**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Кафедра теории вероятностей и математической статистики

**Крагель Алины Олеговны**

**Моделирование базовой случайной величины**

Отчет по лабораторной работе №4

(«Имитационное и статистическое моделирование»)

Студентки 4 курса 9 группы

**Преподаватель**

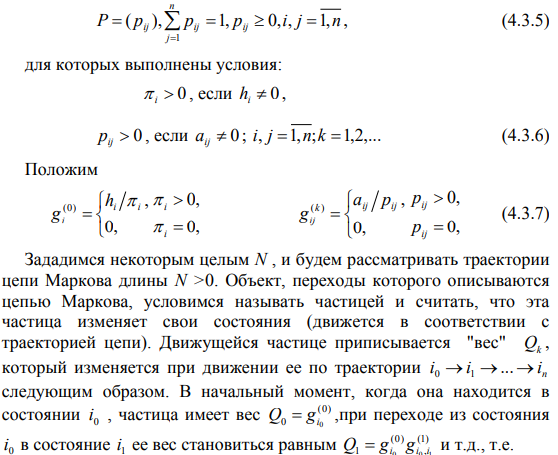
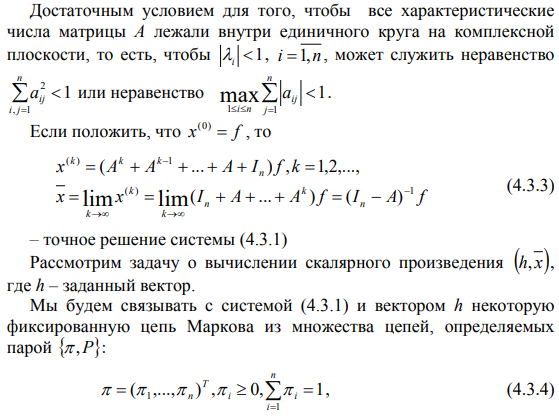
*Гайдук Антон Николаевич*

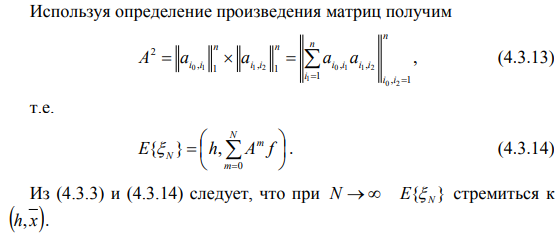
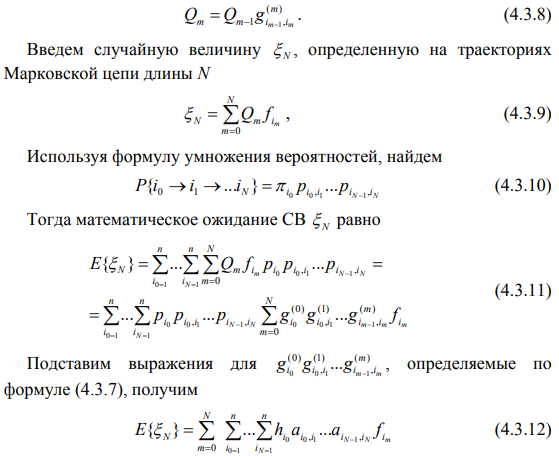
# Теоретическая часть

## Общая схема метода Монте-Карло

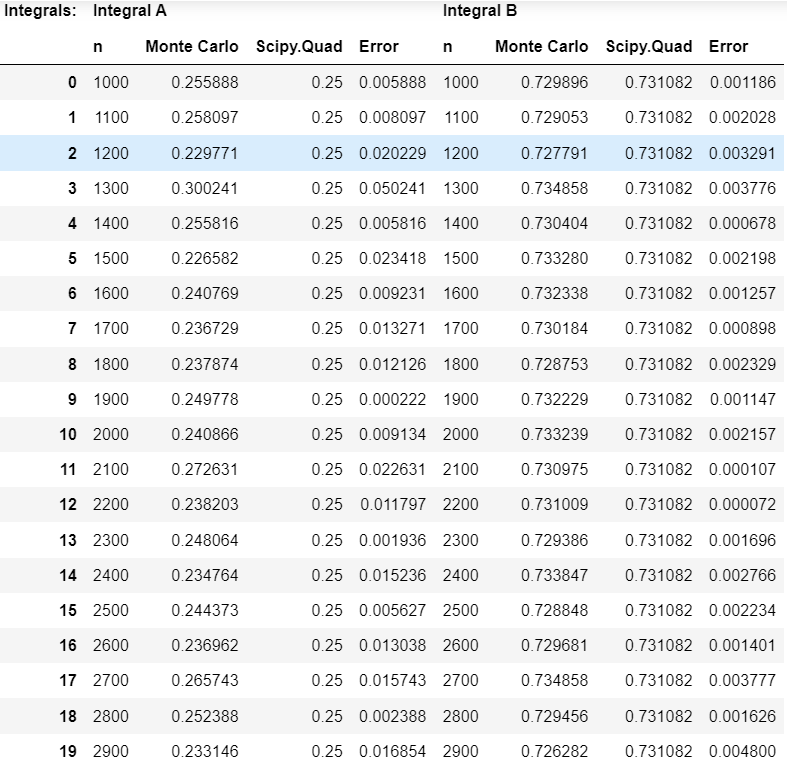
## Вычисление определенного интеграла методом Монте-Карло

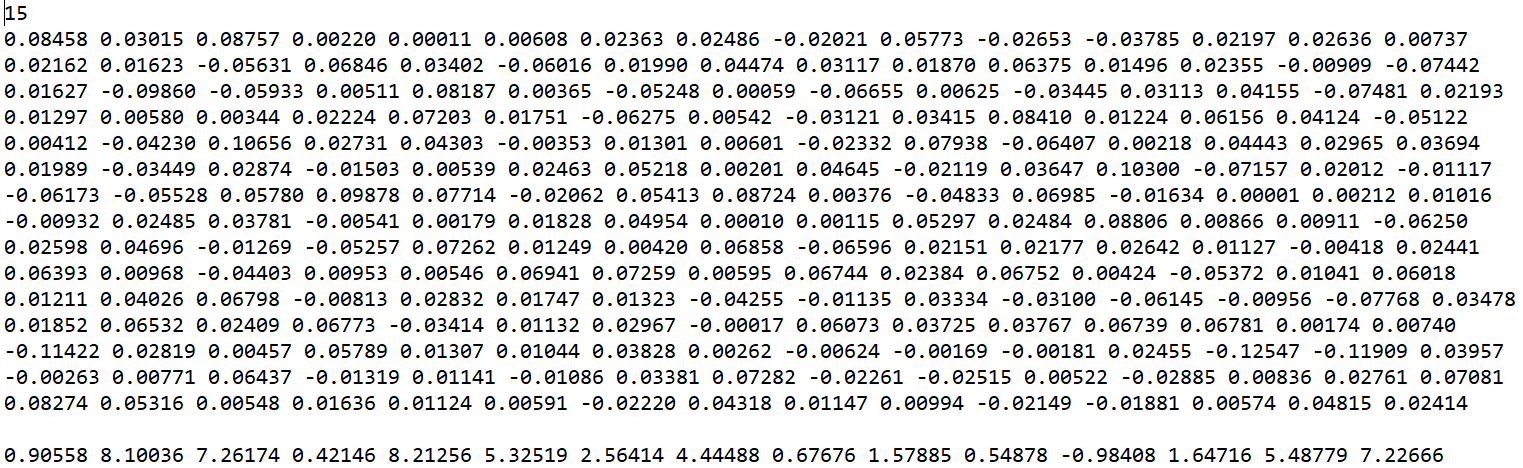
## Решение системы алгебраических уравнений методом Монте-Карло

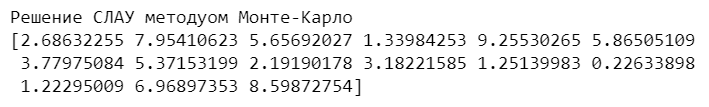
  
  
  
  
  
  
  
  


# Результаты экспериментов

Результаты вычислений интегралов а и б методом Монте-Карло с точностью 10^(-7) на n-ом шаге от 1\_000 до 3\_000 шагов:

Решение системы СЛАУ из заданного варианта (7):



*#!/usr/bin/env python  
# coding: utf-8  
# # S&SM  
# ## L4. Alina Kragel, gr. 9  
# In[21]:*import numpy as np  
import pandas as pd  
import scipy.integrate as integrate  
from math import exp, log, sqrt  
from random import uniform  
from scipy.linalg import eigvals, solve  
  
  
*# ### Вычислить приближенное значение по методу Монте-Карло интегралов ниже  
# ![image.png](attachment:image.png)  
# In[2]:*def function\_a(x):  
 return x / sqrt(1 - 4 \* x\*\*2)  
  
  
*# In[3]:*def function\_b(x):  
 return (x \* np.log(x)) / (1 + x \*\* 3)  
  
  
*# In[4]:*def integral\_a(function, a, b, n):  
 return sum([function(uniform(a, b)) for \_ in range(n)]) \* (b - a) / n  
  
  
*# In[9]:*def integral\_b(function, n):  
 result = 0  
 temp = 1  
 left = 0  
  
 while abs(temp) > 10 \*\* (-7):  
 temp = 0  
 for \_ in range(n):  
 c\_dot = uniform(left, left + 1)  
 temp += function(c\_dot)  
 temp /= n  
 result += temp  
 left += 1  
   
 return result  
  
  
*# In[10]:*def get\_data():  
 bunch = []  
 for n in range(1\_000, 3\_000, 100):  
 mc1 = integral\_a(function\_a, 0, 0.5, n)  
 print(mc1)  
 math1 = integrate.quad(function\_a, 0, 0.5)[0]  
 print(math1)  
  
 mc2 = integral\_b(function\_b, n)  
 print(mc2)  
 math2 = integrate.quad(function\_b, 0, np.inf)[0]  
 print(math2)  
  
 bunch.append([n, mc1, math1, abs(mc1 - math1), n, mc2, math2, abs(mc2 - math2)])  
 return bunch  
  
  
*# In[11]:*def get\_report\_for\_test():  
 data = get\_data()  
 df = pd.DataFrame(data,  
 columns=pd.MultiIndex.from\_product([[**'Integral A'**, **'Integral B'**],  
 [**'n'**, **'Monte Carlo'**, **'Scipy.Quad'**, **'Error'**]],  
 names=[**'Integrals:'**, **' '**]))  
 return df  
  
  
*# In[12]:*results = get\_report\_for\_test()  
results  
  
  
*# ### Решить СЛАУ методом Монте-Карло, вариант ниже  
# ![image.png](attachment:image.png)  
# In[27]:*def check\_matrix(matrix):  
 data = []  
 for eigval in eigvals(A):  
 eigval\_modulo = abs(eigval)  
 assert eigval\_modulo < 1, **'modulo > 1'** data.append([np.round(eigval, 2), np.round(abs(eigval), 2)])  
 return data  
  
  
*# In[28]:*def solve(size, chain\_length, matrix):  
 m = size \* 1000   
   
 h = np.eye(size)   
 pi = np.ones\_like(f) \* (1.0 / size) *# Вектор вероятностей начальных состояний цепи Маркова* P = np.array(np.ones\_like(matrix)) \* (1 / size) *# Матрица переходов* ksi = np.zeros((m, size), dtype=float)  
 idxs = np.array(np.random.rand(m, chain\_length) // (1 / size), dtype=int)  
 Q = np.zeros((m, chain\_length, size), dtype=float)  
  
 for j in range(m):  
 idx = idxs[j][0]  
 Q[j][0][idx] = 0 if not pi[idx] else h[idx, idx] / pi[idx]  
 for k in range(1, chain\_length):  
 old\_state = idxs[j][k - 1]  
 new\_state = idxs[j][k]  
 Q[j][k][idx] = (  
 0 if not P[old\_state][new\_state]  
 else Q[j][k - 1][idx] \* matrix[old\_state][new\_state] / P[old\_state][new\_state]  
 )  
  
 ksi[j] = np.dot(f[idxs[j]], Q[j])  
  
 return ksi.mean(axis=0)  
  
  
*# In[29]:*with open(**"C://Users//aowl2//OneDrive//Рабочий стол//bsu/7 s//s&sm//l4//07-student.txt"**) as file:  
 data = file.readlines()  
  
  
*# In[30]:*matrix = data[1:-2]  
A = [[float(num) for num in line.split(**' '**)[:-1]] for line in matrix]  
  
  
*# In[31]:*vector = data[-1]  
f = [float(num) for num in vector.split(**' '**)[:-1]]  
f = np.array(f)  
  
  
*# In[35]:*check\_data = check\_matrix(A)  
print(**'Решение СЛАУ методуом Монте-Карло'**)  
answer = solve(15, 100, A)  
print(answer)