16   y0 : numpy.ndarray
17     initial conditions
```

```
18    a : numpy.ndarray
19      vector function a
20    b : numpy.ndarray
21      matrix function b
22    k : float
23      precision constant
24    times : tuple
25      integration limits and step
26    Returns
27    =======
28    y : numpy.ndarray
29      vector of solution
30    t : list
31      list of time moments
32    """
33    start_time = time()
34
35    logger = logging.getLogger(__name__)
36
37    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Stratonovich 2.0
        start")
38
39    # Ranges
40    n = b.shape[0]
41    m = b.shape[1]
42    t1 = times[0]
43    dt = times[1]
44    t2 = times[2]
45
46    # Defining context
47    args = symbols(f"x1:{n + 1}")
48    ticks = int((t2 - t1) / dt)
49    q = get_q(dt, k, 2)
50    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 2, m)
57    sym_y = StrongTaylorStratonovich2p0(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
69          f"Taylor-Stratonovich 2.0 subs are finished")
70
71    # Substitution values
```

```
  t = [t1 + i * dt for i in range(ticks)]
  y = np.zeros((n, ticks))
  y[:, 0] = y0[:, 0]

  # Dynamic substitutions with integration
  for p in range(ticks - 1):
    values = [*y[:, p], t[p], np.random.randn(q[0] + 2, m)]
    for tr in range(n):
      y[tr, p + 1] = y_compiled[tr](*values)

  logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
        f"Taylor-Stratonovich 2.0 calculations are finished")

  return y, t
```

Listing 127: **Strong Taylor–Stratonovich scheme with convergence order** 2.0

```
from sympy import Function, sympify, Add

from mathematics.sde.nonlinear.symbolic.aj import Aj
from mathematics.sde.nonlinear.symbolic.g import G
from mathematics.sde.nonlinear.symbolic.lj import Lj
from mathematics.sde.nonlinear.symbolic.stratonovich.j0 import J0
from mathematics.sde.nonlinear.symbolic.stratonovich.j00 import J00
from mathematics.sde.nonlinear.symbolic.stratonovich.j000 import J000
from mathematics.sde.nonlinear.symbolic.stratonovich.j0000 import J0000
from mathematics.sde.nonlinear.symbolic.stratonovich.j01 import J01
from mathematics.sde.nonlinear.symbolic.stratonovich.j1 import J1
from mathematics.sde.nonlinear.symbolic.stratonovich.j10 import J10


class StrongTaylorStratonovich2p0(Function):
  """
  Strong Taylor-Stratonovich scheme with convergence order 2.0
  """
  nargs = 8

  def __new__(cls, *args, **kwargs):
    """
    Creates new StrongTaylorStratonovich2p0 object with given args
    Parameters
    ==========
    i : int
      component of stochastic process
    yp : numpy.ndarray
      initial conditions
    a : numpy.ndarray
      algebraic, given in the variables x and t
    b : numpy.ndarray
      algebraic, given in the variables x and t
    dt : float
      integration step
    ksi : numpy.ndarray
```

```
37          matrix of Gaussian random variables
38      q : tuple
39          amounts of q for stochastic integrals approximations
40      Returns
41      =======
42      sympy.Expr
43          formula to simplify and substitute
44      """
45      i, yp, a, b, dt, ksi, dxs, q = sympify(args)
46      n, m = b.shape[0], b.shape[1]
47
48      aj = Aj(i, a, b, dxs)
49
50      return Add(
51
52          yp[i, 0], aj[i, 0] * dt,
53
54          *[b[i, i1] * J0(i1, dt, ksi)
55            for i1 in range(m)],
56
57          *[G(b[:, i1], b[i, i2], dxs) *
58            J00(i1, i2, q[0], dt, ksi)
59            for i2 in range(m)
60            for i1 in range(m)],
61
62          *[G(b[:, i1], aj[i, 0], dxs) *
63            (dt * J0(i1, dt, ksi) + J1(i1, dt, ksi)) -
64            Lj(a, b[i, i1], dxs) *
65            J1(i1, dt, ksi)
66            for i1 in range(m)],
67
68          *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
69            J000(i1, i2, i3, q[1], dt, ksi)
70            for i3 in range(m)
71            for i2 in range(m)
72            for i1 in range(m)],
73
74              dt ** 2 / 2 * Lj(a, aj[i, 0], dxs),
75
76          *[G(b[:, i1], Lj(a, b[i, i2], dxs), dxs) *
77            (J10(i1, i2, q[2], dt, ksi) - J01(i1, i2, q[2], dt, ksi)) -
78            Lj(a, G(b[:, i1], b[i, i2], dxs), dxs) * J10(i1, i2, q[2], dt, ksi) +
79            G(b[:, i1], G(b[:, i2], aj[i, 0], dxs), dxs) *
80            (J01(i1, i2, q[2], dt, ksi) + dt * J00(i1, i2, q[0], dt, ksi))
81            for i2 in range(m)
82            for i1 in range(m)],
83
84          *[G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs) *
85            J0000(i1, i2, i3, i4, q[3], dt, ksi)
86            for i4 in range(m)
87            for i3 in range(m)
88            for i2 in range(m)
89            for i1 in range(m)]
90
91      )
```

```
92
93    def doit(self, **hints):
94        """
95        Tries to expand or calculate function
96        Returns
97        =======
98        sympy.Expr
99        """
100       return StrongTaylorStratonovich2p0(*self.args, **hints)
```

Listing 128: **Strong Taylor–Stratonovich scheme with convergence order** 2.5 **modeling subprogram**

```
1   import logging
2   from time import time
3
4   import numpy as np
5   from sympy import Matrix, symbols, MatrixSymbol, lambdify
6
7   from mathematics.sde.nonlinear.q import get_q
8   from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_stratonovich_2p5 import
        StrongTaylorStratonovich2p5
9
10
11  def strong_taylor_stratonovich_2p5(y0: np.array, a: Matrix, b: Matrix, k: float, times:
        tuple):
12      """
13      Performs modeling of Strong Taylor-Stratonovich scheme with convergence order 2.5
14      Parameters
15      ==========
16      y0 : numpy.ndarray
17          initial conditions
18      a : numpy.ndarray
19          vector function a
20      b : numpy.ndarray
21          matrix function b
22      k : float
23          precision constant
24      times : tuple
25          integration limits and step
26      Returns
27      =======
28      y : numpy.ndarray
29          vector of solution
30      t : list
31          list of time moments
32      """
33      start_time = time()
34
35      logger = logging.getLogger(__name__)
36
37      logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Stratonovich 2.5
            start")
```

```
38
39    # Ranges
40    n = b.shape[0]
41    m = b.shape[1]
42    t1 = times[0]
43    dt = times[1]
44    t2 = times[2]
45
46    # Defining context
47    args = symbols(f"x1:{n + 1}")
48    ticks = int((t2 - t1) / dt)
49    q = get_q(dt, k, 2.5)
50    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 3, m)
57    sym_y = StrongTaylorStratonovich2p5(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
69          f"Taylor-Stratonovich 2.5 subs are finished")
70
71    # Substitution values
72    t = [t1 + i * dt for i in range(ticks)]
73    y = np.zeros((n, ticks))
74    y[:, 0] = y0[:, 0]
75
76    # Dynamic substitutions with integration
77    for p in range(ticks - 1):
78      values = [*y[:, p], t[p], np.random.randn(q[0] + 3, m)]
79      for tr in range(n):
80        y[tr, p + 1] = y_compiled[tr](*values)
81
82    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
83          f"Taylor-Stratonovich 2.5 calculations are finished")
84
85    return y, t
```

Listing 129: **Strong Taylor–Stratonovich scheme with convergence order** 2.5

```
1  from sympy import Function, sympify, Add
2
```

```
3  from mathematics.sde.nonlinear.symbolic.aj import Aj
4  from mathematics.sde.nonlinear.symbolic.g import G
5  from mathematics.sde.nonlinear.symbolic.l import L
6  from mathematics.sde.nonlinear.symbolic.lj import Lj
7  from mathematics.sde.nonlinear.symbolic.stratonovich.j0 import J0
8  from mathematics.sde.nonlinear.symbolic.stratonovich.j00 import J00
9  from mathematics.sde.nonlinear.symbolic.stratonovich.j000 import J000
10 from mathematics.sde.nonlinear.symbolic.stratonovich.j0000 import J0000
11 from mathematics.sde.nonlinear.symbolic.stratonovich.j00000 import J00000
12 from mathematics.sde.nonlinear.symbolic.stratonovich.j001 import J001
13 from mathematics.sde.nonlinear.symbolic.stratonovich.j01 import J01
14 from mathematics.sde.nonlinear.symbolic.stratonovich.j010 import J010
15 from mathematics.sde.nonlinear.symbolic.stratonovich.j1 import J1
16 from mathematics.sde.nonlinear.symbolic.stratonovich.j10 import J10
17 from mathematics.sde.nonlinear.symbolic.stratonovich.j100 import J100
18 from mathematics.sde.nonlinear.symbolic.stratonovich.j2 import J2
19
20
21 class StrongTaylorStratonovich2p5(Function):
22   """
23   Strong Taylor-Stratonovich scheme with convergence order 2.5
24   """
25   nargs = 8
26
27   def __new__(cls, *args, **kwargs):
28     """
29     Creates new StrongTaylorStratonovich2p5 object with given args
30     Parameters
31     ==========
32     i : int
33       component of stochastic process
34     yp : numpy.ndarray
35       initial conditions
36     a : numpy.ndarray
37       algebraic, given in the variables x and t
38     b : numpy.ndarray
39       algebraic, given in the variables x and t
40     dt : float
41       integration step
42     ksi : numpy.ndarray
43       matrix of Gaussian random variables
44     q : tuple
45       amounts of q for stochastic integrals approximations
46     Returns
47     =======
48     sympy.Expr
49       formula to simplify and substitute
50     """
51     i, yp, a, b, dt, ksi, dxs, q = sympify(args)
52     n, m = b.shape[0], b.shape[1]
53
54     aj = Aj(i, a, b, dxs)
55
56     return Add(
57
```

```
        yp[i, 0], aj[i, 0] * dt,

        *[b[i, i1] * J0(i1, dt, ksi)
         for i1 in range(m)],

        *[G(b[:, i1], b[i, i2], dxs) *
         J00(i1, i2, q[0], dt, ksi)
         for i2 in range(m)
         for i1 in range(m)],

        *[G(b[:, i1], aj[i, 0], dxs) *
         (dt * J0(i1, dt, ksi) + J1(i1, dt, ksi)) -
         Lj(a, b[i, i1], dxs) *
         J1(i1, dt, ksi)
         for i1 in range(m)],

        *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
         J000(i1, i2, i3, q[1], dt, ksi)
         for i3 in range(m)
         for i2 in range(m)
         for i1 in range(m)],

        dt ** 2 / 2 * Lj(a, aj[i, 0], dxs),

        *[G(b[:, i1], Lj(a, b[i, i2], dxs), dxs) *
         (J10(i1, i2, q[2], dt, ksi) - J01(i1, i2, q[2], dt, ksi)) -
         Lj(a, G(b[:, i1], b[i, i2], dxs), dxs) * J10(i1, i2, q[2], dt, ksi) +
         G(b[:, i1], G(b[:, i2], aj[i, 0], dxs), dxs) *
         (J01(i1, i2, q[2], dt, ksi) + dt * J00(i1, i2, q[0], dt, ksi))
         for i2 in range(m)
         for i1 in range(m)],

        *[G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs) *
         J0000(i1, i2, i3, i4, q[3], dt, ksi)
         for i4 in range(m)
         for i3 in range(m)
         for i2 in range(m)
         for i1 in range(m)],

        *[G(b[:, i1], Lj(a, aj[i, 0], dxs), dxs) *
         (J2(i1, dt, ksi) / 2 + dt * J1(i1, dt, ksi) + dt ** 2 / 2 * J0(i1, dt, ksi)) +
         Lj(a, Lj(a, b[i, i1], dxs), dxs) * J2(i1, dt, ksi) / 2 -
         Lj(a, G(b[:, i1], aj[i, 0], dxs), dxs) * (J2(i1, dt, ksi) + dt * J1(i1, dt, ksi))
         for i1 in range(m)],

        *[G(b[:, i1], Lj(a, G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
         (J100(i1, i2, i3, q[6], dt, ksi) - J010(i1, i2, i3, q[5], dt, ksi)) +
         G(b[:, i1], G(b[:, i2], Lj(a, b[i, i3], dxs), dxs), dxs) *
         (J010(i1, i2, i3, q[5], dt, ksi) - J001(i1, i2, i3, q[4], dt, ksi)) +
         G(b[:, i1], G(b[:, i2], G(b[:, i3], aj[i, 0], dxs), dxs), dxs) *
         (dt * J000(i1, i2, i3, q[1], dt, ksi) + J001(i1, i2, i3, q[4], dt, ksi)) -
         Lj(a, G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
         J100(i1, i2, i3, q[6], dt, ksi)
         for i3 in range(m)
         for i2 in range(m)
```

```
        for i1 in range(m)],

       *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(
        b[:, i4], b[i, i5], dxs), dxs), dxs), dxs) *
        J00000(i1, i2, i3, i4, i5, q[7], dt, ksi)
        for i5 in range(m)
        for i4 in range(m)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       dt ** 3 / 6 * L(a, b, L(a, b, a[i, 0], dxs), dxs)

    )

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    sympy.Expr
    """
    return StrongTaylorStratonovich2p5(*self.args, **hints)
```

Listing 130: **Strong Taylor–Stratonovich scheme with convergence order** 3.0 **modeling subprogram**

```
import logging
from time import time

import numpy as np
from sympy import Matrix, symbols, MatrixSymbol, lambdify

from mathematics.sde.nonlinear.q import get_q
from mathematics.sde.nonlinear.symbolic.schemes.strong_taylor_stratonovich_3p0 import
    StrongTaylorStratonovich3p0


def strong_taylor_stratonovich_3p0(y0: np.array, a: Matrix, b: Matrix, k: float, times:
    tuple):
  """
  Performs modeling of Strong Taylor–Stratonovich scheme with convergence order 3.0
  Parameters
  ==========
  y0 : numpy.ndarray
    initial conditions
  a : numpy.ndarray
    vector function a
  b : numpy.ndarray
    matrix function b
  k : float
    precision constant
  times : tuple
```

```
25      integration limits and step
26    Returns
27    =======
28    y : numpy.ndarray
29      vector of solution
30    t : list
31      list of time moments
32    """
33    start_time = time()
34
35    logger = logging.getLogger(__name__)
36
37    logger.info(f"[{(time() - start_time):.3f} seconds] Strong Taylor-Stratonovich 3.0
        start")
38
39    # Ranges
40    n = b.shape[0]
41    m = b.shape[1]
42    t1 = times[0]
43    dt = times[1]
44    t2 = times[2]
45
46    # Defining context
47    args = symbols(f"x1:{n + 1}")
48    ticks = int((t2 - t1) / dt)
49    q = get_q(dt, k, 3)
50    logger.info(f"[{(time() - start_time):.3f} seconds] Using C = {k}")
51    logger.info(f"[{(time() - start_time):.3f} seconds] Using dt = {dt}")
52    logger.info(f"[{(time() - start_time):.3f} seconds] Using q = {q}")
53
54    # Symbols
55    sym_i, sym_t = symbols("i t")
56    sym_ksi = MatrixSymbol("ksi", q[0] + 3, m)
57    sym_y = StrongTaylorStratonovich3p0(sym_i, Matrix(args), a, b, dt, sym_ksi, args, q)
58
59    args_extended = list()
60    args_extended.extend(args)
61    args_extended.extend([sym_t, sym_ksi])
62
63    # Compilation of formulas
64    y_compiled = list()
65    for tr in range(n):
66      y_compiled.append(lambdify(args_extended, sym_y.subs(sym_i, tr), "numpy"))
67
68    logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
69          f"Taylor-Stratonovich 3.0 subs are finished")
70
71    # Substitution values
72    t = [t1 + i * dt for i in range(ticks)]
73    y = np.zeros((n, ticks))
74    y[:, 0] = y0[:, 0]
75
76    # Dynamic substitutions with integration
77    for p in range(ticks - 1):
78      values = [*y[:, p], t[p], np.random.randn(q[0] + 3, m)]
```

```
    for tr in range(n):
      y[tr, p + 1] = y_compiled[tr](*values)

  logger.info(f"[{(time() - start_time):.3f} seconds] Strong "
        f"Taylor–Stratonovich 3.0 calculations are finished")

  return y, t
```

Listing 131: **Strong Taylor–Stratonovich scheme with convergence order** 3.0

```
from sympy import Function, sympify, Add

from mathematics.sde.nonlinear.symbolic.aj import Aj
from mathematics.sde.nonlinear.symbolic.g import G
from mathematics.sde.nonlinear.symbolic.lj import Lj
from mathematics.sde.nonlinear.symbolic.stratonovich.j0 import J0
from mathematics.sde.nonlinear.symbolic.stratonovich.j00 import J00
from mathematics.sde.nonlinear.symbolic.stratonovich.j000 import J000
from mathematics.sde.nonlinear.symbolic.stratonovich.j0000 import J0000
from mathematics.sde.nonlinear.symbolic.stratonovich.j00000 import J00000
from mathematics.sde.nonlinear.symbolic.stratonovich.j000000 import J000000
from mathematics.sde.nonlinear.symbolic.stratonovich.j0001 import J0001
from mathematics.sde.nonlinear.symbolic.stratonovich.j001 import J001
from mathematics.sde.nonlinear.symbolic.stratonovich.j0010 import J0010
from mathematics.sde.nonlinear.symbolic.stratonovich.j01 import J01
from mathematics.sde.nonlinear.symbolic.stratonovich.j010 import J010
from mathematics.sde.nonlinear.symbolic.stratonovich.j0100 import J0100
from mathematics.sde.nonlinear.symbolic.stratonovich.j02 import J02
from mathematics.sde.nonlinear.symbolic.stratonovich.j1 import J1
from mathematics.sde.nonlinear.symbolic.stratonovich.j10 import J10
from mathematics.sde.nonlinear.symbolic.stratonovich.j100 import J100
from mathematics.sde.nonlinear.symbolic.stratonovich.j1000 import J1000
from mathematics.sde.nonlinear.symbolic.stratonovich.j11 import J11
from mathematics.sde.nonlinear.symbolic.stratonovich.j2 import J2
from mathematics.sde.nonlinear.symbolic.stratonovich.j20 import J20


class StrongTaylorStratonovich3p0(Function):
  """
  Strong Taylor–Stratonovich scheme with convergence order 3.0
  """
  nargs = 8

  def __new__(cls, *args, **kwargs):
    """
    Creates new StrongTaylorStratonovich3p0 object with given args
    Parameters
    ==========
    i : int
      component of stochastic process
    yp : numpy.ndarray
      initial conditions
    a : numpy.ndarray
```

```
        algebraic, given in the variables x and t
    b : numpy.ndarray
        algebraic, given in the variables x and t
    dt : float
        integration step
    ksi : numpy.ndarray
        matrix of Gaussian random variables
    q : tuple
        amounts of q for stochastic integrals approximations
    Returns
    =======
    sympy.Expr
        formula to simplify and substitute
    """
    i, yp, a, b, dt, ksi, dxs, q = sympify(args)
    n, m = b.shape[0], b.shape[1]

    aj = Aj(i, a, b, dxs)

    return Add(

      yp[i, 0], aj[i, 0] * dt,

      *[b[i, i1] * J0(i1, dt, ksi)
        for i1 in range(m)],

      *[G(b[:, i1], b[i, i2], dxs) *
        J00(i1, i2, q[0], dt, ksi)
        for i2 in range(m)
        for i1 in range(m)],

      *[G(b[:, i1], aj[i, 0], dxs) *
        (dt * J0(i1, dt, ksi) + J1(i1, dt, ksi)) -
        Lj(a, b[i, i1], dxs) *
        J1(i1, dt, ksi)
        for i1 in range(m)],

      *[G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs) *
        J000(i1, i2, i3, q[1], dt, ksi)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

      dt ** 2 / 2 * Lj(a, aj[i, 0], dxs),

      *[G(b[:, i1], Lj(a, b[i, i2], dxs), dxs) *
        (J10(i1, i2, q[2], dt, ksi) - J01(i1, i2, q[2], dt, ksi)) -
        Lj(a, G(b[:, i1], b[i, i2], dxs), dxs) * J10(i1, i2, q[2], dt, ksi) +
        G(b[:, i1], G(b[:, i2], aj[i, 0], dxs), dxs) *
        (J01(i1, i2, q[2], dt, ksi) + dt * J00(i1, i2, q[0], dt, ksi))
        for i2 in range(m)
        for i1 in range(m)],

      *[G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs) *
        J0000(i1, i2, i3, i4, q[3], dt, ksi)
```

```
        for i4 in range(m)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       *[G(b[:, i1], Lj(a, aj[i, 0], dxs), dxs) *
        (J2(i1, dt, ksi) / 2 + dt * J1(i1, dt, ksi) + dt ** 2 / 2 * J0(i1, dt, ksi)) +
        Lj(a, Lj(a, b[i, i1], dxs), dxs) * J2(i1, dt, ksi) / 2 -
        Lj(a, G(b[:, i1], aj[i, 0], dxs), dxs) * (J2(i1, dt, ksi) + dt * J1(i1, dt, ksi))
        for i1 in range(m)],

       *[G(b[:, i1], Lj(a, G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
        (J100(i1, i2, i3, q[6], dt, ksi) - J010(i1, i2, i3, q[5], dt, ksi)) +
        G(b[:, i1], G(b[:, i2], Lj(a, b[i, i3], dxs), dxs), dxs) *
        (J010(i1, i2, i3, q[5], dt, ksi) - J001(i1, i2, i3, q[4], dt, ksi)) +
        G(b[:, i1], G(b[:, i2], G(b[:, i3], aj[i, 0], dxs), dxs), dxs) *
        (dt * J000(i1, i2, i3, q[1], dt, ksi) + J001(i1, i2, i3, q[4], dt, ksi)) -
        Lj(a, G(b[:, i1], G(b[:, i2], b[i, i3], dxs), dxs), dxs) *
        J100(i1, i2, i3, q[6], dt, ksi)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(
        b[:, i4], b[i, i5], dxs), dxs), dxs), dxs) *
        J00000(i1, i2, i3, i4, i5, q[7], dt, ksi)
        for i5 in range(m)
        for i4 in range(m)
        for i3 in range(m)
        for i2 in range(m)
        for i1 in range(m)],

       dt ** 3 / 6 * Lj(a, Lj(a, aj[i, 0], dxs), dxs),

       *[G(b[:, i1], G(b[:, i2], Lj(a, aj[i, 0], dxs), dxs), dxs) *
        (J02(i1, i2, q[6], dt, ksi) / 2 + dt * J01(i1, i2, q[2], dt, ksi) +
         dt ** 2 / 2 * J00(i1, i2, q[2], dt, ksi)) +
        Lj(a, Lj(a, G(b[:, i1], b[i, i2], dxs), dxs), dxs) / 2 *
        J20(i1, i2, q[10], dt, ksi) +
        G(b[:, i1], Lj(a, G(b[:, i2], aj[i, 0], dxs), dxs), dxs) *
        (J11(i1, i2, q[9], dt, ksi) - J02(i1, i2, q[8], dt, ksi) +
         dt * (J10(i1, i2, q[2], dt, ksi) - J01(i1, i2, q[2], dt, ksi))) +
        Lj(a, G(b[:, i1], Lj(a, b[i, i2], dxs), dxs), dxs) *
        (J11(i1, i2, q[9], dt, ksi) - J20(i1, i2, q[10], dt, ksi)) +
        G(b[:, i1], Lj(a, Lj(a, b[i, i2], dxs), dxs), dxs) *
        (J02(i1, i2, q[8], dt, ksi) / 2 + J20(i1, i2, q[10], dt, ksi) / 2 -
        J11(i1, i2, q[9], dt, ksi)) -
        Lj(a, G(b[:, i1], G(b[:, i2], aj[i, 0], dxs), dxs), dxs) *
        (dt * J10(i1, i2, q[2], dt, ksi) + J11(i1, i2, q[9], dt, ksi))
        for i2 in range(m)
        for i1 in range(m)],

       *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(
        b[:, i4], aj[i, 0], dxs), dxs), dxs), dxs) *
        (dt * J0000(i1, i2, i3, i4, q[3], dt, ksi) +
```

```
        J0001(i1, i2, i3, i4, q[11], dt, ksi)) +
       G(b[:, i1], G(b[:, i2], Lj(a, G(b[:, i3], b[i, i4], dxs), dxs), dxs), dxs) *
       (J0100(i1, i2, i3, i4, q[13], dt, ksi) - J0010(i1, i2, i3, i4, q[12], dt, ksi)) -
       Lj(a, G(b[:, i1], G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs), dxs) *
       J1000(i1, i2, i3, i4, q[14], dt, ksi) +
       G(b[:, i1], Lj(a, G(b[:, i2], G(b[:, i3], b[i, i4], dxs), dxs), dxs), dxs) *
       (J1000(i1, i2, i3, i4, q[14], dt, ksi) - J0100(i1, i2, i3, i4, q[13], dt, ksi)) +
       G(b[:, i1], G(b[:, i2], G(b[:, i3], Lj(a, b[i, i4], dxs), dxs), dxs), dxs) *
       (J0010(i1, i2, i3, i4, q[12], dt, ksi) - J0001(i1, i2, i3, i4, q[11], dt, ksi))
       for i4 in range(m)
       for i3 in range(m)
       for i2 in range(m)
       for i1 in range(m)],

      *[G(b[:, i1], G(b[:, i2], G(b[:, i3], G(b[:, i4], G(
       b[:, i5], b[i, i6], dxs), dxs), dxs), dxs), dxs) *
       J000000(i1, i2, i3, i4, i5, i6, q[15], dt, ksi)
       for i6 in range(m)
       for i5 in range(m)
       for i4 in range(m)
       for i3 in range(m)
       for i2 in range(m)
       for i1 in range(m)]

    )

  def doit(self, **hints):
    """
    Tries to expand or calculate function
    Returns
    =======
    sympy.Expr
    """
    return StrongTaylorStratonovich3p0(*self.args, **hints)
```

6.3 Source Codes for Linear Stationary Systems of Itô SDEs

Listing 132: **Implementation of supplementary functions**

```
import numpy as np


class NotASquareMatrix(Exception):
  pass


def vec_to_eye(vector):
  """
  Converts vector to eye matrix
  Parameters
  ==========
```

```
    vector : iterable
    Returns
    =======
    numpy.ndarray
    """
    n = len(vector)
    matrix = np.zeros((n, n))

    for i in range(len(matrix)):
        matrix[i][i] = vector[i]

    return matrix


def diagonal_to_column(matrix):
    """
    Converts diagonal matrix to column vector
    Parameters
    ==========
    matrix : numpy.ndarray
    Returns
    =======
    column : numpy.ndarray
    """
    height = np.shape(matrix)[0]
    if height != np.shape(matrix)[1]:
        raise NotASquareMatrix()

    column = np.zeros((height, 1))
    for i in range(height):
        column[i][0] = matrix[i][i]

    return column
```

Listing 133: **Implementation of Algorithm 11.2 [61]**

```
import numpy as np
import scipy.linalg as sci


def dindet(n: int, k: int, m_a: np.ndarray, m_b: np.ndarray, dt: float):
    """
    Algorithm 11.2 [61]
    Parameters
    ==========
    n : int
    k : int
    m_a : numpy.array
    m_b : numpy.array
    dt : float
    Returns
    =======
    m_ad : numpy.array
```

```
    m_bd : numpy.array
    """
    m_okn = np.zeros((k, n))
    m_okk = np.zeros((k, k))
    m_idt = np.eye(n + k) * dt
    m_aa = np.vstack((np.hstack((m_a, m_b)),
                      np.hstack((m_okn, m_okk))))
    m_ex_aah = sci.expm(m_aa.dot(m_idt))
    m_ad = m_ex_aah[:n, :n]
    m_bd = m_ex_aah[:n, n:(n + k)]
    return m_ad, m_bd
```

Listing 134: **Implementation of Algorithm 11.6 [61]**

```
import numpy as np

from mathematics.sde.linear.dindet import dindet
from mathematics.sde.linear.matrix import vec_to_eye


def stoch(n: int, m_a: np.ndarray, m_f: np.ndarray, dt: float):
    """
    Root function for set of algorithms implemented below
    Parameters
    ==========
    n : int
    m_a : numpy.ndarray
    m_f : numpy.ndarray
    dt : float
    Returns
    =======
    numpy.ndarray
    """
    v_l2, m_s, m_d1 = algorithm_11_2(n, m_a, m_f, dt)
    mat_l = vec_to_eye(np.sqrt(v_l2))
    return m_s.dot(mat_l)


def algorithm_11_2(n: int, m_a: np.ndarray, m_f: np.ndarray, dt: float):
    """
    Parameters
    ==========
    n : int
    m_a : numpy.ndarray
    m_f : numpy.ndarray
    dt : float
    Returns
    =======
    eigenvalues : numpy.ndarray
    eigenvectors : numpy.ndarray
    m_d1 : numpy.ndarray
    """
    m_ac = algorithm_11_5(n, m_a)
```

```
40    m_g = m_f.dot(np.transpose(m_f))
41    m_gv = algorithm_11_3(n, m_g)
42    m_dd, m_dv = dindet(int(n * (n + 1) / 2), 1, m_ac, m_gv, dt)
43    m_d1 = algorithm_11_4(n, m_dv)
44    eigenvalues, eigenvectors = np.linalg.eig(m_d1)
45    return eigenvalues, eigenvectors, m_d1
46
47
48  def algorithm_11_3(n: int, m_g: np.ndarray):
49    """
50    Algorithm 11.3 [61]
51    Parameters
52    ----------
53    n : int
54    m_g : numpy.ndarray
55    Returns
56    -------
57    m_vec : numpy.ndarray
58      column vector
59    """
60    i2 = 0
61    v_size = 0
62    for i in range(n):
63      n2 = n - i
64      for j in range(n2):
65        if v_size < j + i2:
66          v_size = j + i2
67      i2 = i2 + n - i
68
69    m_vec = np.ndarray((v_size + 1, 1))
70
71    i2 = 0
72    for i in range(n):
73      n2 = n - i
74      for j in range(n2):
75        m_vec[j + i2][0] = m_g[j][j + i]
76      i2 = i2 + n - i
77
78    return m_vec
79
80
81  def algorithm_11_4(n: int, m_dv: np.ndarray):
82    """
83    Algorithm 11.4 [61]
84    Parameters
85    ----------
86    n : int
87    m_dv : numpy.ndarray
88    Returns
89    -------
90    m_d1 : numpy.ndarray
91    """
92    i2 = 0
93    size = 0
94    for i in range(n):
```

```
95      n2 = n - i
96      for j in range(n2):
97        if size < j + i:
98          size = j + i
99      i2 = i2 + n - i
100
101   m_d1 = np.ndarray((size + 1, size + 1))
102
103   i2 = 0
104   for i in range(n):
105     n2 = n - i
106     for j in range(n2):
107       m_d1[j][j + i] = m_dv[j + i2][0]
108       m_d1[j + i][j] = m_dv[j + i2][0]
109     i2 = i2 + n - i
110
111   return m_d1
112
113
114 def algorithm_11_5(n: int, m_a: np.ndarray):
115   """
116   Algorithm 11.5 [61]
117   Parameters
118   ----------
119   n : int
120   m_a : numpy.ndarray
121   Returns
122   -------
123   m_ac : numpy.ndarray
124   """
125   r = 0
126   v_size = 0
127   h_size = 0
128
129   for i in range(n):
130     n2 = n - i
131     for j in range(n2):
132       o = 0
133       for k in range(n):
134         n3 = n - k
135         for m in range(n3):
136           if v_size < m + o:
137             v_size = m + o
138           if h_size < r:
139             h_size = r
140         o = o + n - k
141       r = r + 1
142
143   m_ones = np.zeros((n, n))
144   m_ac = np.ndarray((v_size + 1, h_size + 1))
145
146   r = 0
147   for i in range(n):
148     n2 = n - i
149     for j in range(n2):
```

```
150         i2 = j + i
151         m_ones[j][i2] = 1
152         m_ones[i2][j] = 1
153         m_one_a = m_ones.dot(np.transpose(m_a)) + m_a.dot(m_ones)
154         o = 0
155         for k in range(n):
156           n3 = n - k
157           for m in range(n3):
158             m_ac[m + o][r] = m_one_a[m][m + k]
159           o = o + n - k
160         m_ones = np.zeros((n, n))
161         r = r + 1
162 
163     return m_ac
```

Listing 135: **Implementation of the vector function $\mathbf{u}(t)$**

```
1  import numpy as np
2  from sympy import lambdify, sympify
3  
4  
5  class AbstractDistortion:
6    def t(self, t: float):
7      raise NotImplementedError("Method t is not implemented")
8  
9  
10 class Symbolic(AbstractDistortion):
11 
12   def __init__(self, fn: str):
13     from sympy.abc import t
14     self._u = lambdify(t, sympify(fn), "numpy")
15 
16   def t(self, t):
17     return self._u(t)
18 
19 
20 class ComplexDistortion(AbstractDistortion):
21   """
22   Vector function u(t)
23   """
24 
25   def __init__(self, n: int, mat_u: np.ndarray):
26     self._mat_u = mat_u
27     self._mat_ut = np.ndarray(shape=(n, 1), dtype=float)
28 
29   def t(self, t: float):
30     """
31     Provides vector function u(t) at moment t
32 
33     Parameters
34     ----------
35     t : float
36       moment of time
```

```
    Returns
    -------
    numpy.ndarray
      column u(t)
    """
    for i in range(self._mat_u.shape[0]):
      self._mat_ut[i][0] = self._mat_u[i][0].t(t)
    return self._mat_ut
```

Listing 136: **Modeling of linear system of Itô SDEs**

```
import numpy as np
from numpy import transpose

from mathematics.sde.linear.matrix import diagonal_to_column


class Integral:
  """
  Provides numerical integration
  """

  def __init__(self, n: int):
    self.n, self.t0, self.tk, self.dt, self.t = \
      n, 0, 0, 0, 0
    self.m_a, self.m_ad, self.m_bd, self.m_h, self.m_fd, self.distortion = \
      None, None, None, None, None, None
    self.m_x0, self.m_mx0, self.m_dx0, self.m_xt, self.m_mx, self.m_dx = \
      None, None, None, np.ndarray((n, 0)), np.ndarray((n, 0)), np.ndarray((n, 0))
    self.v_yt, self.v_my, self.v_dy, self.v_t = \
      [], [], [], []
    self.v_ry = []

  def integrate(self):
    """
    Performs numerical integration
    """
    higher_limit = self.t + int((self.tk - self.t0) / self.dt + 1)
    lower_limit = self.t

    self.m_xt = np.hstack((self.m_xt, np.ndarray((self.n, higher_limit - lower_limit))))
    self.m_mx = np.hstack((self.m_mx, np.ndarray((self.n, higher_limit - lower_limit))))
    self.m_dx = np.hstack((self.m_dx, np.ndarray((self.n, higher_limit - lower_limit))))

    for self.t in range(lower_limit, higher_limit):
      t = self.t0 + self.t * self.dt
      ft = np.random.randn(self.n, 1)
      mat_ut = self.distortion.t(t)

      # solution of sde
      xt = self.m_ad.dot(self.m_x0) + self.m_bd.dot(mat_ut) + self.m_fd.dot(ft)
      # exit process of stochastic system
      self.m_xt[:, self.t] = xt[:, 0]
```

```
43          self.v_yt.append(self.m_h.dot(xt)[0][0])
44
45          # expectation of solution of sde
46          mx = self.m_ad.dot(self.m_mx0) + self.m_bd.dot(mat_ut)
47          # expectation of exit process
48          self.m_mx[:, self.t] = mx[:, 0]
49          self.v_my.append(self.m_h.dot(mx)[0][0])
50
51          # dispersion of solution of sde
52          dx = self.m_ad.dot(self.m_dx0).dot(np.transpose(self.m_ad)) + self.m_fd.dot(np.
        transpose(self.m_fd))
53          # dispersion of exit process
54          self.m_dx[:, self.t] = diagonal_to_column(dx)[:, 0]
55          self.v_dy.append(self.m_h.dot(dx).dot(transpose(self.m_h))[0][0])
56
57          self.v_t.append(t)
58
59          self.m_x0, self.m_mx0, self.m_dx0 = xt, mx, dx
```

6.4 Source Codes for Utilities and Initialization

Listing 137: **Initialization module**

```
1  from config import database
2  from init.database import initdb
3  from tools.database import connect, disconnect
4
5
6  def initialization():
7    """
8    Initializes various components of application
9    """
10   connect(database)
11   initdb()
12   disconnect()
```

Listing 138: **Module for database initialization**

```
1  import csv
2  import logging
3  import os
4
5  import sympy as sp
6
7  import config as c
8  from tools import fsys
9  from tools.database import execute
10 from tools.fsys import get_files
11
12 logger = logging.getLogger(__name__)
```

```
13
14
15  def initdb():
16    """
17    Initializes database with necessary table drivers
18    """
19
20    if not fsys.is_locked(".db.lock"):
21      logger.info("Initializing database...")
22      create_files_table()
23      create_c_table()
24      fsys.lock(".db.lock")
25    else:
26      logger.info("Updating database...")
27      update_coefficients()
28
29
30  def create_files_table():
31    """
32    Initializes the Fourier-Legendre coefficients table
33    """
34    execute("DROP TABLE IF EXISTS `files`")
35    execute(
36      "CREATE TABLE `files` ("
37      "  `id`  integer PRIMARY KEY AUTOINCREMENT,"
38      "  `name` text unique"
39      ")"
40    )
41
42
43  def create_c_table():
44    """
45    Initializes the Fourier-Legendre coefficients table
46    """
47    execute("DROP TABLE IF EXISTS `C`")
48    execute(
49      "CREATE TABLE `C` ("
50      "  `id`  integer PRIMARY KEY AUTOINCREMENT,"
51      "  `index` text unique,"
52      "  `value` text,"
53      "  `value_f` double"
54      ")"
55    )
56
57    update_coefficients()
58
59
60  def update_coefficients():
61    """
62    Updates the Fourier-Legendre coefficients table
63    """
64    files = get_files(c.csv, r'c_.*\.csv')
65    loaded_files = [record[0] for record in execute("SELECT `name` FROM `files`")]
66    difference = [f for f in files if f not in loaded_files]
67
```

```
  rows = []
  for file in difference:

    with open(os.path.join(file)) as f:
      reader = csv.reader(f, delimiter=';', quotechar='"')
      for row in reader:
        if len(rows) > c.read_buffer_size:
          execute(f"INSERT INTO `C` (`index`, `value`, `value_f`) VALUES {','.join(rows)}
    ")
          rows.clear()

        rows.append(f"('{row[0]}', '{row[1]}', {float(sp.sympify(row[1]).evalf())})")

    execute(f"INSERT INTO `files` (`name`) VALUES ('{file}')")

  if len(rows) > 0:
    execute(f"INSERT INTO `C` (`index`, `value`, `value_f`) VALUES {','.join(rows)}")
```

Listing 139: **Database module**

```
import logging
import re
import sqlite3

logger = logging.getLogger(__name__)

connection: sqlite3.Connection
cursor: sqlite3.Cursor


def is_connected():
  """
  Checks if application is connected to database
  Returns
  =======
  True or False
  """
  global connection
  if connection is None:
    return False
  else:
    return True


def connect(db: str):
  """
  Connects application to database
  Parameters
  ==========
  db : str
    path to database file
  """
  try:
```

```
34      global connection
35      global cursor
36
37      connection = sqlite3.connect(db)
38      connection.create_function("REGEXP", 2, regex)
39
40      cursor = connection.cursor()
41      logger.info(f"SQLite Database is successfully connected")
42
43      query = "select sqlite_version();"
44      cursor.execute(query)
45      record = cursor.fetchall()
46      logger.info(f"SQLite Database Version is: {record[0][0]}")
47
48      query = "PRAGMA foreign_keys = ON;"
49      cursor.execute(query)
50
51    except sqlite3.Error as error:
52      logger.error(f"Error while connecting to sqlite: {error}")
53
54
55  def disconnect():
56    """
57    Disconnects application from database
58    """
59    try:
60      global connection
61      global cursor
62
63      connection.close()
64      logger.info("The SQLite connection is closed")
65
66    except sqlite3.Error as error:
67      logger.error(f"Error while connecting to sqlite: {error}")
68
69
70  def execute(query: str):
71    """
72    Sends query to database and receives data
73    Parameters
74    ==========
75    query : str
76      query to database
77    Returns
78    =======
79      list of tuples (rows)
80    """
81    try:
82      global connection
83      global cursor
84
85      cursor.execute(query)
86      records = cursor.fetchall()
87      connection.commit()
88      return records
```

```
89
90    except sqlite3.Error as error:
91      logger.error(f"Error while connecting to sqlite: {error}")
92
93
94  def regex(value, pattern):
95    """
96    Regular expression for search in database
97    Parameters
98    ==========
99    value
100     column to apply
101   pattern
102     regular expression
103   Returns
104   =======
105   Search results
106   """
107   c_pattern = re.compile(r"\b" + pattern.lower() + r"\b")
108   return c_pattern.search(value) is not None
```

Listing 140: **File system utilities**

```
1   import os
2   import re
3
4   import config as c
5
6
7   def get_files(path, pattern):
8     """
9     Gives list of files containing the Fourier-Legendre coefficients
10    Returns
11    =======
12    list
13      list of available files
14    """
15    return [os.path.join(path, f)
16        for f in os.listdir(path) if re.match(pattern, f)]
17
18
19  def is_locked(filename: str):
20    """
21    Checks if lock is set
22    Parameters
23    ==========
24    filename : str
25      name of lock file
26    Returns
27    =======
28    True of False
29    """
30    if os.path.isfile(os.path.join(c.resources, filename)):
```

```
31      return True
32    else:
33      return False
34
35
36  def lock(filename):
37    """
38    Performs locking
39    Parameters
40    ==========
41    filename : str
42      name of lock file
43    """
44    f = open(os.path.join(c.resources, filename), "w")
45    f.close()
46
47
48  def unlock(filename):
49    """
50    Performs unlocking
51    Parameters
52    ==========
53    filename : str
54      name of lock file
55    """
56    os.remove(os.path.join(c.resources, filename))
```

7 Future Work

Considering the future work, it is important to say that symbolic algebra gives a wide field for optimizations of modeling process. Symbolic operations is actually operations with strings. Such operations has relatively high complexity and slows down modeling process significantly. One of possible ways to improve modeling performance is to parallelize computations. Since strong numerical schemes for Itô SDEs have massive amount of terms this idea appears justified.

The strong numerical schemes for Itô SDEs seem to be easily optimizable, on the other hand, superpositions of the differential operators (4), (5), and (23) are not. They are called recursively during calculation process which is more difficult to parallelize than strong numerical schemes for Itô SDEs. Differential operators obviously include differentiating which is high cost and optimization of them is a dedicated issue.

In the future, it is possible to improve the SDE-MATH software package in a number of other directions. In particular, high-order strong numerical methods of the Runge-Kutta type [2], [7], [42], [61] (including multistep numerical

methods [2], [7], [42], [61]) for Itô SDEs can be implemented. In addition, software for solving the filtering problem and the problem of stochastic optimal control can also be developed. These improvements will lead to changes of the graphical user interface due to new features.

Bibliography

[1] Ito, K. On Stochastic Differential Equations. Memoirs of the American Mathematical Society, 4 (1951), 1-51.

[2] Kloeden, P.E., Platen, E. Numerical Solution of Stochastic Differential Equations. Springer, Berlin, 1992, 632 pp.

[3] Kloeden, P.E., Platen, E., Schurz, H. Numerical Solution of SDE Through Computer Experiments. Springer, Berlin, 1994, 292 pp.

[4] Arato, M. Linear Stochastic Systems with Constant Coefficients. A Statistical Approach. Springer-Verlag, Berlin, Heidelberg, N.Y., 1982, 289 pp.

[5] Shiryaev, A.N. Essentials of Stochastic Finance: Facts, Models, Theory. World Scientific Publishing Co United States, 1999, 852 pp.

[6] Karatzas, I., Shreve, S. Methods of Mathematical Finance. Springer-Verlag, New York, 1998, 415 pp.

[7] Platen, E., Bruti-Liberati, N. Numerical Solution of Stochastic Differential Equations with Jumps in Finance. Springer, Berlin, Heidelberg, 2010, 868 pp.

[8] Han, X., Kloeden, P.E., Random Ordinary Differential Equations and Their Numerical Solution. Springer, Singapore, 2017, 250 pp.

[9] Allen, E. Modeling with Ito Stochastic Differential Equations. Springer, Dordrecht, 2007, 230 pp.

[10] Merton, R.C. Continuous-Time Finance. Blackwell, Oxford, 1992, 754 pp.

[11] Heston, S.L. A closed-form solution for options with stochastic volatility with applications to bond and currency options, Rev. Financial Studies, 6, 2 (1993), 327–343.

[12] Cox, J.C., Ingersoll, J.E., Ross, S.A. A Theory of the term structure of interest rates, Econometrica, 53 (1985), 385-408.

[13] Kimura, M., Ohta, T. Theoretical aspects of Population genetics. Princeton Univ. Press, Boston, 1971, 232 pp.

[14] Iacus, S.M. Simulation and Inference for Stochastic Differential Equations. With R Examples. Springer-Verlag, New York, 2008, 285 pp.

[15] Ricciardi, L.M. Diffusion Processes and Related Topics in Biology, Lecture Notes in Biomathematics. Springer, New York, 1977, 202 pp.

[16] Stratonovich, R.L., Selected Problems of Fluctuations Theory in Radio Engineering. [In Russian]. Sovetskoe Radio, Moscow, 1961, 556 pp.

[17] Liptser, R.Sh., Shirjaev, A.N. Statistics of stochastic processes: nonlinear filtering and related problems. [In Russian]. Nauka, Moscow, 1974, 696 pp.

[18] Nasyrov, F.S. Local times, symmetric integrals and stochastic analysis. [In Russian]. Fizmatlit Publ., Moscow, 2011, 212 pp.

[19] Kagirova, G.R., Nasyrov, F.S. On an optimal filtration problem for one-dimensional diffusion processes. Siberian Adv. Math., 28, 3 (2018), 155-165.

[20] Chugai, K.N., Kosachev, I.M., Rybakov, K.A. Approximate filtering methods in continuous-time stochastic systems. Smart Innovation, Systems and Technologies, vol. 173, Eds. Jain L.C., Favorskaya M.N., Nikitin I.S., Reviznikov D.L. Springer, 2020, pp. 351-371. DOI: http://doi.org/10.1007/978-981-15-2600-8_24

[21] Averina, T.A., Rybakov, K.A. Using maximum cross section method for filtering jump-diffusion random processes. Russian Journal of Numerical Analysis and Mathematical Modelling. 35, 2 (2020), 55-67. DOI: http://doi.org/10.1515/rnam-2020-0005

[22] Kloeden, P.E., Platen, E., Schurz, H., Sorensen, M. On effects of discretization on estimators of drift parameters for diffusion processes. J. Appl. Probab., 33 (1996), 1061-1076.

[23] Clark, J.M.C., Cameron, R.J. The maximum rate of convergence of discrete approximations for stochastic differential equations. Stochastic Differential Systems Filtering and Control. Lecture Notes in Control and

Information Sciences, vol 25. Ed. Grigelionis B. Springer, Berlin, Heidelberg, 1980, pp. 162-171.

[24] Kulchitskiy, O.Yu., Kuznetsov, D.F. The unified Taylor–Ito expansion. Journal of Mathematical Sciences (New York), 99, 2 (2000), 1130-1140. DOI: http://doi.org/10.1007/BF02673635

[25] Kuznetsov, D.F. New representations of the Taylor–Stratonovich expansion. Journal of Mathematical Sciences (New York), 118, 6 (2003), 5586-5596. DOI: http://doi.org/10.1023/A:1026138522239

[26] Kuznetsov, D.F. Strong approximation of iterated Ito and Stratonovich stochastic integrals based on generalized multiple Fourier series. Application to numerical solution of Ito SDEs and semilinear SPDEs. Differencialnie Uravnenia i Protsesy Upravlenia, 4 (2020), A.1-A.606. Available at: http://diffjournal.spbu.ru/EN/numbers/2020.4/article.1.8.html

[27] Milstein, G.N. Numerical Integration of Stochastic Differential Equations. [In Russian]. Ural University Press, Sverdlovsk, 1988, 225 pp.

[28] Kloeden, P.E., Platen, E., Wright, I.W. The approximation of multiple stochastic integrals. Stochastic Analysis and Applications, 10, 4 (1992), 431-441.

[29] Averina, T.A., Prigarin, S.M. Calculation of stochastic integrals of Wiener processes. [In Russian]. Preprint 1048. Novosibirsk, Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of the Russian Academy of Sciences, 1995, 15 pp.

[30] Prigarin, S.M., Belov, S.M. One application of series expansions of Wiener process. [In Russian]. Preprint 1107. Novosibirsk, Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of the Russian Academy of Sciences, 1998, 16 pp.

[31] Wiktorsson, M. Joint characteristic function and simultaneous simulation of iterated Ito integrals for multiple independent Brownian motions. The Annals of Applied Probability, 11, 2 (2001), 470-487.

[32] Ryden, T., Wiktorsson, M. On the simulation of iterated Ito integrals. Stochastic Processes and their Applications, 91, 1 (2001), 151-168.

[33] Gaines, J.G., Lyons, T.J. Random generation of stochastic area integrals. SIAM J. Appl. Math., 54 (1994), 1132-1146.

[34] Milstein, G.N., Tretyakov, M.V. Stochastic Numerics for Mathematical Physics. Springer, Berlin, 2004, 616 pp.

[35] Allen, E. Approximation of triple stochastic integrals through region subdivision. Communications in Applied Analysis (Special Tribute Issue to Professor V. Lakshmikantham), 17 (2013), 355-366.

[36] Rybakov, K.A. Applying spectral form of mathematical description for representation of iterated stochastic integrals. [In Russian]. Differencialnie Uravnenia i Protsesy Upravlenia, 4 (2019), 1-31. Available at: http://diffjournal.spbu.ru/EN/numbers/2019.4/article.1.1.html

[37] Tang, X., Xiao, A. Asymptotically optimal approximation of some stochastic integrals and its applications to the strong second-order methods. Advances in Computational Mathematics, 45 (2019), 813-846.

[38] Zahri, M. Multidimensional Milstein scheme for solving a stochastic model for prebiotic evolution. Journal of Taibah University for Science, 8, 2 (2014), 186-198.

[39] Li, C.W., Liu, X.Q. Approximation of multiple stochastic integrals and its application to stochastic differential equations. Nonlinear Anal. Theor. Meth. Appl., 30, 2 (1997), 697-708.

[40] Rybakov K. Application of Walsh series to represent iterated Stratonovich stochastic integrals. IOP Conference Series: Materials science and engineering. 2020, vol. 927, id 012080. DOI: http://doi.org/10.1088/1757-899X/927/1/012080

[41] Rybakov, K.A. Modeling and analysis of output processes of linear continuous stochastic systems based on orthogonal expansions of random functions. J. of Computer and Systems Sci. Int., 59, 3 (2020), 322-337. DOI: http://doi.org/10.1134/S1064230720030156

[42] Kuznetsov, D.F. Numerical Integration of Stochastic Differential Equations. 2. [In Russian]. Polytechnical University Publishing House, Saint-Petersburg, 2006, 764 pp. DOI: http://doi.org/10.18720/SPBPU/2/s17-227

[43] Kuznetsov, D.F. A method of expansion and approximation of repeated stochastic Stratonovich integrals based on multiple Fourier series on full orthonormal systems. [In Russian]. Differencialnie Uravnenia i Protsesy

Upravlenia, 1 (1997), 18-77. Available at: http://diffjournal.spbu.ru/EN/numbers/1997.1/article.1.2.html

[44] Kuznetsov, D.F. Comparative analysis of the efficiency of application of Legendre polynomials and trigonometric functions to the numerical integration of Ito stochastic differential equations. Computational Mathematics and Mathematical Physics, 59, 8 (2019), 1236-1250. DOI: http://doi.org/10.1134/S0965542519080116

[45] Kuznetsov, D.F. Application of the method of approximation of iterated stochastic Itô integrals based on generalized multiple Fourier series to the high-order strong numerical methods for non-commutative semilinear stochastic partial differential equations. Differencialnie Uravnenia i Protsesy Upravlenia, 3 (2019), 18-62. Available at: http://diffjournal.spbu.ru/EN/numbers/2019.3/article.1.2.html

[46] Kuznetsov, D.F. Application of multiple Fourier–Legendre series to strong exponential Milstein and Wagner–Platen methods for non-commutative semilinear stochastic partial differential equations. Differencialnie Uravnenia i Protsesy Upravlenia, 3 (2020), 129-162. Available at: http://diffjournal.spbu.ru/EN/numbers/2020.3/article.1.6.html

[47] Jentzen, A., Röckner, M. A Milstein scheme for SPDEs. Foundations Comp. Math., 15, 2 (2015), 313-362.

[48] Becker, S., Jentzen, A., Kloeden, P.E. An exponential Wagner–Platen type scheme for SPDEs. SIAM J. Numer. Anal., 54, 4 (2016), 2389-2426.

[49] Mishura, Y.S., Shevchenko, G.M. Approximation schemes for stochastic differential equations in a Hilbert space. Theor. Prob. Appl., 51, 3 (2007), 442-458.

[50] Bao, J., Reisinger, C., Renz, P., Stockinger, W. First order convergence of Milstein schemes for McKean equations and interacting particle systems arXiv:2004.03325v1 [math.PR], 2020, 27 pp.

[51] Son, L.N., Tuan, A.H., Dung, T.N., Yin G. Milstein-type procedures for numerical solutions of stochastic differential equations with Markovian switching. SIAM J. Numer. Anal., 55, 2 (2017), 953–979.

[52] Sun, Y., Yang, J., Zhao W. Ito–Taylor schemes for solving mean-field stochastic differential equations. Numer. Math. Theor. Meth. Appl., 10, 4 (2017), 798-828.

[53] Higham, D.J. An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations. SIAM Rev., 43, 3 (2001), 525–546.

[54] Cyganowski, S., Grune, L., Kloeden, P.E. Maple for stochastic differential equations. Theory and Numerics of Differential Equations. Eds. Blowey, J.F., Coleman, J.P., Craig, A.W. Universitext. Springer, Berlin, Heidelberg, 2001, pp. 127-177.

[55] Higham, D.J., Kloeden, P.E. MAPLE and MATLAB for stochastic differential equations in finance. Programming Languages and Systems in Computational Economics and Finance. Advances in Computational Economics, vol 18, Ed. Nielsen, S.S. Springer, Boston, MA, 2002, pp. 233-269.

[56] Cyganowski, S., Grune, L., Kloeden P.E. MAPLE for jump-diffusion stochastic differential equations in finance. Programming Languages and Systems in Computational Economics and Finance. Advances in Computational Economics, vol. 18, Ed. Nielsen, S.S. Springer, Boston, MA, 2002, pp. 441-460.

[57] Gilsing, H., Shardlow, T. SDELab: A package for solving stochastic differential equations in MATLAB. Journal of Computational and Applied Mathematics, 2, 205 (2007), 1002-1018.

[58] Kuznetsov, D.F. Stochastic Differential Equations: Theory and Practice of Numerical Solution. With MatLab programs, 3rd Edition. [In Russian]. Polytechnical University Publishing House, Saint-Petersburg, 2009, 768 pp. DOI: http://doi.org/10.18720/SPBPU/2/s17-230

[59] Kiesewetter, S., Polkinghorne, R., Opanchuk, B., Drummond, P.D. xSPDE: Extensible software for stochastic equations. SoftwareX, 5 (2016), 12-15.

[60] 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. DepCoS-RELCOMEX 2016. Advances in Intelligent Systems and Computing, vol. 470, Eds. Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. Springer, Cham, 2016, pp. 169-179.

[61] Kuznetsov, D.F. Stochastic differential equations: theory and practice of numerical solution. With MATLAB programs, 6th Edition. [In Russian]. Differencialnie Uravnenia i Protsesy Upravlenia, 4 (2018), A.1-A.1073.

Available at: http://diffjournal.spbu.ru/EN/numbers/2018.4/article.2.1.html

[62] Kulchitskiy, O.Yu., Kuznetsov, D.F. Numerical Simulation of Stochastic Systems of Linear Stationary Differential Equations. [In Russian]. Differencialnie Uravnenia i Protsesy Upravlenia, 1 (1998), 41-65. Available at: http://diffjournal.spbu.ru/pdf/j010.pdf

[63] Gihman, I.I., Skorohod A.V. Stochastic Differential Equations and its Applications. [In Russian]. Naukova Dumka, Kiev, 1982, 612 pp.

[64] Kuznetsov, D.F. On Numerical Modeling of the Multidimensional Dynamic Systems Under Random Perturbations With the 1.5 and 2.0 Orders of Strong Convergence. Automation and Remote Control, 79, 7 (2018), 1240-1254. DOI: http://doi.org/10.1134/S0005117918070056

[65] Kuznetsov, D.F. On Numerical Modeling of the Multidimentional Dynamic Systems Under Random Perturbations With the 2.5 Order of Strong Convergence. Automation and Remote Control, 80, 5 (2019), 867-881. DOI: http://doi.org/10.1134/S0005117919050060

[66] Kuznetsov, D.F. Explicit one-step mumerical method with the strong convergence order of 2.5 for Ito stochastic differential equations with a multi-dimensional nonadditive noise based on the Taylor–Stratonovich expansion. Computational Mathematics and Mathematical Physics, 60, 3 (2020), 379-389. DOI: http://doi.org/10.1134/S0965542520030100

[67] Kuznetsov, D.F. Strong approximation of iterated Ito and Stratonovich stochastic integrals based on generalized multiple Fourier series. Application to numerical solution of Ito SDEs and semilinear SPDEs. arXiv:2003.14184 [math.PR], 2020, 652 pp.

[68] Kuznetsov, D.F. Multiple Ito and Stratonovich stochastic integrals: approximations, properties, formulas. Polytechnical University Publishing House: St.-Petersburg, 2013, 382 pp. DOI: http://doi.org/10.18720/SPBPU/2/s17-234

[69] Kuznetsov, D.F. Expansion of iterated Stratonovich stochastic integrals based on generalized multiple Fourier series. Ufa Mathematical Journal, 11, 4 (2019), 49-77. DOI: http://doi.org/10.13108/2019-11-4-49 Available at: http://matem.anrb.ru/en/article?art_id=604

[70] Kuznetsov, D.F. Development and application of the Fourier method for the numerical solution of Ito stochastic differential equations. Computational Mathematics and Mathematical Physics, 58, 7 (2018), 1058-1070. DOI: http://doi.org/10.1134/S0965542518070096

[71] Kuznetsov, D.F. Mean Square Approximation of Solutions of Stochastic Differential Equations Using Legendres Polynomials. Journal of Automation and Information Sciences (Begell House), 32, Issue 12, (2000), 69-86. DOI: http://doi.org/10.1615/JAutomatInfScien.v32.i12.80

[72] Kuznetsov, D.F. Multiple Ito and Stratonovich stochastic integrals: Fourier–Legendre and trigonometric expansions, approximations, formulas. Differencialnie Uravnenia i Protsesy Upravlenia, 1 (2017), A.1–A.385. Available at: http://diffjournal.spbu.ru/EN/numbers/2017.1/article.2.1.html

[73] Kuznetsov, M.D., Kuznetsov, D.F. Optimization of the mean-square approximation procedures for iterated Ito stochastic integrals of multiplicities 1 to 5 from the unified Taylor–Ito expansion based on multiple Fourier–Legendre series arXiv:2010.13564 [math.PR], 2020, 59 pp.

[74] Kuznetsov, M.D., Kuznetsov, D.F. Implementation of strong numerical methods of orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Ito SDEs with non-commutative noise based on the unified Taylor–Ito and Taylor–Stratonovich expansions and multiple Fourier–Legendre series. arXiv:2009.14011 [math.PR], 2020, 188 pp.

[75] Kuznetsov, D.F., Kuznetsov, M.D. A software package for implementation of strong numerical methods of convergence orders 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 for Ito SDEs with non-commutative multi-dimensional noise. 19th International Conference "Aviation and Cosmonautics" (AviaSpace-2020). Abstracts (Moscow, MAI, 23-27 November, 2020), Publishing house "Pero", 2020, 569-570.

[76] Kuznetsov, D.F., Kuznetsov, M.D. Mean-square approximation of iterated stochastic integrals from strong exponential Milstein and Wagner-Platen methods for non-commutative semilinear SPDEs based on multiple Fourier-Legendre series. Springer Proceedings in Mathematics & Statistics. Recent Developments in Stochastic Methods and Applications. Ed. Shiryaev, A.N. Samouylov, K.E, Kozyrev, D.V., 2021 (to appear).