|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
|  |  |
| ФАКУЛЬТЕТ | Аэрокосмический |
|  |  |
| КАФЕДРА | Вычислительная математика и математическая физика |

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

|  |  |
| --- | --- |
| Студент | Клонов Александр Сергеевич |
|  | *фамилия, имя, отчество* |
|  |  |
| Группа | АК3-61Б |
|  |  |
| Тип практики | учебная |
|  |  |
| Название  предприятия | МГТУ им. Н.Э. Баумана |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Клонов А.С. |
|  | *подпись, дата* |  | *фамилия, и.о.* |
|  |  |  |  |
| Руководитель практики |  |  | Краснов И.К. |
|  | *подпись, дата* |  | *фамилия, и.о.* |

|  |  |
| --- | --- |
| Оценка |  |

|  |
| --- |
| *2022 г.* |

Оглавление

[Теоретическая часть 3](#_Toc104043736)

[Метод Монте-Карло 3](#_Toc104043737)

[Уравнение теплопроводности 6](#_Toc104043738)

[Практическая часть 8](#_Toc104043739)

[Задача о невыходе траектории случайного поля за границы данной области методом Монте-Карло 8](#_Toc104043740)

[Результаты 9](#_Toc104043741)

[Вывод 14](#_Toc104043742)

[Приложение. Список источников 15](#_Toc104043743)

[Приложение. Код программы 16](#_Toc104043744)

# Теоретическая часть

## Метод Монте-Карло

Методы Монте-Карло (ММК) — группа численных методов для изучения случайных процессов. Суть метода заключается в следующем: процесс описывается математической моделью с использованием генератора случайных величин, модель многократно обсчитывается, на основе полученных данных вычисляются вероятностные характеристики рассматриваемого процесса. Например, чтобы узнать методом Монте-Карло, какое в среднем будет расстояние между двумя случайными точками в круге, нужно взять координаты большого числа случайных пар точек в границах заданной окружности, для каждой пары вычислить расстояние, а потом для них посчитать среднее арифметическое.

Методы используются для решения задач в различных областях физики, химии, математики, экономики, оптимизации, теории управления и др.

Имитационное моделирование по методу Монте-Карло представляет собой определение математического ожидания (среднего значения случайной величины) путем проведения определенного количества симуляций (испытаний).

Предположим, требуется найти математическое ожидание α для случайной величины: 



Классическая формула расчета математического ожидания выглядит так:

, где:

- значение величины от 1 до n;

 - вероятность от 1 до n.

Моделирование методом Монте-Карло выполняется следующим образом: проводится n симуляций (испытаний). В результате получится какое-то количество значений X. Далее определяется их среднее арифметическое, которое и будет приблизительным значением α.

Метод Монте-Карло относится к методам моделирования различных явлений, событий, параметров или процессов, как благоприятных, так и неблагоприятных, с целью определения вероятности их наступления. Для этого генерируется определенное количество случайных величин, отвечающих установленным критериям, а затем на их основе вычисляют приблизительное значение искомой величины.

ММК применяется в следующих областях:

1. Физика, химия, биология – для моделирования различных явлений.
2. Экономика и финансы – для оценки и прогнозирования инвестиций, расчета доходности финансовых инструментов, сроков окупаемости и др. Метод Монте-Карло широко применяется для оценки рисков;
3. Игровая индустрия – для моделирования искусственного интеллекта и др.
4. Технология и др. инженерные науки используют метод Монте-Карло в прогнозировании НТП.
5. Социология – для изучения общественного мнения (люди, принимающие участие в опросах, отбираются в случайном порядке).
6. По сути, методу можно найти применение во многих сферах, где необходимы расчеты и прогнозирование.

Имитационное моделирование методом Монте-Карло – это автоматизированный процесс, позволяющий рассматривать вероятность наступления различных событий. Каждая смоделированная ситуация является уникальной, что дает возможность оценить целый спектр рисков.

Далее система распределяет вероятности. Для оценки различных параметров применяются варианты распределения:

* Нормальное распределение. Кривая нормального распределения или Гауссова кривая, выглядит так:

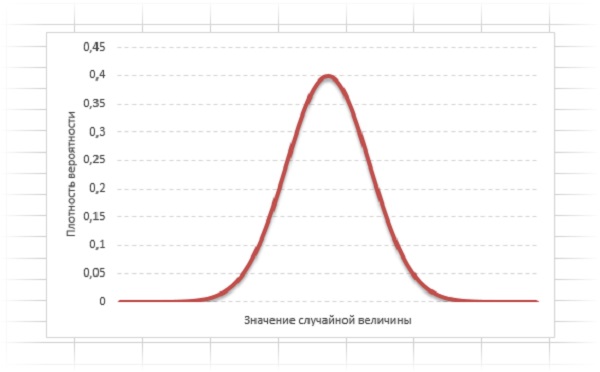


Рис1. График плотности случайного распределения

* Равномерное распределение. Все события могут наступить с одинаковой вероятностью, пользователю требуется лишь установить минимум и максимум.

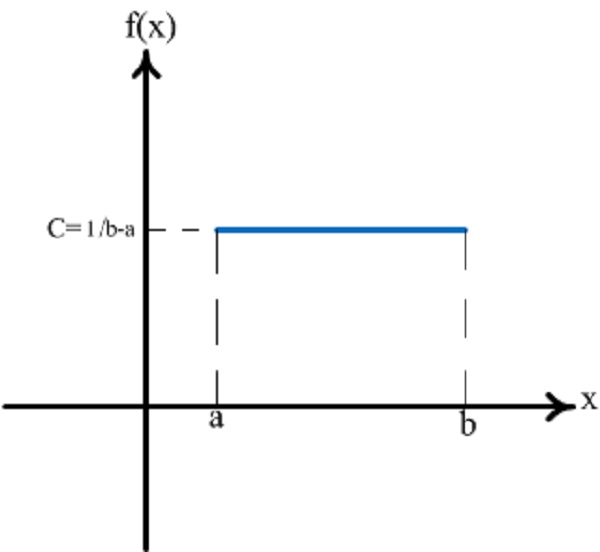


Рис2. График равномерного распределения

* Логнормальное распределение. Используется для расчета параметров, которые не могут принимать отрицательное значение, но могут расти до бесконечности.
* Дискретное распределение. Может применяться, к примеру, для компьютерных игр, где вычисляется процент побед и поражений. Пользователю требуется вычислить нужные значения из числа возможных.

## Уравнение теплопроводности

Уравнение теплопроводности — дифференциальное уравнение в частных производных второго порядка, которое описывает распределение температуры в заданной области пространства и ее изменение во времени.

В пространстве с произвольной системой координат  уравнение теплопроводности имеет вид:

 (1)

где a — положительная константа (число является коэффициентом температуропроводности),— оператор Лапласа и  — функция тепловых источников. Искомая функция  задает температуру в точке с координатами r в момент времени t.

Данное уравнение можно объяснить следующим образом. Скорость изменения температуры во времени пропорциональна кривизне распределения температуры по пространству (второй производной). Иными словами, чем выше кривизна "горбов" температуры в теле, тем быстрее в этих местах идёт выравнивание температуры.

В пространстве с декартовыми координатами  уравнение теплопроводности принимает вид:

 (2)

Уравнение теплопроводности называется однородным, если , т.е. внутри системы нет источников и "стоков" тепла.

# Практическая часть

## Задача о невыходе траектории случайного поля за границы данной области методом Монте-Карло

Постановка задачи:

Необходимо смоделировать однородное уравнение теплопроводности, где краевое условие или начальная функция имеет случайное распределение и найти такую границу, при которой вероятность попадания всех значений данного случайного процесса будет равна 90%.

Алгоритм:

1. Устанавливаем количество всех опытов N, и заводим счетчик k выполненных условий;
2. Генерируем случайное число с нормальным распределением;
3. Решаем уравнение теплопроводности с коэффициентом, полученным из п.1 – для этого воспользуемся алгоритмом построения явной сетки. Он заключается в следующем: разобьем отрезок  на n узлов с шагом h. Так как будем использовать явную сетку, необходимо выбрать такой шаг по времени , чтобы он удовлетворял критерию устойчивости сетки. Критерий устойчивости явной сетки для уравнений теплопроводности выглядит следующим образом: . Значения частных производных запишем следующим образом:





Где уравнение (1.1) соответствует частной производной по времени, а (1.2) частной производной второго порядка по координате. Тогда уравнение будет иметь следующий вид:



Значения для следующего временного слоя  будем находить по значениям функции в предыдущем слое:



1. Проверяем условие невыхода, если хоть одно значение в узле сетки будет превышать по модулю значение установленной границы, то условие не выполняется, иначе инкрементируем счетчик;
2. Выполняем п.2-п.4 столько раз, сколько было установлено в п.1;
3. Считаем вероятность невыхода как , в случае если она меньше допустимой 0.9, то увеличиваем значение границы с некоторым шагом и повторяем весь алгоритм.

## Результаты

Результаты программы для следующих параметров:

Начальное условие: 

Краевые условия: , где число распределенное по нормальному закону ;

L = 1, t = 1, a = 1, число опытов = 1000, начальная граница = 1.5, шаг границы = 0.05. Шаг h = 0.1, шаг 

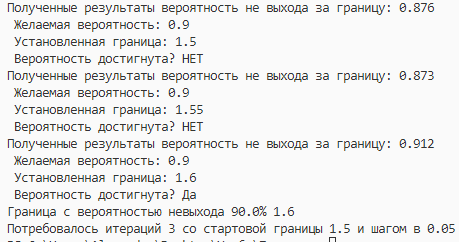


Рис3 – результат выполнения программы с заданными параметрами

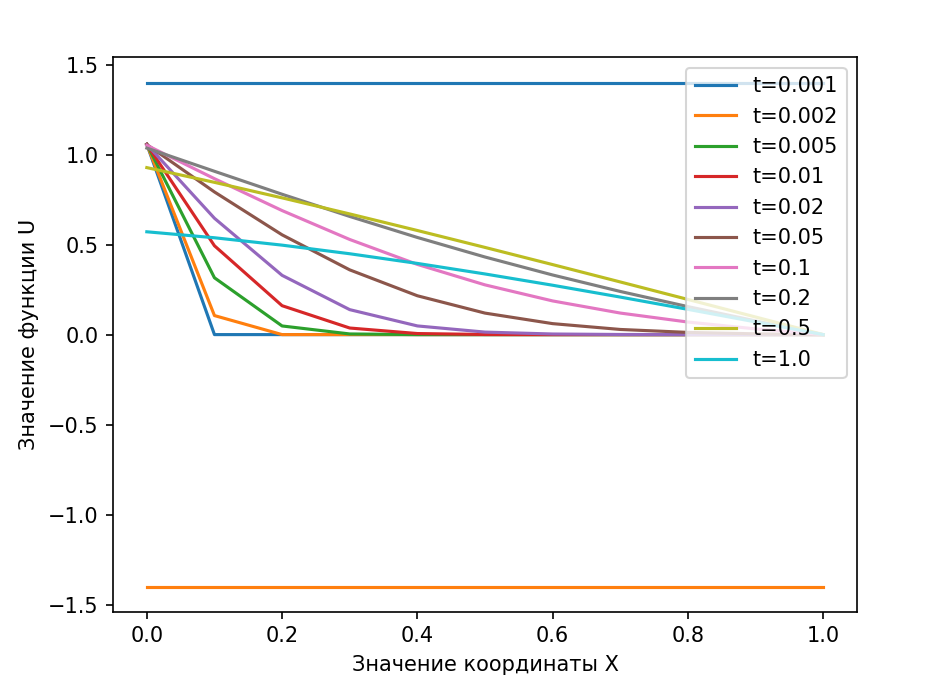


Рис.4 – График одного уравнения, значения которого не вышли за пределы границы.

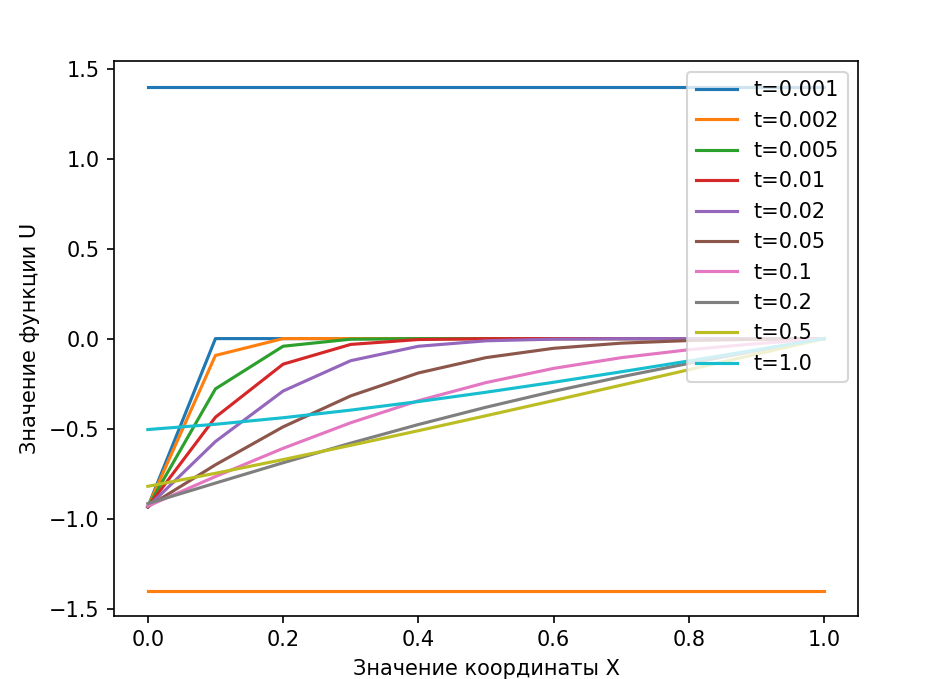


Рис.5 – График еще одного уравнения, значения которого не вышли за пределы границы.

Для сравнения, решим детерминированную задачу, где краевое условие на левом конце будет следующим:



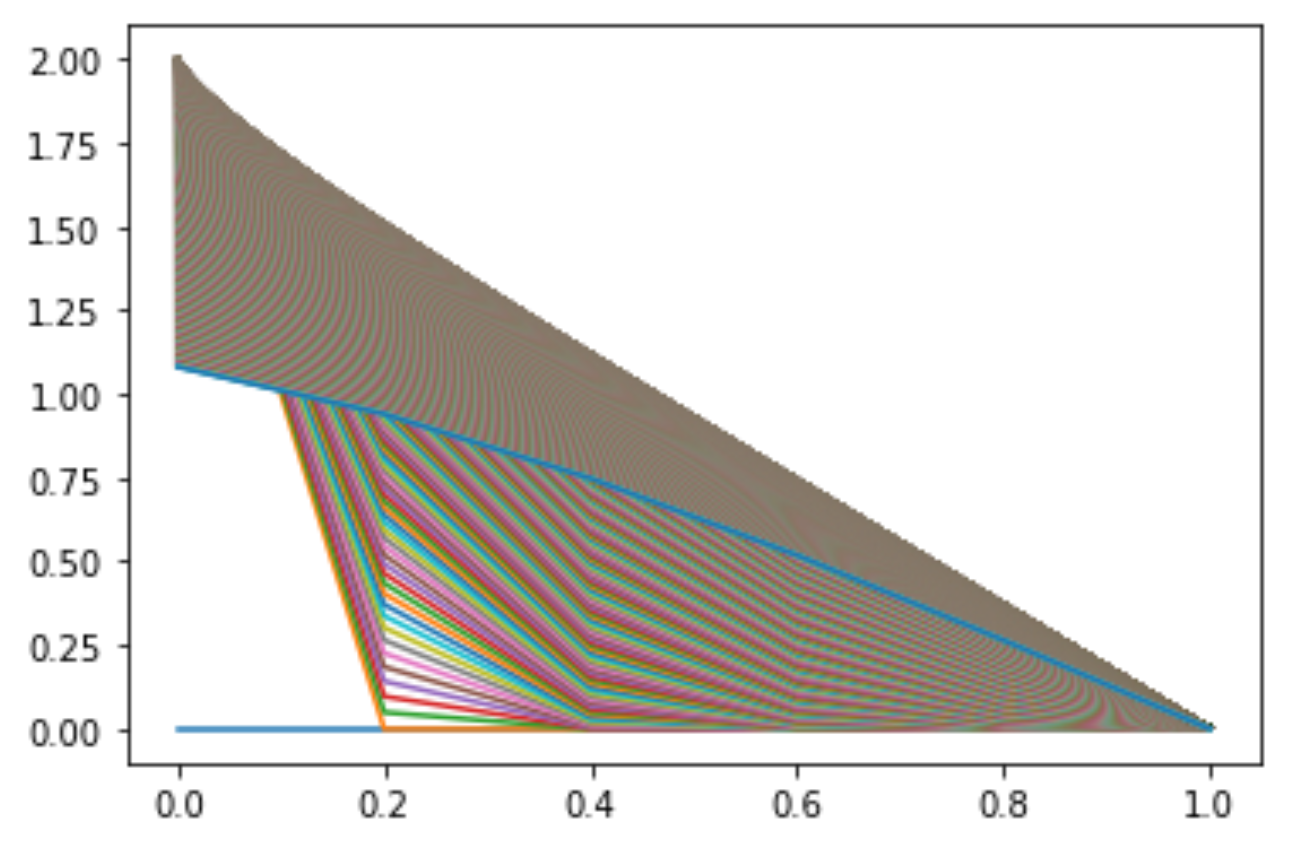


Рис.6 – График детерминированной задачи

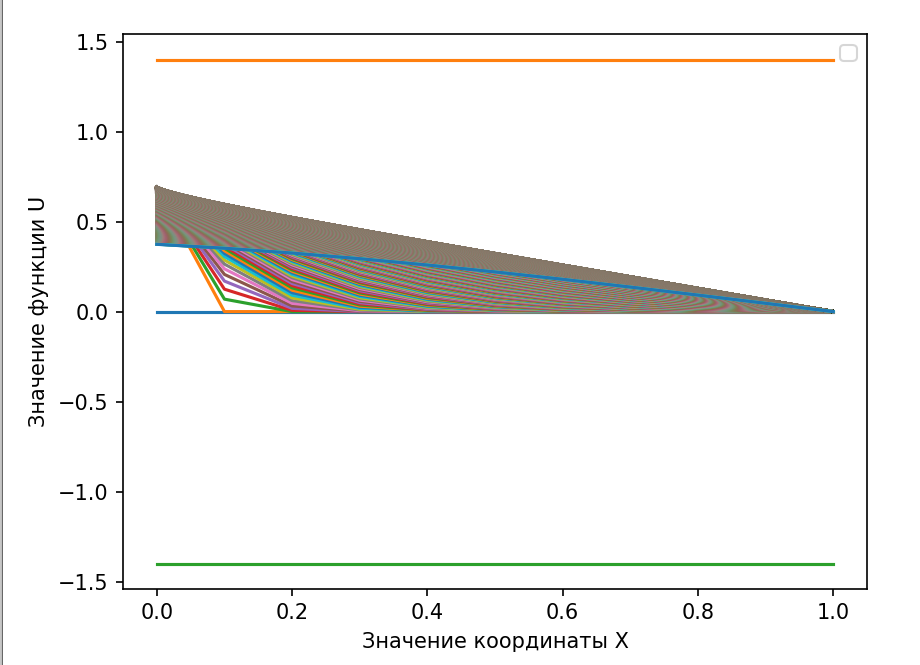


Рис.7 – График уравнения для всех моментов времени

Поведение графиков на отрезке [0, L] совпадает, различие наблюдается на левом конце в виду различия в краевом условии.

Уменьшим число опытов для отображения результата в графическом виде:

N = 10

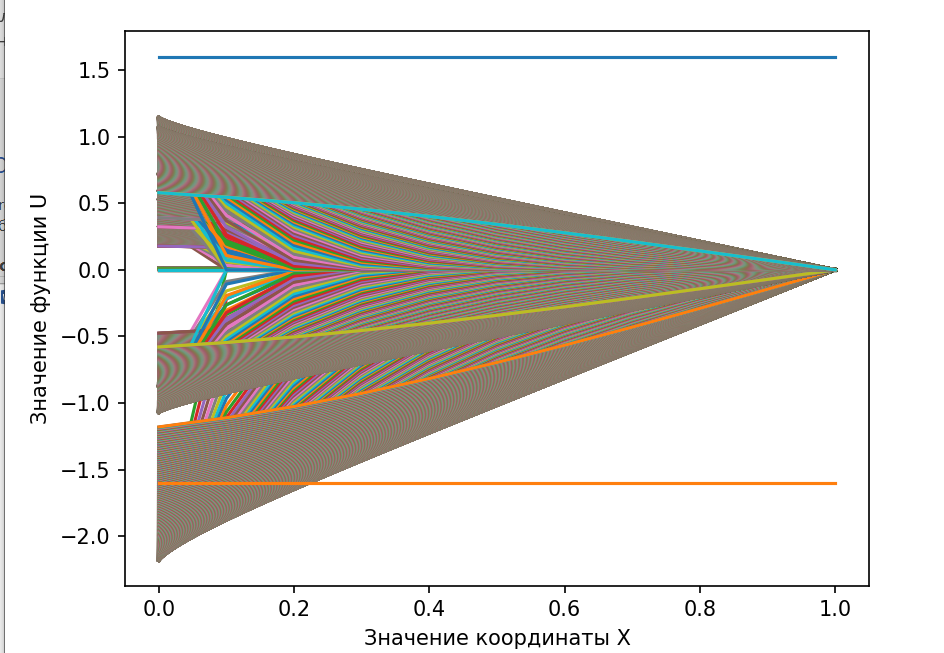


Рис.8 – Графики уравнений для всех моментов времени при N=10.

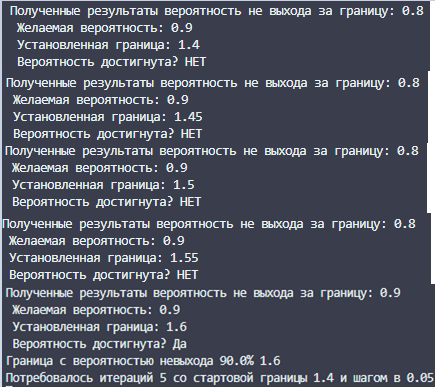


Рис.9 – результат выполнения программы при N=10

На рисунках 8 и 9 заметно, что значения одного из 10 графиков вышли за пределы допустимой границы, но так как такой график был только один и ввиду небольшой выборке (N=10), желаемая вероятность была достигнута.

## Вывод

Была реализована программа для решения задачи о невыходе траектории случайного поля за границы данной области методом Монте-Карло. Программа позволяет задать различные параметры такие как условия самой функции вроде шага сетки или областей функции, так и параметры самого эксперимента – начальную границу, шаг границы, число экспериментов в серии, желаемую вероятность невыхода. Также была решена детерминированная задача, ее результаты были представлены в виде эталонного графика, с которым сравнивались результаты для задач со случайными параметрами.

Увеличение числа опытов, а также уменьшение шага границы позволяет более точно определить границу для желаемой вероятности.

Стоит также добавить, что каждая итерация является достаточно ресурсоемкой задачей, поэтому для ускорения расчетов был реализован параллельный алгоритм, где каждый эксперимент считается в отдельном потоке независимо от других.

# Приложение. Список источников

1. <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9C%D0%BE%D0%BD%D1%82%D0%B5-%D0%9A%D0%B0%D1%80%D0%BB%D0%BE>
2. <https://retireearly.ru/financial-literacy/metod-monte-karlo>
3. <https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D0%B5%D0%BF%D0%BB%D0%BE%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%B8>
4. <https://www.rsatu.ru/upload/medialibrary/ac1/Lektsiya-13.pdf>

# Приложение. Код программы

Код основной программы на языке программирования Python3.9

from queue import Queue

from random import random

from threading import Thread, Lock

import numpy as np

import pandas as pd

from sympy import false

import matplotlib.pyplot as plt

def generateCoefficient():

    V = 0

    n = 12

    for \_ in range(0, n):

        V += random()

    mV = n/2

    sigmaV = np.sqrt(n/12)

    z = (V - mV)/sigmaV

    return z

def calculateNet(L, t, a, randCoefficient, h, taw):

    if taw < h\*\*2 / (2 \* a) == false:

        raise ValueError('Сетка с данными параметрами неусточива')

    n = int(t/taw) + 1

    m = int(L/h) + 1

    U = np.zeros((n, m))

*# Краевые условия*

    for i in range(1, n):

        U[i, 0] = randCoefficient \* np.cos(i \* taw)

        U[i, -1] = 0

*# Начальное условие*

    for i in range(1, m):

        U[0, i] = 0

    for j in range(1, n):

        for i in range(1, m - 1):

            U[j, i] = (a \* taw/h\*\*2) \* (U[j - 1][i - 1] - 2 \*

                                     U[j - 1][i] + U[j - 1][i + 1]) + U[j - 1][i]

    U = pd.DataFrame(U)

    return U

def checkBorders(border: float, L, t, a, randCoefficient, h, taw):

    global netQueue

    U = calculateNet(L, t, a, randCoefficient, h, taw)

    tau, x = U.shape

    for i in range(0, tau):

        for j in range(0, x):

            if np.abs(U[j][i]) >= border:

*# print(U[j][i])*

                return

    netQueue.put(U)

    mutex.acquire()

    global k

    k += 1

    mutex.release()

def printOneGraph(queueNet, border, h):

    axis = np.arange(0, 1.1, h)

    topBorder = []

    bottomBorder = []

    for \_ in axis:

        topBorder.append(border)

        bottomBorder.append(-border)

    U = queueNet.get()

    tau, x = U.shape

    for i in range(0, tau):

        if i == 1 or i == 2 or i == 5 or i==10 or i==20 or i==50 or i==100 or i==200 or i==500 or i==1000:

            res = np.array(U.iloc[[i]])

            res = np.concatenate(res)

            plt.plot(axis, res, label=f't={i}')

    plt.legend(loc=1)

    plt.xlabel('Значение координаты X')

    plt.ylabel('Значение функции U')

    plt.plot(axis, topBorder, axis, bottomBorder)

    plt.show()

def printGraph(queueNet, border, h):

    axis = np.arange(0, 1.1, h)

    topBorder = []

    bottomBorder = []

    for \_ in axis:

        topBorder.append(border)

        bottomBorder.append(-border)

    while not queueNet.empty():

        U = queueNet.get()

        tau, x = U.shape

        for i in range(0, tau):

            res = np.array(U.iloc[[i]])

            res = np.concatenate(res)

            plt.plot(axis, res)

    plt.plot(axis, topBorder, axis, bottomBorder)

    plt.show()

k = 0

mutex = Lock()

netQueue = Queue()

def startExperiment(iterations, startingBorder, borderStep, desiredProbability, L, t, a, h, taw):

    global k, netQueue

    threadQueue = []

    border = startingBorder

    borderIterations = 0

    while k/iterations < desiredProbability:

        netQueue.queue.clear()

        k = 0

        for i in range(0, iterations + 1):

            th = Thread(target=checkBorders, args=(border, L, t, a, generateCoefficient(), h, taw))

            threadQueue.append(th)

            th.start()

        for i in threadQueue:

            i.join()

        print(f'Полученные результаты вероятность не выхода за границу: {k/iterations} \n Желаемая вероятность: {desiredProbability} \n Установленная граница: {border} \n Вероятность достигнута? {"Да" if k/iterations >= desiredProbability else "НЕТ"}')

        border += borderStep

        borderIterations += 1

    border -= borderStep *# лишний на выходе из алгоритма*

    print(f'Граница с вероятностью невыхода {desiredProbability \* 100}% {border} \nПотребовалось итераций {borderIterations} со стартовой границы {startingBorder} и шагом в {borderStep}')

    printGraph(netQueue, border, h)

Код с параметрами для выполнения основной функции

from task import startExperiment

*## Параметры эксперимента - число опытов, счетчик успешных случаев, граница, необходимая точность*

N = 10

k = 0

border = 1.5

step = 0.05

desiredProbability = 0.9

*## Параметры уравнения, границы и коэффициент a*

L = 1

t = 1

a = 1

*# Параметры сетки*

h = 0.1

taw = 0.001

startExperiment(iterations=N, startingBorder=border, borderStep=step, desiredProbability=desiredProbability, L=L, t=t, a=a, h=h, taw=taw)