МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)

Институт информационных технологий, математики и механики

Кафедра: высокопроизводительных вычислений и системного программирования

Направление подготовки: «Прикладная математика и информатика»
Магистерская программа: «Вычислительные методы и суперкомпьютерные технологии»

ОТЧЕТ

по четвёртой лабораторной работе

на тему:

«Численное моделирование случайных процессов»

Выполнил: студент
группы 3824М1ПМвм
Ивлев А.Д.

Нижний Новгород
2025

Оглавление

1. Введение

Моделирование случайных процессов используется для создания последовательности состояний, которые статистически похожи на поведение реального случайного процесса во времени.

Численный подход к моделированию особо полезен, когда невозможно или крайне трудно найти точное решение уравнений случайного процесса. Так как данный подход даёт статистически похожее решение, то он позволяет качественно оценить динамику процесса, но в отдельные моменты времени погрешность может быть достаточно велика. Поэтому, для уточнения решения, может потребоваться множество запусков методов моделирования случайных процессов.

Пусть далее для описания случайных процессов примем обозначения: $x \in \mathbb{R}^N$ - состояние системы, $c_i \in \mathbb{R}^N = const$ изменение состояния после одного из возможных случайных переходов. Тогда вероятность изменения системы с данным переходом за отрезок времени Δt: $Prob(x(\Delta t) = X + c_i | x(0) = X) = a_i(X)\Delta t$, где $a_i(x) \geq 0$ – функция склонности к i-му переходу в зависимости от состояния. Данный подход к описанию случайного процесса допустим, только при условии $\forall\, i$: $a_i(X)\Delta t \ll 1$.

2. Постановка задачи

В данный работе будут рассмотрены два метода моделирования случайных процессов. Они будут протестированы на ДУ, задающих некоторые случайные процессы химической генетики и молекулярной динамики. Также будет проведено сравнение времени работы методов.

В качестве первого из исследуемых случайных процессов будет рассмотрена модель авторепрессора, которая задаётся нелинейным автономным ДУ 1-го порядка:

$$\dot{x} = \frac{\alpha}{1+x^n} - x \qquad (1)$$

Состояние системы x одномерно. У данной системы два возможных случайных перехода, которые описываются значениями: $a_1(x) = \frac{\alpha}{1+x^n}$, $c_1 = 1$, $a_2(x) = x$, $c_2 = -1$.

В качестве второго из исследуемых случайных процессов будет рассмотрена модель, полученная из модели авторепрессора путём выделения этапов транскрипции и трансляции. Данная модель задаётся системой из двух нелинейных автономных ДУ 1-го порядка:

$$\begin{cases} \dot{y} = \beta m - \gamma y \\ \dot{m} = \frac{\alpha}{1+y^n} - m \end{cases} \qquad (2)$$

Состояние системы $x = (y,\ m)^T \in \mathbb{R}^2$. У данной системы четыре возможных случайных перехода, которые описываются значениями: $a_1(x) = \beta m,\ c_1 = (1,0)^T$, $a_2(x) = \gamma y$, $c_2 = (-1,0)^T$, $a_3(x) = \frac{\alpha}{1+y^n}$, $c_3 = (0,1)^T$, $a_4(x) = m$, $c_4 = (0,-1)^T$.

- Реализовать метод Монте-Карло и метод Гиллеспи для моделирования случайных процессов
- Показать работоспособность методов на указанных примерах (1), (2)
- Сравнить скорость работы данных методов для процесса (2)

3. Численные методы моделирования случайных процессов

3.1Метод Монте-Карло

Данный метод основан на случайном выборе одного из переходов, который произойдёт за отрезок времени Δt на основе их вероятностей.

Условия применимости:

- Шаг моделирования по времени Δt достаточно мал, что $\forall\, x, i$ выполняется условие $a_i(x)\Delta t \ll 1$

Алгоритм

- **Инициализация**:
  - Задать начальное состояние $x(t=0) = X \in \mathbb{R}^N$, шаг моделирования по времени Δt и время моделирования T
  - Положим предельные значения $R_0(x) = 0$, $R_{k+1}(x) = 1$ и переход $c_{k+1} = 0 \in \mathbb{R}^N$, который обозначает, что ничего не произошло
- **Итерационный процесс**:
  - Выбрать случайно равномерно распределённое r на отрезке [0, 1]
  - Подсчитать предельные значения $R_i(x) = \sum_{j=1}^{i} a_i(x)\Delta t,\ i = \overline{1,k}$
  - Если $R_{i-1}(x) \le r < R_i(x)$, тогда произошёл i-й переход и новое состояние $x = x + c_i$
  - Повторять до достижения критерия остановки
- **Критерий остановки**:
  - Время моделирование $t \le T$

Плюсы:

- Простота реализации

Минусы:

- Необходимость выбора шага моделирования, удовлетворяющего условиям применимости.
- Количество итераций метода зависит только от выбранных шага и времени моделирования и никак ни учитывает поведение системы, поэтому оно может быть достаточно велико.
- Дискретность времени переходов.

Реализация

```
1 def method1(x0, a, dt, N):
2     res = []
```

```
    x = x0[:]
    for i in range(N):
        x1 = x[:]
        for k in range(len(x0)):
            r = random.random()
            tmp_prob = 0
            for j in range(len(a[k])):
                tmp_prob += prob(a[k][j][0](x1), dt)
                if (r < tmp_prob):
                    x[k]+=a[k][j][1]
                    break
        res.append(x[:])
    return res
```

Где x0 – начальное состояние, a – список содержащий $a_i(x)$ и c_i, dt – шаг моделирования, N – количество итераций. Возвращает список состояний в просчитанные моменты времени.

3.2Метод Гиллеспи

Данный метод основан на оценке времени перехода системы в один из исходов, на основе их склонностей к переходу $a_i(x)$.

Алгоритм

- **Инициализация**:
  - Задать начальное состояние $x(t = 0) = X \in \mathbb{R}^N$ и время моделирования T
  - Положим предельное значения $R_0(x) = 0$
- **Итерационный процесс**:
  - Выбрать случайно равномерно распределённые r_1 и r_2 на отрезке [0, 1]
  - Подсчитать $a_0(x) = \sum_i a_i(x)$
  - Подсчитать время случайного перехода $\tau = \frac{1}{a_0(x)} \ln\left(\frac{1}{r_1}\right)$
  - Подсчитать предельные значения $R_i(x) = \sum_{j=1}^{i} \frac{a_i(x)}{a_0(x)}$, $i = \overline{1,k}$
  - Если $R_{i-1}(x) \le r_2 < R_i(x)$, тогда произошёл i-й переход и новое состояние $x = x + c_i$
  - Следующий момент времени $t = t + \tau$
  - Повторять до достижения критерия остановки
- **Критерий остановки**:
  - Время моделирование $t \le T$

Плюсы:

- Простота реализации
- Нет условий применимости
- Непрерывность (случайные моменты) времени переходов

Минусы:

- Количество итераций метода зависит от поведения системы, что хорошо для простых системы, но плохо для более сложных

Реализация

```
def method2(x0, a, T):
    x = x0[:]
    t = 0
    res = [[x0, 0]]
    while (t < T):
        x1 = x[:]
        a0 = 0
        for k in range(len(x0)):
            for j in range(len(a[k])):
                a0 += a[k][j][0](x1)
        if(a0 != 0):
            tau = 1/a0*np.log(1/random.random())
        else:
            tau = T
        tmp_prob = 0
        r = random.random()
        flag = False
        for k in range(len(x0)):
            if(flag):
                break
            for j in range(len(a[k])):
                if(a0 != 0):
                    tmp_prob += a[k][j][0](x)/a0
                else:
                    tmp_prob = 1
                if (r < tmp_prob):
                    x[k]+=a[k][j][1]
                    flag = True
                    break
        t += tau
        res.append([x[:], t])
    if(res[-1][-1] > T):
        res.pop()
    return res
```

Где x0 – начальное состояние, a – список содержащий $a_i(x)$ и c_i, T – время моделирования. Возвращает список пар состояние, момент времени.

4. Результаты экспериментов

Программная реализация выполнялась на языке python. Полный код доступен по ссылке: https://github.com/Faert/NLD_Lab.

Сначала рассмотрим простой пример (1) с $\alpha = 0$. Тогда для системы известно точное решение $x(t) = x(0)e^{-t}$.

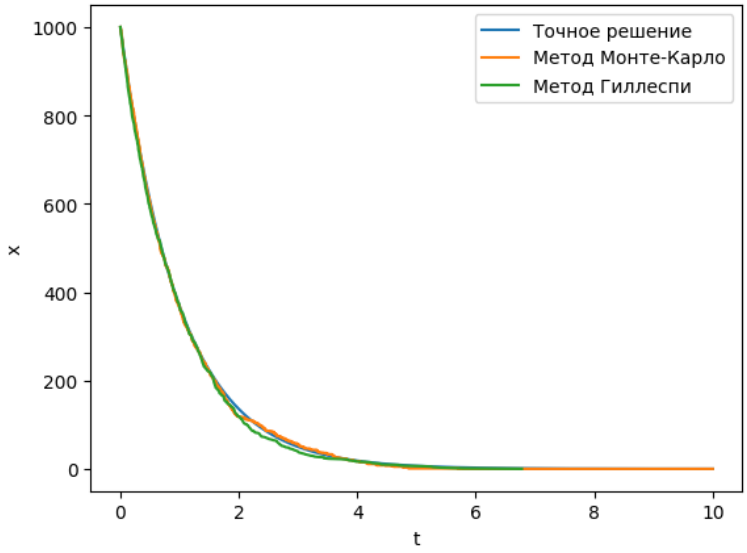

Рисунок 1. Динамика состояния системы (1) с $\alpha = 0. x(0) = 1000, T = 10, \Delta t = 10^{-4}$

Можно заметить, что решения данными методами действительно дают статистически точное решение. При множественных запусках методов выполняется: $M\big(x(t)\big) \approx x(0)e^{-t}$.

Для дальнейшего исследования зафиксируем параметры исследуемых примеров (1) и (2). Пусть $\alpha = 200$, n $= 6$. $\beta = \gamma = 1$. Так же, так как точное решение неизвестно, в качестве эталонного решение взято решение, полученное методом Рунге-Кутта 4. Зафиксируем $\Delta t = 10^{-4}, T = 10, x(0) = 1000$ для (1) и $x(0) = (1000, 1000)^T$ для (2).

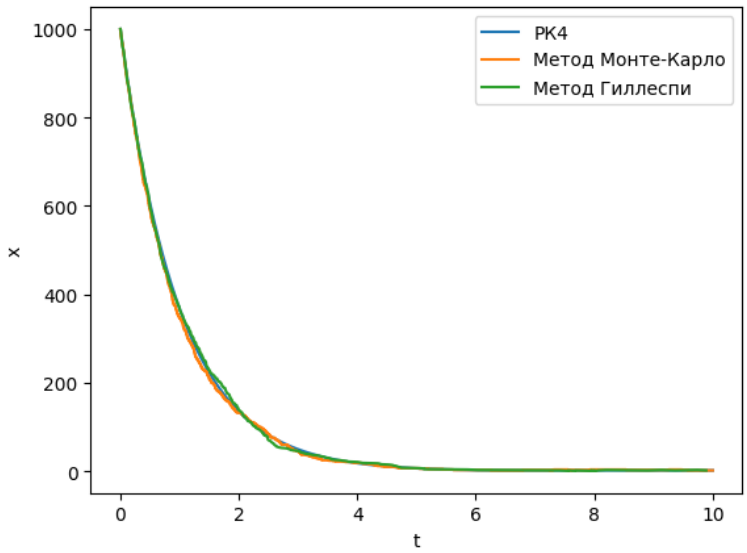

Рисунок 2. Динамика состояния системы (1)

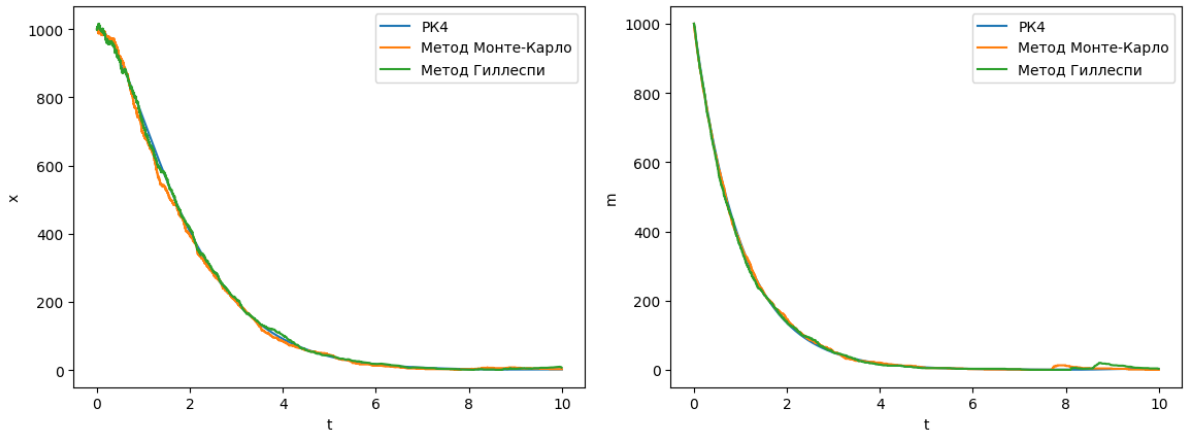

Рисунок 3. Динамика компонент состояния $x = (y,\ m)^T$ системы (2)

Для систем (1) и (2) решения методами Монте-Карло и Гиллеспи дали решение приближенное к решению, найденному методом Рунге-Кутта 4.

Теперь сравним время работы данных методов в зависимости от начального состояния для (2). Для этого зафиксируем $T = 1000$, а начальные состояния возьмём из набора {8, 16, 32, 64, 128, 256, 512, 1024} (y(0) = m(0)).

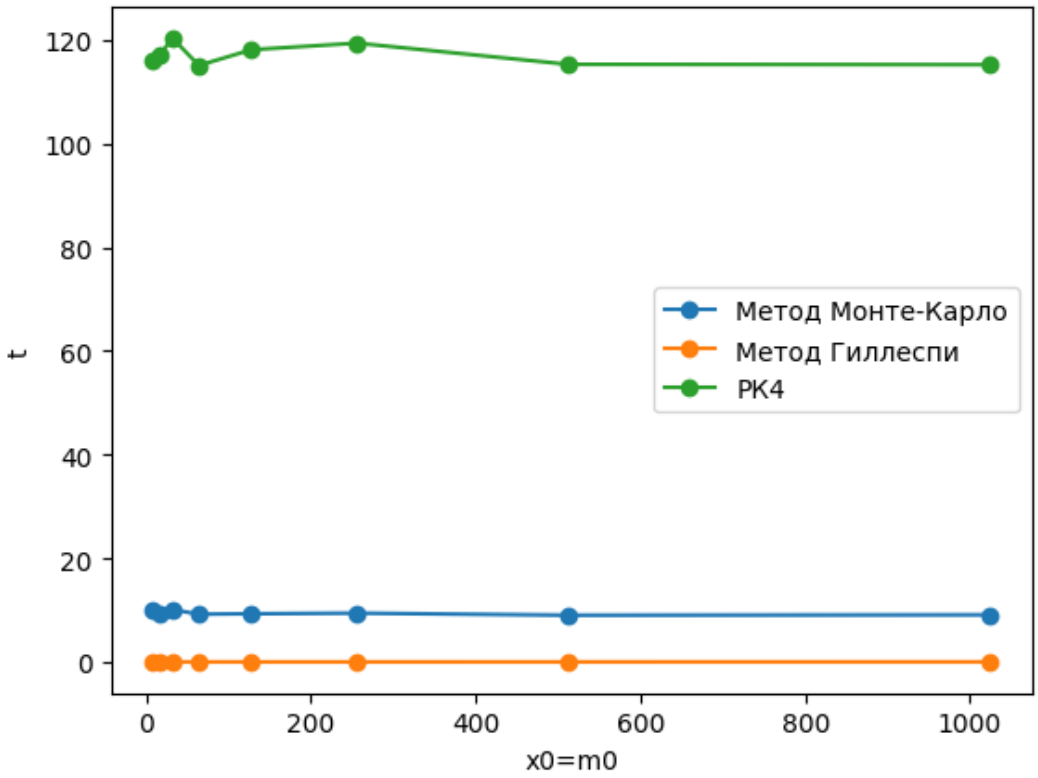

Рисунок 4. Время работы методов в зависимости от начального состояния

Для всех методов не выявлено зависимости времени выполнения от начального состояния. Среднее время выполнение для метода Гиллеспи составило 0.036 секунд, что на порядки меньше, чем среднее время для метода Монте-Карло, которое составило 9.485. Также на график добавлен метод Рунге-Кутта, чтобы показать, что моделирование случайных процессов быстрее, чем нахождение приближённого решения систем ДУ их задающих.

Теперь сравним время работы данных методов в зависимости от времени моделирования для (2). Для этого зафиксируем $y(0) = m(0) = 1000$, а времени моделирования возьмём из набора {8, 16, 32, 64, 128, 256, 512, 1024}.

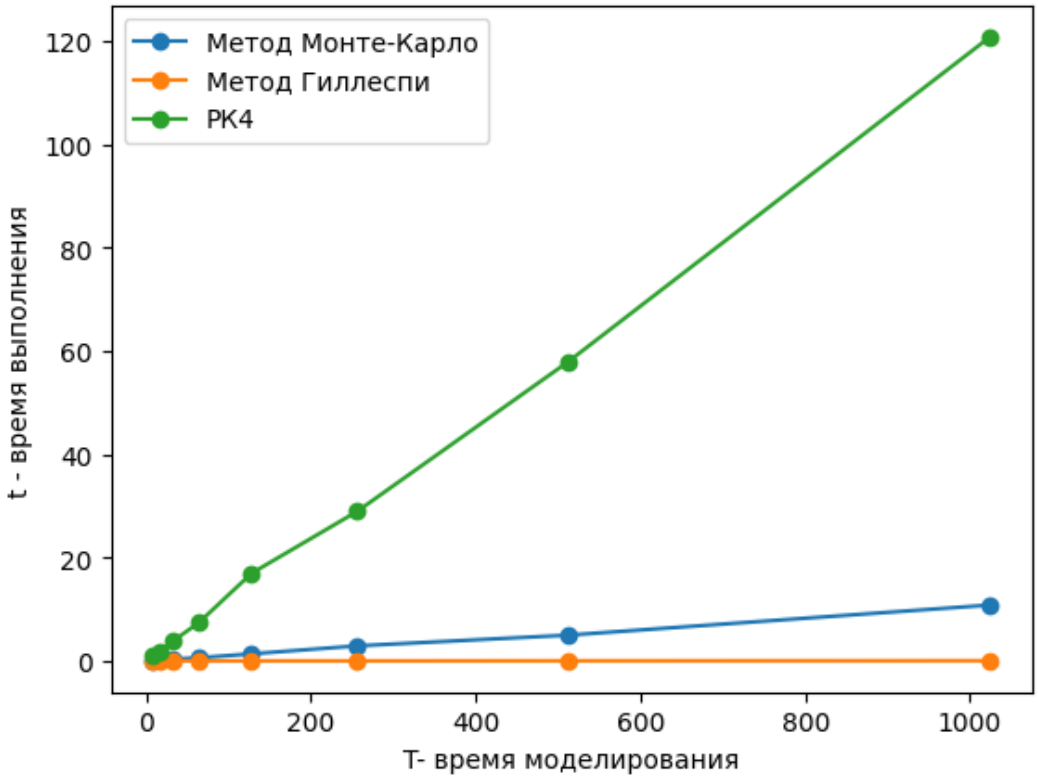

Рисунок 5. Время работы методов в зависимости от времени моделирования

При увеличении в 2 раза времени моделирования T время выполнения методов Монте-Карло и Рунге-Кутта 4 также увеличивалось примерно в 2 раза, так как в 2 раза увеличивалось количество их итераций. Метод Гиллеспи почти не зависит от увеличения времени моделирования, так как состояние системы (2), начиная с некоторого момента, почти не меняется.

5. Заключение

В рамках данной работы были реализованы методы моделирования случайных процессов Монте-Карло и Гиллеспи. На примерах (1) и (2) были протестирована их работа. Данные методы действительно дают статистически точное решение.

Было произведено сравнение времени работы данных методов. Метод Гиллеспи дал результат лучше, чем метод Монте-Карло, так как количество его итераций учитывает от поведение системы.