|  |  |
| --- | --- |
| 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**

**по лабораторной работе № 2**

**по курсу: «Моделирование»**

|  |  |
| --- | --- |
| **Тема Марковские процессы**  **Студент Якуба Д. В.**  **Группа ИУ7-73Б**  **Оценка (баллы)**  **Преподаватель Рудаков И.В.** |  |

Москва, 2021

1. Задание

Написать программу, которая позволяет определить время пребывания случайной системы в каждом из состояний. Количество состояний не более десяти. Для каждого состояния также рассчитать предельную вероятность.

2. Теория

Случайный процесс, протекающий в некоторой системе, называют марковским, если он обладает следующим свойством: для каждого момента времени вероятность любого состояния системы в будущем (зависит только от её состояния в настоящем и не зависит от того, когда и каким образом система пришла в это состояние (как процесс развивался в прошлом).

Функционирование системы может быть задано размеченным графом, где дуги обозначают интенсивности переходов, а узлы – состояния системы.

Для решения поставленной задачи может быть составлена система, состоящая из уравнений Колмогорова, каждое из которых имеет вид:

где – вероятность нахождения системы в состоянии в момент времени , – интенсивность перехода системы из состояния в состояние .

3. Выполнение

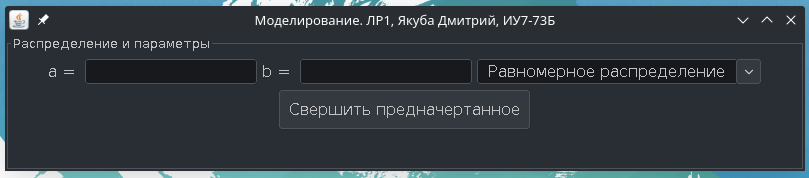
На рисунках 2.1-2.2 предоставлен интерфейс разработанного приложения для ввода параметров равномерного распределения и распределения Гаусса.

Рис. 2.1 – Интерфейс для ввода параметров равномерного распределения

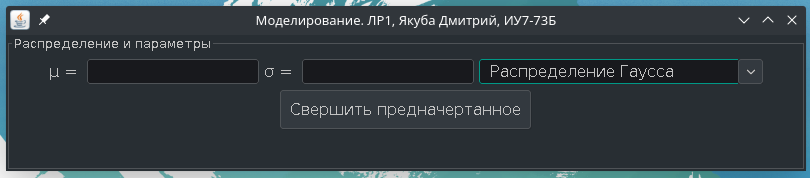
**3.1 Равномерное распределение**

Рис. 2.2 – Интерфейс для ввода параметров распределения Гаусса

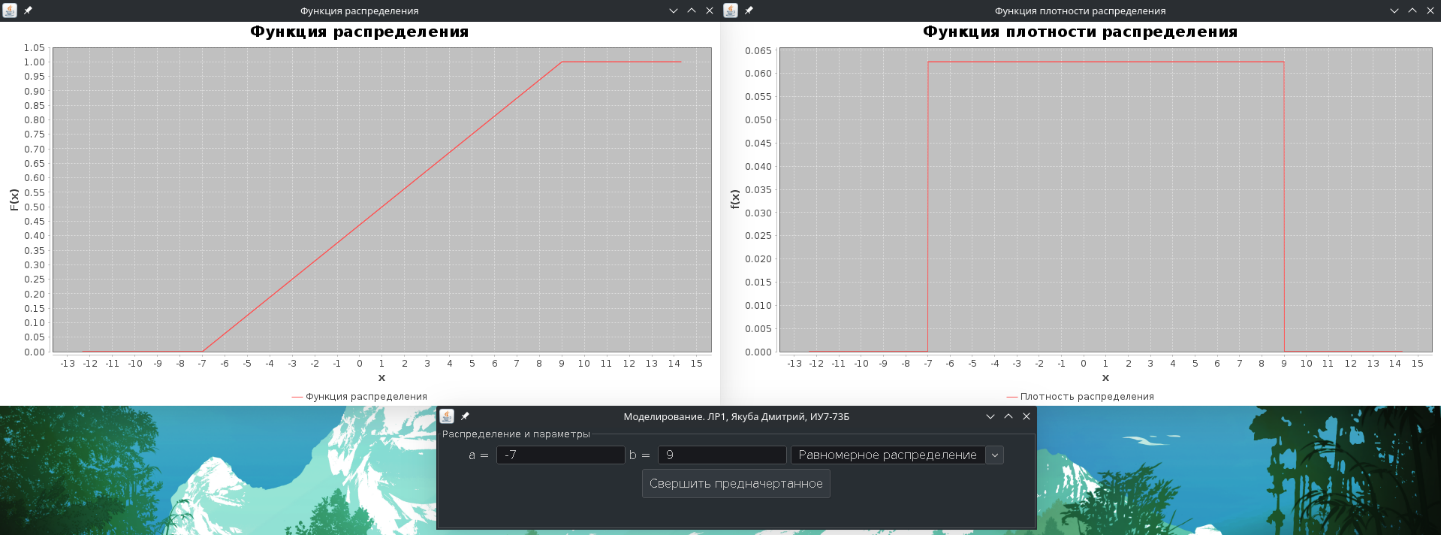
На рисунках 2.3-2.5 предоставлен пример работы приложения при построении графика функции распределения и плотности распределения для равномерного распределения.

Рис. 2.3 – Пример работы приложения при построении графика функции распределения и плотности распределения для равномерного распределения с параметрами (все три окна)



Рис. 2.4 - Пример работы приложения при построении графика функции распределения и плотности распределения для равномерного распределения с параметрами (окно графика функции распределения)



Рис. 2.5 – Пример работы приложения при построении графика функции распределения и плотности распределения для равномерного распределения с параметрами (окно графика функции плотности распределения)

**3.2 Распределение Гаусса**

На рисунке 2.6-2.8 предоставлен пример работы приложения при построении графика функции распределения и плотности распределения для распределения Гаусса.

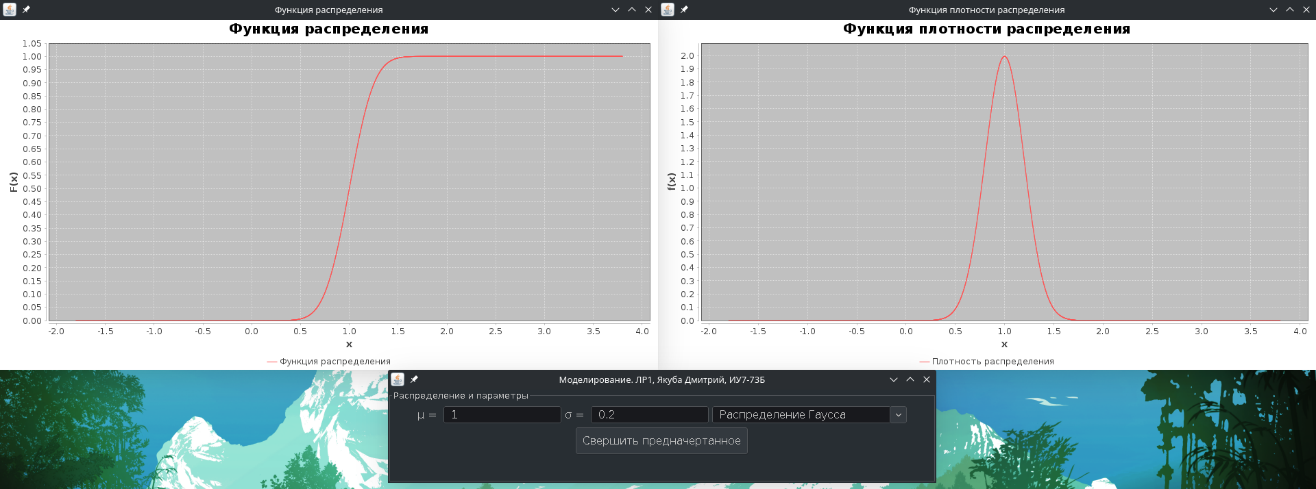


Рис. 2.6 – Пример работы приложения при построении графика функции распределения и плотности распределения для распределения Гаусса с параметрами (все три окна)



Рис. 2. 7 - Пример работы приложения при построении графика функции распределения и плотности распределения для распределения Гаусса с параметрами (окно графика функции распределения)



Рис. 2. 8 – Пример работы приложения при построении графика функции распределения и плотности распределения для распределения Гаусса с параметрами (окно графика функции плотности распределения)

4. Листинг

В данном разделе предоставлены используемые методы для построения требуемых графиков (используемый ЯП – Kotlin).

|  |
| --- |
| private *fun* makeAndAddPlotsForEvenDistribution(parameterA\_: Float, parameterB\_: Float)  {  *var* parameterA = parameterA\_  *var* parameterB = parameterB\_      if (parameterA > parameterB)      {          parameterA = parameterB.also { parameterB = parameterA }      }  *val* sizeOfPlate = parameterB - parameterA  *val* stepOnPlate = sizeOfPlate / 1e3  *var* currentX = parameterA - sizeOfPlate / 3  *val* distributionSeries = XYSeries("Функция распределения")  *val* densitySeries = XYSeries("Плотность распределения")      for (i in 0 until (5 \* sizeOfPlate / (3 \* stepOnPlate)).toInt())      {  *val* curDistr = when          {          currentX < parameterA -> 0          currentX > parameterB -> 1          else -> (currentX - parameterA) / (parameterB - parameterA)          }          distributionSeries.add(currentX, curDistr)  *val* curDensity = when (currentX)          {          in parameterA..parameterB -> 1 / (parameterB - parameterA)          else              -> 0          }          densitySeries.add(currentX, curDensity)          currentX += stepOnPlate.toFloat()      }      createWindowWithPlot(          ChartPanel(          ChartFactory.createXYLineChart(              "Функция распределения",              "x",              "F(x)",              XYSeriesCollection(distributionSeries),              PlotOrientation.VERTICAL, true, true, false          )          ), "Функция распределения"      )      createWindowWithPlot(          ChartPanel(          ChartFactory.createXYLineChart(              "Функция плотности распределения",              "x",              "f(x)",              XYSeriesCollection(densitySeries),              PlotOrientation.VERTICAL, true, true, false          )          ), "Функция плотности распределения"      )  }  private *fun* makeAndAddPlotsForGaussianDistribution(parameterMu: Float, parameterSigma: Float)  {  *val* startPosition = parameterMu - 2 - parameterSigma \* 4  *val* endPosition = parameterMu + 2 + parameterSigma \* 4  *val* step = 1e-3  *val* distributionSeries = XYSeries("Функция распределения")  *val* densitySeries = XYSeries("Плотность распределения")  *var* currentX = startPosition.toDouble()      for (i in 0 until ((endPosition - startPosition) / step).toInt())      {  *val* curDistribution = 0.5 \* (1 + org.apache.commons.math.special.Erf.erf(          (currentX - parameterMu) / sqrt(2 \* parameterSigma \* parameterSigma)          ))  distributionSeries.add(currentX, curDistribution)  *val* curDensity =          (1 / (sqrt(2.0 \* PI \* parameterSigma \* parameterSigma))) \* exp(-(((currentX - parameterMu) \* (currentX - parameterMu)) / (2 \* parameterSigma \* parameterSigma)))          densitySeries.add(currentX, curDensity)          currentX += step      }      createWindowWithPlot(          ChartPanel(          ChartFactory.createXYLineChart(              "Функция распределения",              "x",              "F(x)",              XYSeriesCollection(distributionSeries),              PlotOrientation.VERTICAL, true, true, false          )          ), "Функция распределения"      )      createWindowWithPlot(          ChartPanel(          ChartFactory.createXYLineChart(              "Функция плотности распределения",              "x",              "f(x)",              XYSeriesCollection(densitySeries),              PlotOrientation.VERTICAL, true, true, false          )          ), "Функция плотности распределения"      )  } |