|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования 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 «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ**

*к лабораторной работе №2*

*По курсу: «Моделирование»*

*Тема:* ***«Марковские процессы»***

Студент ИУ7-74Б

Жабин Д.В.

Преподаватель

Рудаков И.В.

*Москва, 2022 г.*

**Задание**

Смоделировать работу сложной системы, имеющей не более 10 состояний. По заданной матрице интенсивностей для каждого из состояний системы рассчитать предельную вероятность и время стабилизации.

# Теоретическая часть

Случайный процесс, протекающий в системе S, называется марковским, если он обладает следующим свойством: для каждого момента времени 𝑡0 вероятность любого состояния системы в будущем (при 𝑡 > 𝑡0) зависит только от ее состояния в настоящем (при 𝑡 = 𝑡0) и не зависит от того, когда и каким образом система пришла в это состояние. Вероятностью i-го состояния называется вероятность 𝑝𝑖(𝑡) того, что в момент t система будет находиться в состоянии 𝑆𝑖. Для любого момента t сумма вероятностей всех состояний равна единице.

Для решения поставленной задачи, необходимо составить систему уравнений Колмогорова по следующим принципам: в левой части каждого из уравнений стоит производная вероятности i-го состояния; в правой части — сумма произведений вероятностей всех состояний (из которых идут стрелки в данное состояние), умноженная на интенсивности соответствующих потоков событий, минус суммарная интенсивность всех потоков, выводящих систему из данного состояния, умноженная на вероятность данного (i-го состояния).

## Пример

Система имеет 3 состояния с матрицей интенсивностей, описанной в табл. 1.

Таблица 1 – матрица интенсивностей

|  |  |  |
| --- | --- | --- |
| 0 |  |  |
|  | 0 |  |
|  |  | 0 |

(1)

Для получения предельных вероятностей, то есть вероятностей в стационарном режиме работы при необходимо приравнять левые части уравнений к нулю. Таким образом получается система линейных уравнений. Для решения полученной системы необходимо добавить условие нормировки ().

Также необходимо найти время, в которое достигается вероятностная константа. Для этого найдем все значения вероятности , как функции времени, с интервалом в некотором интервале [t0, tN]. Когда найденная вероятность будет равна найденной ранее вероятностной константе с точностью до заданной погрешности, можно считать искомое время найденным. На каждом шаге необходимо вычислять приращения для каждой вероятности:

.

**Текст программы**

|  |
| --- |
| from random import random  from numpy import linalg  from prettytable import PrettyTable  deltaT = 1e-3  eps = 1e-3  def randMatrix(N):  matrix = [[0.0 for j in range(N)] for i in range(N)]  for i in range(N):  for j in range(N):  if i != j:  matrix[i][j] = round(random(), 4)  return matrix  def getCoeffs(matrix):  n = len(matrix)  lst = [[None for j in range(n)] for i in range(n)]  for i in range(n):  if i != (n - 1):  for j in range(n):  if j != i:  lst[i][j] = matrix[j][i]  else:  lst[i][j] = -sum(matrix[i])  else:  for j in range(n):  lst[i][j] = 1  return lst  def getLimitProbs(matrix):  coeffs = getCoeffs(matrix)  N = len(matrix)  return linalg.solve(coeffs, [0 if i != (N - 1) else 1 for i in range(N)]).tolist()  def getDeltaProbs(matrix, start\_probs):  n = len(matrix)  return [deltaT \* sum([-sum(matrix[i]) \* start\_probs[j] if i == j  else matrix[j][i] \* start\_probs[j] for j in range(n)]) for i in range(n)]  def getLimitTimes(matrix, limit\_probs):  n = len(matrix)  limit\_times = [0.0] \* n  current\_time = 0.0  start\_probs = [1.0 / n] \* n  current\_probs = start\_probs.copy()  while not all(limit\_times):  deltaP = getDeltaProbs(matrix, current\_probs)  for i in range(n):  if not limit\_times[i] and abs(current\_probs[i] - limit\_probs[i]) <= eps:  limit\_times[i] = current\_time  current\_probs[i] += deltaP[i]  current\_time += deltaT  return limit\_times  def calculate(matrix):  probs = [round(x, 4) for x in getLimitProbs(matrix)]  times = [round(x, 4) for x in getLimitTimes(matrix, probs)]  return probs, times  def output(matrix, res\_p, res\_t):  table\_matrix = PrettyTable()  cols = ["Состояния"]  cols.extend([str(i + 1) for i in range(len(matrix))])  table\_matrix.field\_names = cols  for i in range(len(matrix)):  tmp = [item for item in matrix[i]]  tmp.insert(0, i + 1)  table\_matrix.add\_row(tmp)  print(table\_matrix)  print()  table\_res = PrettyTable()  table\_res.add\_column("Состояния", [i + 1 for i in range(len(res\_p))])  table\_res.add\_column("Предельная вероятность", res\_p)  table\_res.add\_column("Время стабилизации", res\_t)  print(table\_res)  def main():  N = int(input("Введите количество состояний системы: "))  matrix = randMatrix(N)  probs, times = calculate(matrix)  output(matrix, probs, times)  if \_\_name\_\_ == '\_\_main\_\_':  main() |

**Пример работы**

Показаны результаты для следующих вероятностей состояний системы в начальный момент времени: Pj(0) = 1 / n, j = 1̅,̅n.

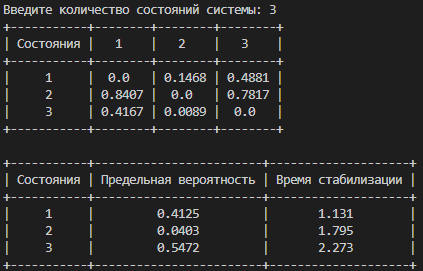


Рисунок 1 – Случай для 3 состояний системы

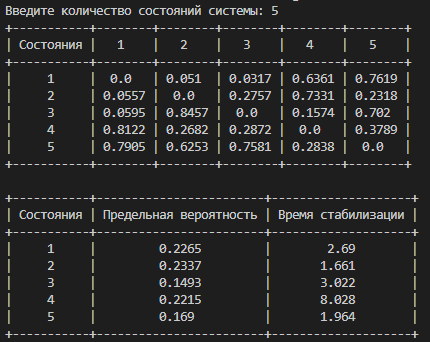


Рисунок 2 – Случай для 5 состояний системы

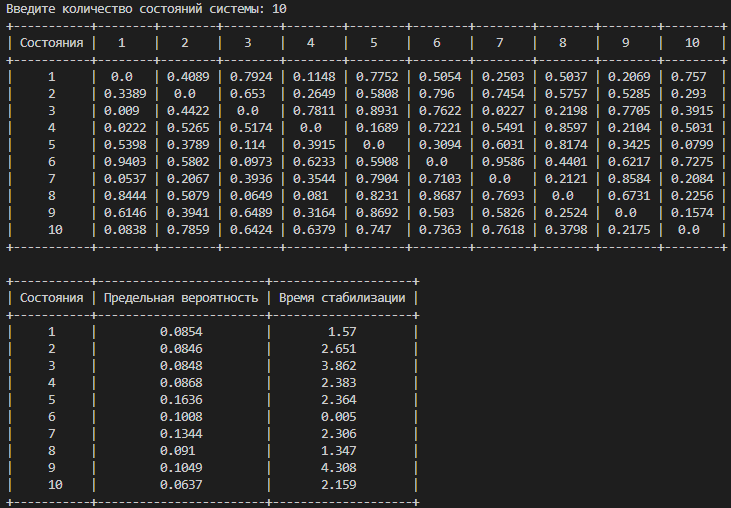


Рисунок 3 – Случай для 10 состояний системы