

РЕАЛИЗАЦИЯ МЕТОДА СТОХАСТИЗАЦИИ ОДНОШАГОВЫХ ПРОЦЕССОВ В СИСТЕМЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ *

© 2018 г. М. Н. Геворкян^{1,*}, А. В. Демидова^{1,**}, Т. Р. Велиева^{1,***},
А. В. Королькова^{1,****}, Д. С. Кулябов^{1,2,*****}, Л. А. Севастьянов^{1,3,*****}

¹ Кафедра прикладной информатики и теории вероятностей,

Российский университет дружбы народов,

117198, Москва, ул. Миклухо-Маклая, 6

² Лаборатория информационных технологий,

Объединенный институт ядерных исследований,

141980, Московская область, Дубна, ул. Жолио-Кюри, 6

³ Лаборатория теоретической физики,

Объединенный институт ядерных исследований,

141980, Московская область, Дубна, ул. Жолио-Кюри, 6

E-mail: gevorkyan_mn@rudn.university*, demidova_av@rudn.university**,

velieva_tr@rudn.university***, korolkova_av@rudn.university****,

kulyabov_ds@rudn.university*****, sebastianov_la@rudn.university*****

Поступила в редакцию 15.09.2017

При моделировании таких явлений как популяционная динамика, исследование управляемых потоков и т.д. возникает проблема адаптации существующих моделей под исследуемое явление. Для этого предлагается получать новые модели из первых принципов на основе метода стохастизации одношаговых процессов. Исследование имеет вид итеративного процесса, заключающегося в получении модели и последующей ее корректировке. Количество таких итераций может быть крайне большим. Целью данной работы является разработка программной реализации средствами компьютерной алгебры метода стохастизации одношаговых процессов. В работе предложено использовать систему компьютерной алгебры *SymPy* в качестве основы для программной реализации. На основе разработанного алгоритма показано получение стохастических дифференциальных уравнений их вида схем взаимодействия. Результаты работы программы продемонстрированы на моделях Ферхюльста и Лотки–Вольтерра.

1. ВВЕДЕНИЕ

Многие физические и технические явления можно описывать в рамках статистического подхода. Обычно подбирается модель, достаточно полно отражающая изучаемое явление, и в нее вносятся некоторые уточнения. Возникает вопрос, как вносить изменения, поскольку данный процесс не однозначен. Используемые модели являются реализацией некоторых первых принци-

пов. Мы считаем, что если строить модели из первых принципов, то вносимые изменения будут выражать внутреннюю структуру модели. Для описания изучаемых нами явлений (сети передачи данных, системы с управлением, популяционная динамика) мы используем модель одношаговых процессов [1, 2].

Нами разработана методика стохастизации моделей, которая позволяет получать из первых принципов и стохастическую модель, и соответствующую ей детерминистическую [3, 4, 5, 6, 7]. Процесс исследования является итеративным: из детерминистической модели мы получаем пер-

*Работа частично поддержана грантами РФФИ № 15-07-08795, 16-07-00556. Также публикация подготовлена при поддержке программы РУДН “5-100”.

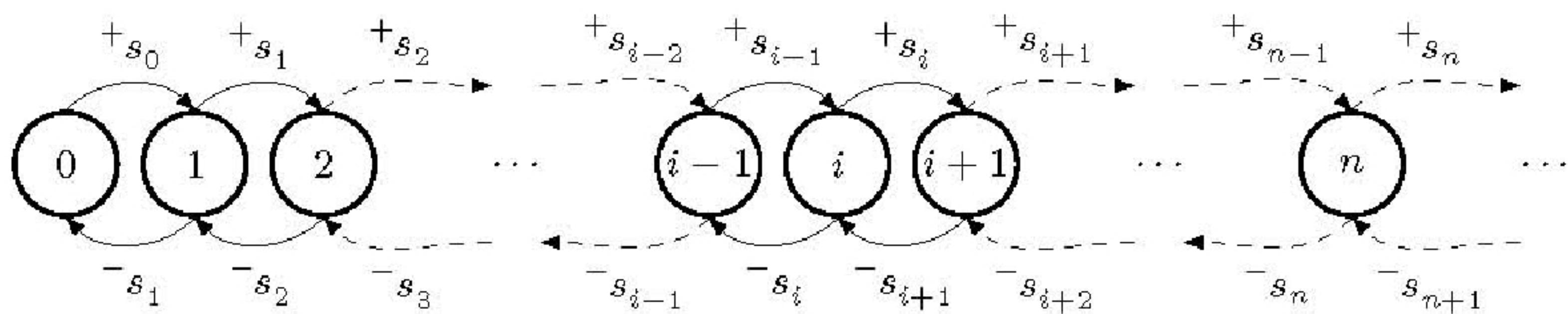


Рис. 1.
Одношаговый процесс.

вичную модель, из первичной модели — стохастическую, стохастическая модель соотносится с детерминистической, на основании этого соотнесения мы получает уточненную первичную модель. Далее процесс повторяется.

Формализм стохастизации может быть реализован разными способами. На данный момент нами применяются представление векторов состояния (комбинаторный подход) и представление чисел заполнения (операторный подход) [8, 9, 10, 11].

В случае применения комбинаторного подхода все действия выполняются в пространстве векторов состояния системы. На протяжении всех модельных манипуляций мы имеем дело с конкретной исследуемой системой. В результате получается описание в виде дифференциального уравнения. Этот подход удобен при конструировании модели, поскольку позволяет легко сравнить результат с другими моделями.

В операторном подходе мы отвлекаемся от конкретной реализации исследуемой системы, работая с абстрактными операторами. В пространство векторов состояний мы переходим только в конце вычислений. Кроме того, конкретную операторную алгебру мы выбираем, исходя их симметрии задачи. Этот подход удобен при теоретических построениях.

Зачастую возникает задача нахождения стохастической модели, эквивалентной ранее созданной детерминистической. Для этого мы используем комбинаторный подход. В этом представлении стохастическая модель имеет вид дифференциального уравнения, что облегчает сравнение с исходной моделью.

Структура статьи следующая. В разделе 2 введены основные обозначения и соглашения. Суть метода стохастизации одношаговых процессов и

его составные части описаны в разделе 3. В разделе 4 описан программный комплекс, реализующий метод стохастизации одношаговых процессов. В разделе 4.1 проводится сравнение систем компьютерной алгебры и выбор конкретной системы для реализации. В разделе 4.2 описаны основные фрагменты программы реализации метода. В разделах 4.3 и 4.4 приведены примеры конкретного применения метода.

2. ОБОЗНАЧЕНИЯ И СОГЛАШЕНИЯ

1. В работе для тензорных величин используются абстрактные индексы [12], то есть компоненты тензора обозначаются подчеркнутым индексом (например, x^i), а тензор как целостный объект обозначается индексом без модификаций (например, x^i),
2. Также будем придерживаться следующих соглашений. Латинские индексы из середины алфавита (i, j, k) будут относиться к пространству векторов состояний системы. Латинские индексы из начала алфавита (a) будут относиться к пространству винеровского процесса. Греческие индексы (α) будут задавать количество разных взаимодействий в кинетических уравнениях.

3. МЕТОДИКА СТОХАСТИЗАЦИИ ДЕТЕРМИНИРОВАННЫХ МОДЕЛЕЙ

Нами разработана эмпирическая методика стохастизации одношаговых процессов. Методика formalизована таким образом, что для ее применения достаточно сформулировать исходную задачу в виде марковского процесса. При этом лишь часть шагов явно алгоритмизирована.

Первым шагом мы приводим нашу модель к виду одношагового процесса (см. рис. 1). Далее необходимо формализовать этот процесс в виде схем взаимодействия [13, 5]. Аналогами схем взаимодействия являются уравнения химической кинетики, реакции частиц и т.д.

Для схем взаимодействия создан алгоритм (вернее будет сказать, семейство алгоритмов — для разных подходов разные алгоритмы), позволяющий из схем взаимодействия получить основное кинетическое уравнение. Однако это уравнение [2, 1] имеет обычно достаточно сложную структуру, что затрудняет его исследование и решение. Следующим шагом мы получаем приближенные модели в виде уравнений Фоккера–Планка и Ланжевена.

Предложенный подход подразумевает итеративность исследования: полученные приближенные модели уточняются, что приводит к коррекции исходных схем взаимодействия.

При построении моделей мы работаем по следующему итеративному алгоритму (алгоритм 1).

Исходные параметры: детерминистическое уравнение

Результат: стохастическое уравнение

до тех пор, пока не получено

соответствие уравнений выполнять

схема взаимодействия ←

детерминистическое уравнение ;

стохастическое уравнение ← схема взаимодействия;

если стохастическое уравнение

удовлетворяет модели **тогда**

| завершить стохастизацию;

иначе

| вернуться к началу;

конец

конец

Алгоритм 1: Алгоритм работы исследователя

К сожалению, большая часть этого алгоритма не формализована. Получение схем взаимодействия, определение соответствия уравнений — все это тяжело формализуемые задачи. Однако проблема формализации получения стохастических уравнений из схем взаимодействия нами решена (алгоритмы 2 и 3).

Процесс этот несложный, но крайне трудоем-

Исходные параметры: схема взаимодействия (1)

Результат: уравнение Ланжевена (12) и (13)

начало блока

Операторы состояния системы

$I_j^{i\alpha}, F_j^{i\alpha} \leftarrow$ схема взаимодействия (1);

Оператор изменения состояния системы

$r_j^{i\alpha} \leftarrow I_j^{i\alpha}, F_j^{i\alpha}$ (2);

Интенсивности переходов

${}^+s_\alpha, {}^-s_\alpha \leftarrow I_j^{i\alpha}, F_j^{i\alpha}, r_j^{i\alpha}$ (7);

основное кинетическое уравнение ←

${}^+s_\alpha, {}^-s_\alpha, r_j^{i\alpha}$ (6) ;

уравнение Фоккера–Планка ←

основное кинетическое уравнение (8),

(9) ;

уравнение Ланжевена ← *уравнение*

Фоккера–Планка (12), (13) ;

конец

Алгоритм 2: Алгоритм стохастизации

Исходные параметры: схема взаимодействия (1)

Результат: уравнение Ланжевена (12) и (13)

начало блока

Операторы состояния системы

$I_j^{i\alpha}, F_j^{i\alpha} \leftarrow$ Схема взаимодействия (1);

Оператор изменения состояния

системы

$r_j^{i\alpha} \leftarrow I_j^{i\alpha}, F_j^{i\alpha}$ (2);

Интенсивности переходов

${}^+s_\alpha, {}^-s_\alpha \leftarrow I_j^{i\alpha}, F_j^{i\alpha}, r_j^{i\alpha}$ (11);

уравнение Ланжевена ← ${}^+s_\alpha, {}^-s_\alpha, r_j^{i\alpha}$ (12), (13) ;

конец

Алгоритм 3: Упрощенный алгоритм стохастизации

кий. Опишем основные этапы этого процесса более подробно.

3.1. Схемы взаимодействия

Состояние системы будем описывать вектором состояний $\varphi^i \in \mathbb{R}^n$, где n — размерность системы. Оператор $I_j^i \in \mathbb{N}_0^n \times \mathbb{N}_0^n$ задает состояние системы до взаимодействия, оператор $F_j^i \in \mathbb{N}_0^n \times \mathbb{N}_0^n$ —

после. Компонентные индексы размерности системы пробегают значения $i, j = \overline{1, n}$. В результате взаимодействия происходит переход системы в другое состояние.

В системе может происходить s видов различных взаимодействий. Поэтому вместо операторов I_j^i и F_j^i будем рассматривать операторы $I_j^{i\alpha} \in \mathbb{N}_0^n \times \mathbb{N}_0^n \times \mathbb{N}_+^s$ и $F_j^{i\alpha} \in \mathbb{N}_0^n \times \mathbb{N}_0^n \times \mathbb{N}_+^s$. Компонентные индексы количества взаимодействий пробегают значения $\underline{\alpha} = \overline{1, s}$.

Взаимодействие элементов системы описывается с помощью схем взаимодействия.

$$I_j^{i\alpha} \varphi^j \xrightleftharpoons[-k_{\underline{\alpha}}]{+k_{\underline{\alpha}}} F_j^{i\alpha} \varphi^j, \quad \underline{\alpha} = \overline{1, s}. \quad (1)$$

Здесь греческие индексы задают количество взаимодействий, а латинские — размерность системы. Коэффициенты $+k_{\underline{\alpha}}$ и $-k_{\underline{\alpha}}$ имеют смысл интенсивности (скорости) взаимодействия.

Изменение состояния будет задаваться оператором

$$r_j^{i\alpha} = F_j^{i\alpha} - I_j^{i\alpha}. \quad (2)$$

Таким образом, один шаг взаимодействия $\underline{\alpha}$ в прямом и обратном направлениях можно записать соответственно как

$$\begin{aligned} \varphi^i &\rightarrow \varphi^i + r_j^{i\alpha} \varphi^j, \\ \varphi^i &\rightarrow \varphi^i - r_j^{i\alpha} \varphi^j. \end{aligned}$$

Чаще всего модель строится таким образом, что тензоры $I_j^{i\alpha}$ и $F_j^{i\alpha}$ диагональны по латинским индексам. Поэтому мы можем явно использовать диагональные сечения этих матриц и записывать (1) в более привычном виде (как систему линейных уравнений) [2]:

$$I_j^{i\alpha} \varphi^j \delta_i \xrightleftharpoons[-k_{\underline{\alpha}}]{+k_{\underline{\alpha}}} F_j^{i\alpha} \varphi^j \delta_i, \quad (3)$$

где $\delta_i = (1, \dots, 1)$.

Также мы будем использовать следующие обозначения:

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

3.2. Основное кинетическое уравнение

Для одношаговых процессов в качестве кинетического уравнения рассматривается основное кинетическое уравнение [2, 1].

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

где $w(\varphi | \psi, t)$ — вероятность перехода из состояния ψ в состояние φ за единицу времени.

Основное кинетическое уравнение можно рассматривать как реализацию уравнения Колмогорова. Однако основное кинетическое уравнение более удобно и имеет непосредственную физическую интерпретацию [2].

Зафиксировав начальные значения φ_1, t_1 , можно записать данное уравнение для подан-самбля:

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

В системе, описываемой одношаговыми процессами, возможны два вида перехода системы из одного состояния в другое, происходящие в результате взаимодействия элементов в прямом направлении ($\varphi^i + r_j^{i\alpha} \varphi^j$) с вероятностью $+s_{\underline{\alpha}}(\varphi^k)$ и в обратном направлении ($\varphi^i - r_j^{i\alpha} \varphi^j$) с вероятностью $-s_{\underline{\alpha}}(\varphi^k)$ (рис. 1). А матрица вероятностей переходов может быть записана в виде:

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

где $\delta_{i,j}$ — символ Кронекера.

Таким образом, общий вид основного кинетического уравнения (5) для вектора состояний φ^i , изменяющегося шагами длины $r_j^{i\alpha} \varphi^j$, принимает вид:

$$\begin{aligned} \frac{\partial p(\varphi^i, t)}{\partial t} = & \sum_{\underline{\alpha}=1}^s \left\{ -s_{\underline{\alpha}}(\varphi^i + r^{i\underline{\alpha}}, t) p(\varphi^i + r^{i\underline{\alpha}}, t) + \right. \\ & + +s_{\underline{\alpha}}(\varphi^i - r^{i\underline{\alpha}}, t) p(\varphi^i - r^{i\underline{\alpha}}, t) - \\ & \left. - [+s_{\underline{\alpha}}(\varphi^i) + -s_{\underline{\alpha}}(\varphi^i)] p(\varphi^i, t) \right\}. \quad (6) \end{aligned}$$

Получим функции ${}^+s_{\underline{\alpha}}$ и ${}^-s_{\underline{\alpha}}$ для уравнения (6), исходя из комбинаторного подхода.

Интенсивности перехода в единицу времени ${}^+s_{\underline{\alpha}}$ и ${}^-s_{\underline{\alpha}}$ пропорциональны соответственно числу способов выбора числа размещений из φ_-^i по $I_{-\underline{\alpha}}^{i\underline{\alpha}}$ (обозначается как $A_{\varphi_-^i}^{I_{-\underline{\alpha}}^{i\underline{\alpha}}}$) и по $F_{-\underline{\alpha}}^{i\underline{\alpha}}$ (обозначается как $A_{\varphi_-^i}^{F_{-\underline{\alpha}}^{i\underline{\alpha}}}$) и определяются выражениями:

$$\begin{aligned} {}^+s_{\underline{\alpha}} &= {}^+k_{\underline{\alpha}} \prod_{i=1}^n A_{\varphi_-^i}^{I_{-\underline{\alpha}}^{i\underline{\alpha}}} = {}^+k_{\underline{\alpha}} \prod_{i=1}^n \frac{\varphi_-^i!}{(\varphi_-^i - I_{-\underline{\alpha}}^{i\underline{\alpha}})!}, \\ {}^-s_{\underline{\alpha}} &= {}^-k_{\underline{\alpha}} \prod_{i=1}^n A_{\varphi_-^i}^{F_{-\underline{\alpha}}^{i\underline{\alpha}}} = {}^-k_{\underline{\alpha}} \prod_{i=1}^n \frac{\varphi_-^i!}{(\varphi_-^i - F_{-\underline{\alpha}}^{i\underline{\alpha}})!}. \end{aligned} \quad (7)$$

3.3. Уравнение Фоккера–Планка

Уравнение Фоккера–Планка является частным случаем основного кинетического уравнения и может рассматриваться как его приближенная форма. Его можно получить путем разложения основного кинетического уравнения в ряд до членов второго порядка включительно. Для этого можно использовать разложение Крамерса–Мойала [1] (для простоты записано в одномерном случае):

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

где

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

Отбрасывая члены выше второго порядка, получаем уравнение Фоккера–Планка:

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

или в многомерном случае:

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

где

$$\begin{aligned} A^i &:= A^i(\varphi^k) = r^{i\underline{\alpha}} \left[{}^+_s_{\underline{\alpha}} - {}^-s_{\underline{\alpha}} \right], \\ B^{ij} &:= B^{ij}(\varphi^k) = r^{i\underline{\alpha}} r^{j\underline{\alpha}} \left[{}^+_s_{\underline{\alpha}} - {}^-s_{\underline{\alpha}} \right]. \end{aligned} \quad (10)$$

В результате разложения Крамерса–Мойала можно в (7) заменить комбинации типа $\varphi(\varphi - 1) \cdots (\varphi - (n - 1))$ на $(\varphi)^n$, получим для уравнения Фоккера–Планка:

$$\begin{aligned} {}^+_s_{\underline{\alpha}} &= {}^+k_{\underline{\alpha}} \prod_{i=1}^n (\varphi_-^i)^{I_{-\underline{\alpha}}^{i\underline{\alpha}}}, \\ {}^-s_{\underline{\alpha}} &= {}^-k_{\underline{\alpha}} \prod_{i=1}^n (\varphi_-^i)^{F_{-\underline{\alpha}}^{i\underline{\alpha}}}. \end{aligned} \quad (11)$$

Как видно из (10), коэффициенты уравнения Фоккера–Планка можно получить сразу из (2) и (7), то есть в данном случае записывать основное кинетическое уравнение нет необходимости.

3.4. Уравнение Ланжевена

Уравнению Фоккера–Планка соответствует уравнение Ланжевена:

$$d\varphi^i = a^i dt + b_a^i dW^a, \quad (12)$$

где $a^i := a^i(\varphi^k)$, $b_a^i := b_a^i(\varphi^k)$, $\varphi^i \in \mathbb{R}^n$ — вектор состояний системы, $W^a \in \mathbb{R}^m$ — m -мерный винеровский процесс. Винеровский процесс реализуется как $dW = \varepsilon \sqrt{dt}$, где $\varepsilon \sim N(0, 1)$ — нормальное распределение со средним 0 и дисперсией 1. Здесь латинскими индексами из середины алфавита обозначаются величины, относящиеся к векторам состояний (размерность пространства n), а латинскими индексами из начала алфавита обозначаются величины, относящиеся к вектору винеровского процесса (размерность пространства $m \leq n$).

При этом связь между коэффициентами уравнений (9) и (12) выражается следующими соотношениями:

$$A^i = a^i, \quad B^{ij} = b_a^i b^{ja}. \quad (13)$$

Видно, что второй член уравнения Ланжевена представляет из себя квадратный корень, который имеет сложный вид в многомерном случае. Впрочем заметим, что во многих соотношениях используется именно квадрат второго члена уравнения Ланжевена, так что явное вычисление корня зачастую и не требуется.

4. РЕАЛИЗАЦИЯ МОДЕЛИ ОДНОШАГОВЫХ СТОХАСТИЧЕСКИХ ПРОЦЕССОВ В СИСТЕМЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

4.1. Обоснование выбора системы компьютерной алгебры

При реализации рассмотренных алгоритмов перед нами всталась задача выбора системы компьютерной алгебры. Наши потребности вполне укладываются в требования к универсальной системе компьютерной алгебры, но спектр таких систем весьма широк. Поэтому приведем наши критерии выбора:

- Система должна быть свободно распространяемой.
- Взаимодействие с системой должно быть итеративным. В ней должна быть реализована парадигма REPL (Read–Eval–Print Loop).
- Желательно, чтобы система поддерживалась и развивалась некоторым сообществом разработчиков. Неприятно создавать продукт на языке, который в скором времени окажется мертвым.
- Символьные вычисления — лишь один из этапов методики. Полученные уравнения необходимо исследовать чаще всего численными методами. Поэтому необходимо иметь возможность разных форматов вывода результатов. А еще лучше, иметь возможность бесшовно интегрировать искомую систему с другими программными продуктами.
- Также было бы удобно иметь реализацию численных методов в рамках системы компьютерной алгебры.

Выбор универсальных свободных систем компьютерной алгебры не так уж и велик. Рассмотрим несколько основных претендентов.

Система Maxima [14, 15] — классическая система, однако застывшая в своем развитии в конце 90-х. Новые версии выпускаются часто. При

этом они лишь увеличивают стабильность, исправляют ошибки. Добавление новых возможностей идет крайне медленно. Кроме того, возможность взаимодействия с другими программами ограничена.

Система Axiom [16, 17] выделяется математическим подходом к компьютерной алгебре. Она поддерживает систему типов Хиндли–Милнера [18, 19], обладает мощным внутренним языком расширения. Но из-за неразрешенных проблем с копирайтом систему лихорадит. Образовалось несколько ответвлений системы, при этом каждый из вариантов имеет свои планы развития. Чем и когда все это закончится — не понятно. Да и интероперабельность фактически отсутствует.

Наиболее интересной для нас является система SymPy [20, 21]. Эта система появилась как библиотека символьных вычислений для языка Python. Но язык Python стал универсальным языковым kleem (достаточно неожиданно). Применение его в разнообразных проектах привело к взрывному росту сопутствующих средств и библиотек. Поэтому и SymPy развивался вместе с ним. Теперь это достаточно мощная система компьютерной алгебры. Причем большая часть необходимых нам критериев проистекают не из собственно системы SymPy, а из окружающих ее библиотек. Получается, что SymPy удовлетворяет всем нашим критериям:

- В качестве интерактивной оболочки удобно использовать блокнот Jupyter, являющийся компонентом системы iPython [22], реализующей идеологию REPL.
- Язык Python фактически используется как соединительный язык, своего рода язык-克莱й, который позволяет интегрировать между собой разные программные продукты. Кроме того, в рамках библиотеки SciPy [23] поддерживается большое число выходных форматов.
- Выходные данные SymPy возможно естественным образом передать для численных расчетов в библиотеку NumPy [24].

Таким образом, мы остановились в своем выборе на системе SymPy для реализации метода стохастизации одноступенчатых процессов.

4.2. Программная реализации алгоритма стохастизации

Алгоритм получения стохастического дифференциального уравнения из схемы взаимодействия (алгоритмы 2 и 3) реализован как последовательность операций над векторными данными. Исходными данными являются схемы взаимодействия, представленные в следующем виде:

- символный вектор X представляет собой вектор состояния системы φ ;
- символный вектор K представляет собой интенсивности взаимодействия ${}^+k_\alpha$, ${}^-k_\alpha$ (1);
- числовые матрицы I и F представляют собой начальное и конечное состояния в формуле (4).

Основные вычисления реализованы в виде четырех функций.

Первая функция просто реализует нахождение элемента $\frac{\varphi^i!}{(\varphi^i - I^i \alpha)!}$ из формулы (7):

```
def P(x, n):
    """x -- символ, n -- целое число"""
    return sp.prod([x-i for i in range(n)])
```

Следующая функция использует предыдущую для вычисления ${}^+s_\alpha$ и ${}^-s_\alpha$, в качестве аргументов здесь передаются символьные векторы $X = (x^1, x^2, \dots, x^n)^T$ и $K = ({}^-k_1, \dots, {}^-k_s)$, а в качестве результата возвращается список ${}^+s_1, \dots, {}^+s_s$ (приведен код только для прямых реакций):

```
def S(X, K, I):
    res = []
    for i in range(len(K)):
        # Вычисляем элементы произведения
        Ps = [P(x, int(n)) for (x, n) in zip(X, I[i, :])]
        # Находим само произведение
        res.append(K[i] * sp.prod(Ps))
    # в результате получается список [s_1, s_2, s_3, ..., s_s]
    return res
```

Следующие функции — основные функции алгоритма: для получения вектора сноса `drift_vector(X, K, I, F)` и матрицы диффузии `diffusion_matrix(X, K, I, F)` в символьном формате SymPy:

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

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

Результаты работы программы можно экспортировать в формат LATEX. Можно воспользоваться встроенными методами, которые позволяют преобразовать A^i и B^{ij} в код LATEX. Для этого достаточно вызвать комбинацию функций `print(symPy.latex(A))`, где в переменной `A` содержится результат работы функции `drift_vector`, а функция `latex()` экспортирует его в LATEX-код.

При использовании интерактивной оболочки Jupyter необходимо предварительно настроить корректное отображение нотации TEX. Для этого вначале Jupyter-блокнота следует импортировать модуль `Latex`:

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

и вызвать функцию

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

Кроме того, можно экспортить получившиеся выражения в формат, соответствующий синтаксису большого спектра языков программирования. Поддерживаются как универсальные языки программирования (C/C++, Fortran), так и проблемно-специфичные (Mathematica, Julia). Данные функции доступны как методы класса `symPy.printing`. Например, результатом работы следующего кода будет C++-функция `vector<double> f(vector<double> x):`

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

4.3. Пример реализации. Модель Ферхюльста

Для демонстрации метода рассмотрим модель Ферхюльста [25, 26, 27]. В популяционной динамике эта модель описывает ограниченный рост популяции.

Детерминистическая модель имеет следующий вид:

$$\dot{\varphi} = \lambda\varphi - \beta\varphi - \gamma\varphi^2, \quad (14)$$

где λ — коэффициент интенсивности размножения, β — коэффициент интенсивности вымирания, γ — коэффициент интенсивности уменьшения популяции.

На основании (14) запишем схему взаимодействия (3):

$$\begin{array}{c} \varphi \xrightarrow[\gamma]{\lambda} 2\varphi, \\ 0 \xleftarrow{\beta} \varphi. \end{array} \quad (15)$$

Первое прямое соотношение (15) означает репродукцию особи, первое обратное соотношение — соперничество между особями. Второе соотношение описывает гибель особи.

Данная модель одномерна ($n = 1$). Количество взаимодействий s равно 2.

Из (15) запишем матрицы $I_{-\alpha}$ и $F_{-\alpha}$:

$$I_{-\alpha} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad F_{-\alpha} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

В программу передаются следующие значения:

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

Поскольку задача одномерна, то вектор сноса и матрица диффузии являются скалярами:

$$\begin{aligned} A(\varphi) &= \lambda\varphi - \beta\varphi - \gamma\varphi^2, \\ B(\varphi) &= \lambda\varphi + \beta\varphi - \gamma\varphi^2. \end{aligned}$$

Стохастическое дифференциальное уравнение, соответствующее уравнению (14), будет иметь вид:

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

Таким образом мы достигли своей цели — модель стохастизирована. Следует заметить, что даже такая простая одномерная модель при ручных расчетах является достаточно трудоемкой.

4.4. Пример реализации. Модель «хищник–жертва»

Системы, описывающие взаимодействие двух видов популяций типа «хищник–жертва», широко исследованы, и для таких систем существует большое количество разнообразных моделей. Самой первой моделью «хищник–жертва» принято считать модель, полученную независимо друг от друга А. Лоткой [28, 29] и В. Вольтеррой [30].

Детерминистическую систему можно записать в следующем виде:

$$\begin{cases} \dot{x} = k_1x - k_2xy, \\ \dot{y} = k_2xy - k_3y, \end{cases} \quad (16)$$

где x , y — взаимодействующие сущности (x — жертва, y — хищник), коэффициенты k_1, k_2, k_3 — интенсивности взаимодействия.

Введем вектор состояний $\varphi_-^i = (x, y)^T$. На основании (16) запишем схему взаимодействия (3):

$$\begin{aligned} x &\xrightarrow{k_1} 2x, \\ x + y &\xrightarrow{k_2} 2y, \\ y &\xrightarrow{k_3} 0. \end{aligned} \quad (17)$$

Схема (17) имеет стандартную интерпретацию. Первое соотношение означает, что жертва, которая съедает единицу пищи, немедленно репродуцируется. Второе соотношение описывает поглощение жертвы хищником и мгновенное репродуцирование хищника. Это единственная рассматриваемая возможность гибели жертвы. Последнее соотношение — это естественная смерть хищника.

Для данной модели размерность системы n равна 2, количество взаимодействий s равно 3.

Из (17) запишем матрицы $I_{-\alpha}$ и $F_{-\alpha}$:

$$I_{-\alpha} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad F_{-\alpha} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix}.$$

В программу передаются следующие значения:

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

В результате будут вычислены вектор сноса и матрица диффузии

$$A^i(x, y) = \begin{pmatrix} k_1x - k_2xy \\ k_2xy - k_3y \end{pmatrix},$$

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

Эти соотношения позволяют записать непосредственно уравнение Фоккера–Планка и соответствующее ему стохастическое дифференциальное уравнение. Для этого достаточно извлечь квадратный корень из матрицы B^{ij} :

$$\begin{aligned} d\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} k_1x - k_2xy \\ k_2xy - k_3y \end{pmatrix} dt + b_a^i \begin{pmatrix} dW^1 \\ dW^2 \end{pmatrix}, \\ b_a^i b_a^j a = B^{ij} &= \begin{pmatrix} k_1x + k_2xy & -k_2xy \\ -k_2xy & k_2xy + k_3y \end{pmatrix}. \end{aligned} \quad (18)$$

Отсюда видно, что детерминистическая часть системы (18) совпадает с исходной системой (16). Поскольку операция извлечения корня из матрицы в аналитическом виде представляет из себя достаточно сложную задачу, исследование этой системы оптимальнее проводить численно.

5. ЗАКЛЮЧЕНИЕ

Исследовательская работа часто носит итеративный характер. Результаты вычислений оцениваются по определенным критериям. Эти действия приходится выполнять многократно. Системы компьютерной алгебры позволяют автоматизировать этот процесс.

В данной работе рассмотрена простейшая реализация алгоритма стохастизации одношаговых процессов по заданным схемам взаимодействия. Для этого был проведен анализ систем компьютерной алгебры на основе предложенных авторами критериев. В соответствии с этими критериями в качестве системы компьютерной алгебры для реализации метода выбрана система *SymPy*. Приведены существенные элементы кода реализации метода стохастизации одношаговых процессов. Работа программного комплекса продемонстрирована на примере двух моделей: модели Ферхольста и модели Лотки–Вольтерра.

При всей своей кажущейся простоте данный программный продукт резко увеличивает производительность труда исследователя. Можно сделать вывод, что системы компьютерной алгебры,

наряду с системами численных расчетов, стали необходимым инструментом исследователя.

СПИСОК ЛИТЕРАТУРЫ

- Гардинер К. В. Стохастические методы в естественных науках. Мир, 1986.
- Ван-Кампен Н. Г. Стохастические процессы в физике и химии. М. : Высшая школа, 1990.
- Korolkova A. V., Eferina E. G., Laneev E. B., Gudkova I. A., Sevastianov L. A., Kulyabov D. S. Stochasticization Of One-Step Processes In The Occupations Number Representation // Proceedings 30th European Conference on Modelling and Simulation. 2016, Jun., pp. 698–704.
- Eferina E. G., Hnatich M., Korolkova A. V., Kulyabov D. S., Sevastianov L. A., Velieva T. R. Diagram Representation for the Stochasticization of Single-Step Processes // Distributed Computer and Communication Networks. DCCN 2016. Communications in Computer and Information Science / Ed. by V. M. Vishnevskiy, K. E. Samouylov, D. V. Kozyrev. Cham: Springer, 2016, vol. 678, pp. 483–497.
- Hnatič M., Eferina E. G., Korolkova A. V., Kulyabov D. S., Sevastyanov L. A. Operator Approach to the Master Equation for the One-Step Process // EPJ Web of Conferences. 2015, vol. 108, pp. 58–59. arXiv: 1603.02205.
- Gevorkyan M. N., Demidova A. V., Zaryadov I. S., Sobolewski R., Korolkova A. V., Kulyabov D. S., Sevastianov L. A. Approaches to Stochastic Modeling of Wind Turbines // Proceedings 31st European Conference on Modelling and Simulation, ECMS 2017 / Ed. by K. Varadi, A. Vidovics-Dancs, J. P. Radics, Z. Z. Paprika, P. T. Zwierczyk, P. Horak. Budapest: European Council for Modelling and Simulation, 2017, May, pp. 622–627.
- Gevorkyan M. N., Velieva T. R., Korolkova A. V., Kulyabov D. S., Sevastyanov L. A. Stochastic Runge–Kutta Software Package for Stochastic Differential Equations // Dependability Engineering and Complex Systems. Springer International Publishing, 2016, vol. 470, pp. 169–179, arXiv: 1606.06604.
- Grassberger P., Scheunert M. Fock-Space Methods for Identical Classical Objects // Fortschritte der Physik. 1980, vol. 28, no. 10. pp. 547–578.

9. Täuber U. C. Field-Theory Approaches to Nonequilibrium Dynamics // Ageing and the Glass Transition. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, vol. 716. pp. 295–348. arXiv: 0511743.
10. Janssen H.-K., Täuber U. C. The field theory approach to percolation processes // *Annals of Physics*. 2005, Jan., vol. 315, no. 1, pp. 147–192, arXiv: 0409670.
11. Mobilia M., Georgiev I. T., Täuber U. C. Fluctuations and correlations in lattice models for predator-prey interaction // *Physical Review E*. — 2006., Apr., vol. 73, no. 4, pp. 040903. arXiv: 0508043.
12. Пенроуз Р., Риндлер В. Спиноры и пространство-время. Два-спинорное исчисление и релятивистские поля. Москва: Мир, 1987, Т. 1. С. 527.
13. Demidova A. V., Korolkova A. V., Kulyabov D. S., Sevastyanov L. A. The Method of Constructing Models of Peer to Peer Protocols // 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE Computer Society, 2015, pp. 557–562. arXiv: 1504.00576.
14. Bainov D. D., Hristova S. G. Differential Equations with Maxima. Pure and Applied Mathematics. Chapman and Hall/CRC, 2011, Apr., pp. x+291.
15. Timberlake T. K., Mixon J. W. Classical Mechanics with Maxima. Undergraduate Lecture Notes in Physics. New York, NY: Springer New York, 2016, pp. xi+258.
16. Jenks R. D., Sutor R. S. AXIOM: The Scientific Computation System. 1992, pp. xxiv + 742.
17. Eferina E. G., Korolkova A. V., Gevorkyan M. N., Kulyabov D. S., Sevastyanov L. A. One-Step Stochastic Processes Simulation Software Package // *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*. 2014, no. 3, pp. 46–59, arXiv: 1503.07342.
18. Hindley R. The Principal Type-Scheme of an Object in Combinatory Logic // *Transactions of the American Mathematical Society*. 1969, vol. 146, no. December. pp. 29–29.
19. Milner R. A Theory of Type Polymorphism in Programming // *Journal of Computer and System Sciences*. 1978, Dec., vol. 17, no. 3, pp. 348–375.
20. Lamy R. Instant SymPy Starter. Packt Publishing, 2013, pp. 52.
21. Eferina E. G., Kulyabov D. S. Implementation of Diagram Technique for Statistical Systems in Sympy // 6th International conference “Problems of Mathematical Phisics and Mathematical Modelling”. Moscow : NRNU MEPhI, 2017, May, pp. 125–127.
22. Perez F., Granger B. E. IPython: A System for Interactive Scientific Computing // *Computing in Science & Engineering*. 2007, vol. 9, no. 3, pp. 21–29.
23. Oliphant T. E. Python for Scientific Computing // *Computing in Science & Engineering*. 2007, vol. 9, no. 3, pp. 10–20.
24. Oliphant T. E. Guide to NumPy. 2 edition edition. CreateSpace Independent Publishing Platform, 2015, pp. 364.
25. Verhulst P. F. Notice sur la loi que la population suit dans son accroissement. 1838, vol. 10, pp. 113–117.
26. Feller W. Die Grundlagen der Volterraischen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung // *Acta Biotheoretica*. 1939, Bd. 5, H. 1. pp. 11–40.
27. Feller W. On the theory of stochastic processes, with particular reference to applications // *Proceedings of the [First] Berkeley Symposium on Mathematical Statistics and Probability*. 1949, pp. 403–432.
28. Lotka A. J. Contribution to the Theory of Periodic Reaction // *The Journal of Physical Chemistry A*. 1910, vol. 14, no. 3, pp. 271–274.
29. Lotka A. J. Elements of Physical Biology // *Williams and Wilkins Company*. 1925, pp. 435.
30. Volterra V. Variations and fluctuations of the number of individuals in animal species living together // *Journal du Conseil permanent International pour l'Exploration de la Mer*. 1928, vol. 3, no. 1, pp. 3–51.