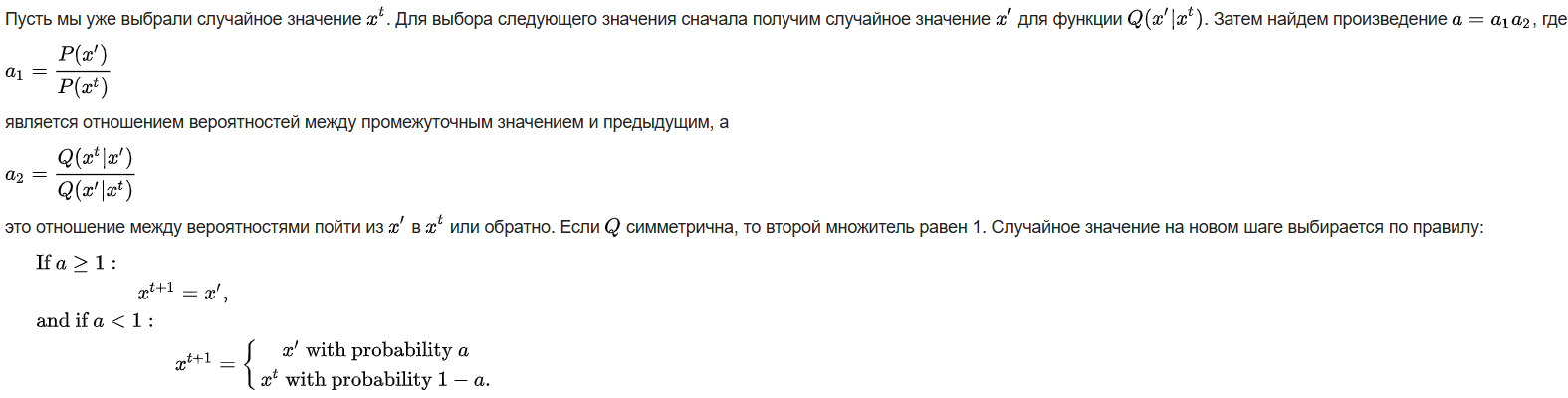
**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 3**

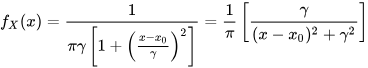
Алгоритм Метрополиса — Гастингса  
**Алгоритм Метрополиса — Гастингса** — алгоритм [семплирования](https://ru.wikipedia.org/wiki/Семплирование_(математическая_статистика)), использующийся, в основном, для сложных [функций распределения](https://ru.wikipedia.org/wiki/Функция_распределения). Он отчасти похож на алгоритм [выборки с отклонением](https://ru.wikipedia.org/wiki/Выборка_с_отклонением), однако здесь вспомогательная функция распределения меняется со временем. Алгоритм был впервые опубликован [Николасом Метрополисом](https://ru.wikipedia.org/wiki/Метрополис,_Николас_Константин) в [1953 году](https://ru.wikipedia.org/wiki/1953_год), и затем обобщён [К. Гастингсом](https://ru.wikipedia.org/w/index.php?title=К._Гастингсом&action=edit&redlink=1) в [1970 году](https://ru.wikipedia.org/wiki/1970_год). [Семплирование по Гиббсу](https://ru.wikipedia.org/wiki/Семплирование_по_Гиббсу) является частным случаем алгоритма Метрополиса — Гастингса и более популярно за счёт простоты и скорости, хотя и реже применимо.

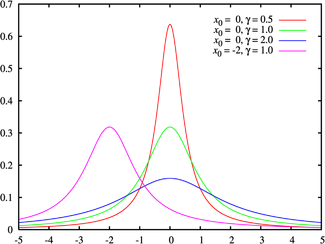


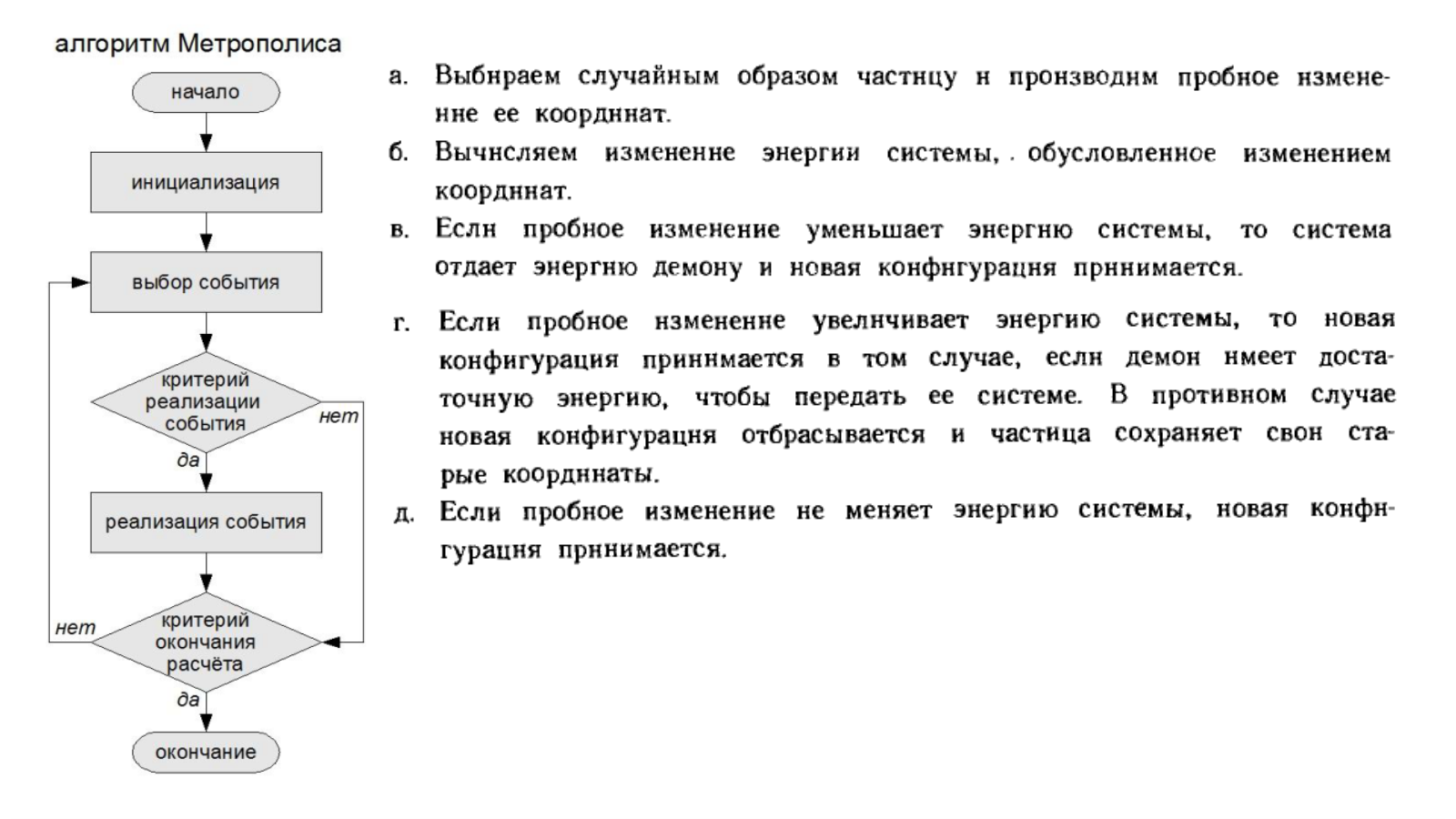
Распределение Коши

**Распределе́ние**[**Коши́**](https://ru.wikipedia.org/wiki/Коши,_Огюстен_Луи) в [теории вероятностей](https://ru.wikipedia.org/wiki/Теория_вероятностей) (также называемое в [физике](https://ru.wikipedia.org/wiki/Физика) **распределе́нием**[**Ло́ренца**](https://ru.wikipedia.org/wiki/Лоренц,_Людвиг_Валентин) и [**распределе́нием**](https://ru.wikipedia.org/wiki/Формула_Брейта_—_Вигнера)[**Бре́йта**](https://ru.wikipedia.org/wiki/Брейт,_Грегори)**—**[**Ви́гнера**](https://ru.wikipedia.org/wiki/Вигнер,_Юджин)) — класс [абсолютно непрерывных распределений](https://ru.wikipedia.org/wiki/Распределение_вероятностей). [Случайная величина](https://ru.wikipedia.org/wiki/Случайная_величина), имеющая распределение Коши, является стандартным примером величины, не имеющей [математического ожидания](https://ru.wikipedia.org/wiki/Математическое_ожидание) и [дисперсии](https://ru.wikipedia.org/wiki/Дисперсия_случайной_величины).

Распределение:





**Код:**

#include <iostream>

#include <fstream>

#include <vector>

#include <random>

#include <cmath>

#include <algorithm>

#include <string>

//const std::string& FILE\_NAME = "samples.txt";

const int NUM\_RUNS = 5;

// Функция для плотности распределения Коши

double cauchy\_density(double x, double x0, double gamma) {

return 1.0 / (M\_PI \* gamma \* (1.0 + std::pow((x - x0) / gamma, 2)));

}

// Реализация метода Метрополиса

std::vector<double> metropolis\_algorithm(double start\_point, double scale, int total\_samples, int warmup, std::pair<double, double> bounds) {

std::vector<double> data;

double current\_position = start\_point;

// Генераторы случайных чисел

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_real\_distribution<> uniform\_dist(-1.0, 1.0);

std::uniform\_real\_distribution<> acceptance\_dist(0.0, 1.0);

for (int step = 0; step < total\_samples + warmup; ++step) {

// Предлагаем новую точку

double candidate\_position = current\_position + uniform\_dist(gen);

// Проверка, чтобы точка была в пределах заданного диапазона

if (candidate\_position < bounds.first || candidate\_position > bounds.second) {

continue;

}

// Вероятность принятия нового состояния

double acceptance\_prob = cauchy\_density(candidate\_position, start\_point, scale) / cauchy\_density(current\_position, start\_point, scale);

// Принимаем новую точку с вероятностью acceptance\_prob

if (acceptance\_dist(gen) < std::min(1.0, acceptance\_prob)) {

current\_position = candidate\_position;

}

// Добавляем точку в выборку после фазы разогрева

if (step >= warmup) {

data.push\_back(current\_position);

}

}

return data;

}

void write\_samples\_to\_file(std::ofstream& output, const std::vector<double>& samples, int run\_number) {

output << "Run " << run\_number << ":\n";

for(double sample : samples)

output << sample << '\n';

output << '\n';

}

int main() {

// Параметры распределения Коши

double x\_center = 0.0; // Сдвиг по оси

double gamma = 0.5; // Гамма

std::vector<int> nums\_samples = {100, 1000}; // Количество семплов

int warmup\_steps = 500; // Отбрасываемое количество точек

double initial\_x = 0.0; // Стартовая точка

std::pair<double, double> boundary = {-1.0, 9.0}; // Границы выборки

for (int num\_samples : nums\_samples) {

std::string filename = "samples" + std::to\_string(num\_samples) + ".txt";//создаем имя файла

std::ofstream output(filename);//создаем поток вывода в файл

if (!output.is\_open()){//открываем поток и если он не открылся выводим ошибку и убиваем прогу

std::cerr << "Error in opening file " << filename << '\n';

exit(1);

}

for (int run = 0; run < NUM\_RUNS; ++run) {

// Получаем выборку методом Метрополиса

std::vector<double> generated\_samples = metropolis\_algorithm(initial\_x, gamma, num\_samples, warmup\_steps, boundary);

write\_samples\_to\_file(output, generated\_samples, run);

std::cout << "Run #" << run << " for sample size " << num\_samples << " saved to " << filename << '\n';

}

/\*for (double sample : generated\_samples) {

output << sample << std::endl;

}\*/

output.close();

}

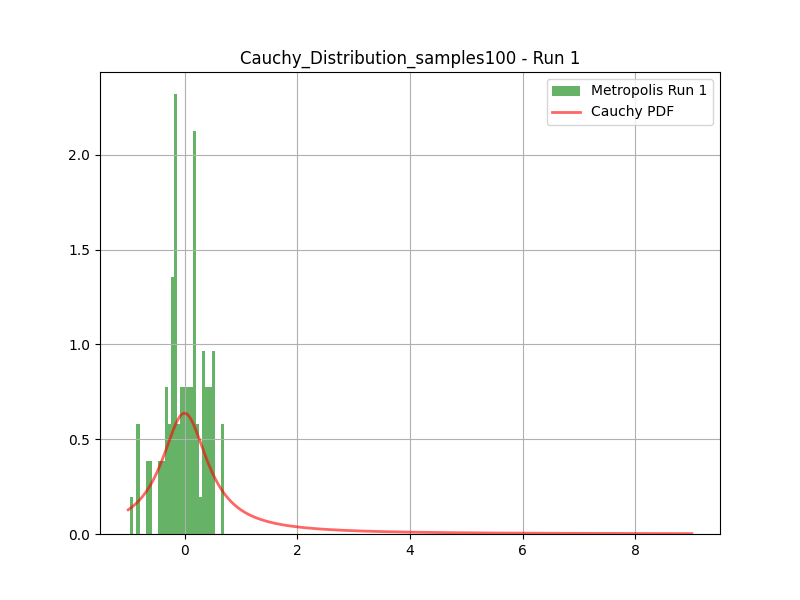
return 0;

}

Построение графиков:

import numpy as np  
import matplotlib.pyplot as plt  
  
# Функция для плотности распределения Коши  
def cauchy\_density(x, x0, gamma):  
 return 1 / (np.pi \* gamma \* (1 + ((x - x0) / gamma) \*\* 2))  
  
# Функция для парсинга данных из файла с несколькими запусками  
def parse\_runs(file\_path):  
 with open(file\_path, 'r') as f:  
 data = f.read()  
  
 runs = data.split('Run') # Разбиваем на части по ключевому слову 'Run'  
 parsed\_runs = []  
  
 for run in runs[1:]: # Пропускаем первую пустую часть  
 run\_data = run.strip().splitlines()[1:] # Пропускаем строку 'Run N:'  
 run\_samples = np.array([float(x) for x in run\_data])  
 parsed\_runs.append(run\_samples)  
  
 return parsed\_runs  
  
  
# Параметры распределения Коши  
x\_center = 0.0  
gamma = 0.5  
boundary = [-1, 9] # Диапазон выборки  
  
samples100 = parse\_runs("/home/buzzismaloy/University\_shit/7th-term/monte-karlo/third-lab/samples100.txt")  
samples1000 = parse\_runs("/home/buzzismaloy/University\_shit/7th-term/monte-karlo/third-lab/samples1000.txt")  
  
  
# Построение и сохранение графиков для каждого запуска  
def plot\_and\_save\_runs(runs, title\_prefix, boundary, x\_center, gamma, save\_dir):  
 for i, run\_samples in enumerate(runs, 1):  
 plt.figure(figsize=(8, 6))  
 # Построение гистограммы для выборки  
 plt.hist(run\_samples, bins=30, density=True, alpha=0.6, color='green', label=f'Metropolis Run {i}')  
  
 # Теоретическая плотность распределения Коши  
 x\_values = np.linspace(boundary[0], boundary[1], 1000)  
 plt.plot(x\_values, cauchy\_density(x\_values, x\_center, gamma), 'r-', lw=2, alpha=0.6, label='Cauchy PDF')  
  
 # Оформление графика  
 plt.title(f'{title\_prefix} - Run {i}')  
 plt.legend()  
 plt.grid(True)  
  
 # Сохранение графика в файл  
 file\_name = f'{save\_dir}/{title\_prefix}\_Run\_{i}.png'  
 plt.savefig(file\_name)  
 plt.close()  
  
  
  
plot\_and\_save\_runs(samples100, 'Cauchy\_Distribution\_samples100', boundary, x\_center, gamma,  
 "/home/buzzismaloy/University\_shit/7th-term/monte-karlo/third-lab/plots")  
  
plot\_and\_save\_runs(samples1000, 'Cauchy\_Distribution\_samples1000', boundary, x\_center, gamma,  
 "/home/buzzismaloy/University\_shit/7th-term/monte-karlo/third-lab/plots")

**Графики:**

****

