**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря**

**Сікорського**

**Факультет обчислювальної техніки**

**Кафедра обчислювальної техніки**

ЗВІТ

з лабораторної роботи 1

з навчальної дисципліни «Основи комп’ютерного моделювання»

Тема: Перевірка генератора випадкових чисел

на відповідність закону розподілу

Виконав

Студент 4 курсу ІП-94

Рекечинський Дмитро

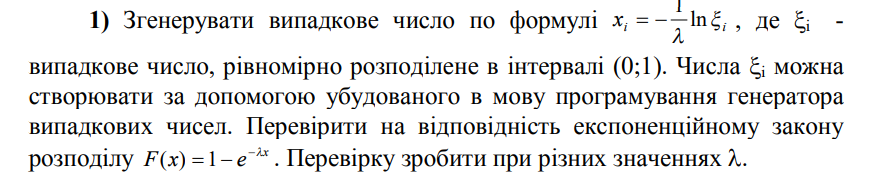
Перевірив

Іваніщев Б. В.

Київ 2022

**Завдання до роботи:** Згенерувати 10000 випадкових чисел способом, указаним у варіанті. Побудувати гістограму частот, знайти середнє і дисперсію цих випадкових чисел. По виду гістограми частот визначити вид закону розподілу. Відповідність заданому закону розподілу перевірити за допомогою критерію згоди хі-квадрат.

**Варіант завдання:** 9423 mod 3 + 1 = 1



**Лістинг програми**

Примітка: Програма написана за допомогою компільованої мови програмування Nim.

Команда для компіляції та запуску: nim c -r solution.nim

Вміст програми у файлі solution.nim:

# sequence.sorted()

import std/algorithm

# ln(), sequence.sum(), sqrt(), exp()

import std/math

# initRand(), rand()

import std/random

# sequence.mapIt()

import std/sequtils

#############################

# Types

#############################

type

StatisticValues = tuple[average, variance: float]

HistogramUnit = tuple[lowerBound, upperBound: float, occurencies: int]

Histogram = seq[HistogramUnit]

#############################

# Constants, global variables

#############################

const AMOUNT\_OF\_GENERATED\_NUMBERS = 10000

const NUMBER\_OF\_FRACTIONAL\_DIGITS = 2

const MINIMAL\_NUMBER\_OF\_OCCURENCIES = 5

# For the current task, there is only one parameter of distribution law:

# lambda

const NUMBER\_OF\_DISTRIBUTION\_LAW\_PARAMETERS = 1

# Recommended amount of intervals

const AMOUNT\_OF\_INTERVALS = 20

# Level of significance equals 0.05

const PEARSONS\_CHI\_SQUARE\_TEST = [

3.8,

6.0,

7.8,

9.5,

11.1,

12.6,

14.1,

15.5,

16.9,

18.3,

19.7,

21.0,

22.4,

13.7,

25.0,

26.3,

27.6,

28.9,

30.1,

31.4

]

var randomizer = initRand()

#############################

# Functions

#############################

proc randomNumber(): float = randomizer.rand(1.0)

proc givenFunction(lambd: float): float = (-1.0 / lambd) \* ln(randomNumber())

proc calculateError(expected, actual: float): float =

abs((expected - actual) / actual)

proc generateSequence(lambd: float): seq[float] =

for \_ in 1..AMOUNT\_OF\_GENERATED\_NUMBERS:

let number = givenFunction(lambd)

result.add(number)

proc calculateStatisticValues(sequence: seq[float]): StatisticValues =

let sumOfNumbers = sequence.sum()

let amount = float(AMOUNT\_OF\_GENERATED\_NUMBERS)

let average = sumOfNumbers / amount

let variancePart = sequence.mapIt((it - average)^2).sum()

let squaredVariance = variancePart / (amount - 1.0)

let variance = sqrt(squaredVariance)

return (average, variance)

proc calculateHistogram(sequence: seq[float], intervalAmount: int): Histogram =

let sortedSeq = sequence.sorted()

let minimum = sortedSeq[sortedSeq.low]

let maximum = sortedSeq[sortedSeq.high]

let stepValue = (maximum - minimum) / float(intervalAmount)

var intervalStep = minimum

var index = sortedSeq.low

for \_ in 1..intervalAmount:

let lowerBound = intervalStep

intervalStep += stepValue

let upperBound = intervalStep

var occurencies = 0

while true:

var currentValue = sortedSeq[index]

if currentValue <= intervalStep:

inc occurencies

inc index

if index > sortedSeq.high: break

else: break

result.add((lowerBound: lowerBound, upperBound: upperBound, occurencies: occurencies))

proc joinIntervalsHelper (target: var HistogramUnit, source: HistogramUnit) =

let intervalBounds = @[

target.lowerBound,

target.upperBound,

source.lowerBound,

source.upperBound

]

let lowerBound = intervalBounds.min()

let upperBound = intervalBounds.max()

target.lowerBound = lowerBound

target.upperBound = upperBound

target.occurencies += source.occurencies

proc joinIntervals(histogram: var Histogram) =

# Join intervals while any of them has a number of occurencies

# lower than it's allowed

while histogram.anyIt(it.occurencies < MINIMAL\_NUMBER\_OF\_OCCURENCIES):

var index = histogram.high

while index > histogram.low:

let currentInterval = histogram[index]

let occurencies = currentInterval.occurencies

if occurencies < MINIMAL\_NUMBER\_OF\_OCCURENCIES:

var intervalIndex: int

# Join the interval with one which has lower number of occurencies

if index == histogram.high:

intervalIndex = index - 1

else:

let previousIndex = index - 1

let nextIndex = index + 1

intervalIndex = if histogram[previousIndex] > histogram[nextIndex]:

nextIndex

else: previousIndex

joinIntervalsHelper(histogram[intervalIndex], currentInterval)

histogram.delete(index)

break

dec index

proc hypoteticalDistributionLaw(x, lambd: float): float = 1 - exp(-lambd \* x)

proc chiSquaredTest(histogram: Histogram, lambd: float): float =

for interval in histogram:

let lowerBound = interval.lowerBound

let upperBound = interval.upperBound

let theoreticalHitRate =

hypoteticalDistributionLaw(upperBound, lambd) - hypoteticalDistributionLaw(lowerBound, lambd)

let theoreticalOccurencies = (theoreticalHitRate \* AMOUNT\_OF\_GENERATED\_NUMBERS).round().int()

result += (interval.occurencies - theoreticalOccurencies)^2 / theoreticalOccurencies

#############################

# Usage

#############################

const lambdas = [0.05, 0.1, 0.2, 0.25]

for lambd in lambdas:

echo "\n============================="

echo "Lambda: ", lambd

let expectedAverage = 1.0 / lambd

let expectedVariance = 1.0 / lambd

let sequence = generateSequence(lambd)

let statisticValues = calculateStatisticValues(sequence)

echo "Expected average: ", expectedAverage

echo "Actual average: ", statisticValues.average

echo "Relative calculation error: ",

calculateError(expectedAverage, statisticValues.average)

echo "Expected variance: ", expectedVariance

echo "Actual variance: ", statisticValues.variance

echo "Relative calculation error: ",

calculateError(expectedVariance, statisticValues.variance)

var histogram = calculateHistogram(sequence, AMOUNT\_OF\_INTERVALS)

histogram.joinIntervals()

echo "Histogram sequence"

for interval in histogram: echo interval.occurencies

let chiSquaredTestValue = histogram.chiSquaredTest(lambd)

echo "Chi-squared test: ", chiSquaredTestValue

let freedomDegree = histogram.len - 1 - NUMBER\_OF\_DISTRIBUTION\_LAW\_PARAMETERS

let theoreticalChiSquare = PEARSONS\_CHI\_SQUARE\_TEST[freedomDegree]

echo "Theoretical value of chi-square test: ", theoreticalChiSquare

echo "Passed: ", chiSquaredTestValue < theoreticalChiSquare

**Перевірка роботи**

Програма генерує числа при таких значеннях параметру “лямбда”: 0.05, 0.1, 0.2, 0.25.

Усі виводи програми, наведені у звіті, перекладені із англійської в українську. Програма при роботі виводить текст англійською мовою.

1. Значення середнього та дисперсії

Програма виводить на екран результати поточного значення середнього та дисперсії, а також обчислює відносну похибку:

=============================

Лямбда: 0.05

Очікуване середнє: 20.0

Поточне середнє: 20.13555499532282

Відносна похибка: 0.00673212113370145

Очікувана дисперсія: 20.0

Поточна дисперсія: 20.12840319834216

Відносна похибка: 0.006379204404685781

=============================

Лямбда: 0.1

Очікуване середнє: 10.0

Поточне середнє: 9.83717313766225

Відносна похибка: 0.01655220052134259

Очікувана дисперсія: 10.0

Поточна дисперсія: 9.995045164852369

Відносна похибка: 0.000495729140379919

=============================

Лямбда: 0.2

Очікуване середнє: 5.0

Поточне середнє: 5.047954286646781

Відносна похибка: 0.009499746614907548

Очікувана дисперсія: 5.0

Поточна дисперсія: 4.996799547280523

Відносна похибка: 0.000640500522222968

=============================

Лямбда: 0.25

Очікуване середнє: 4.0

Поточне середнє: 3.959847985791018

Відносна похибка: 0.01013978676783997

Очікувана дисперсія: 4.0

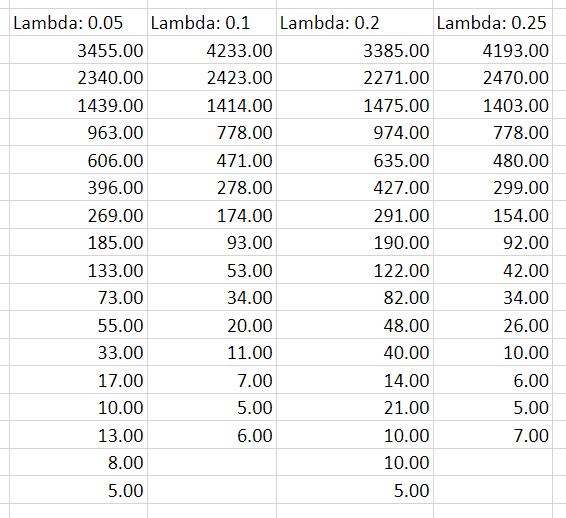
Поточна дисперсія: 4.015701017158519

Відносна похибка: 0.003909906910756255

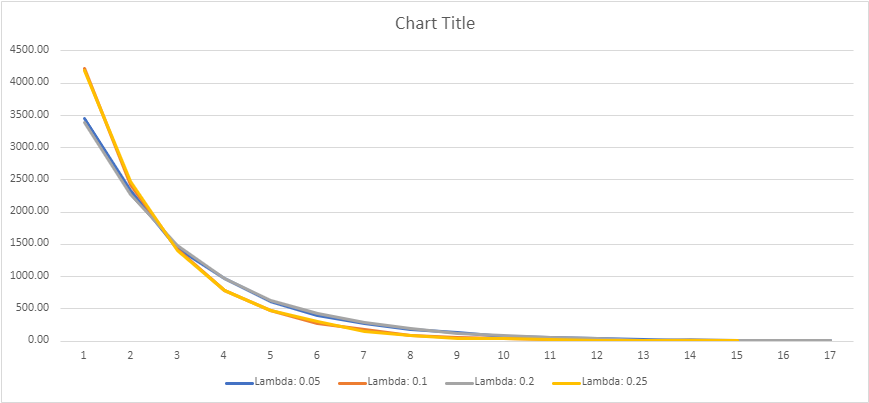
В усіх випадках похибка не перевищує 1.014%, що є задовільним у межах лабораторної роботи.

2. Зображення гістограми частот

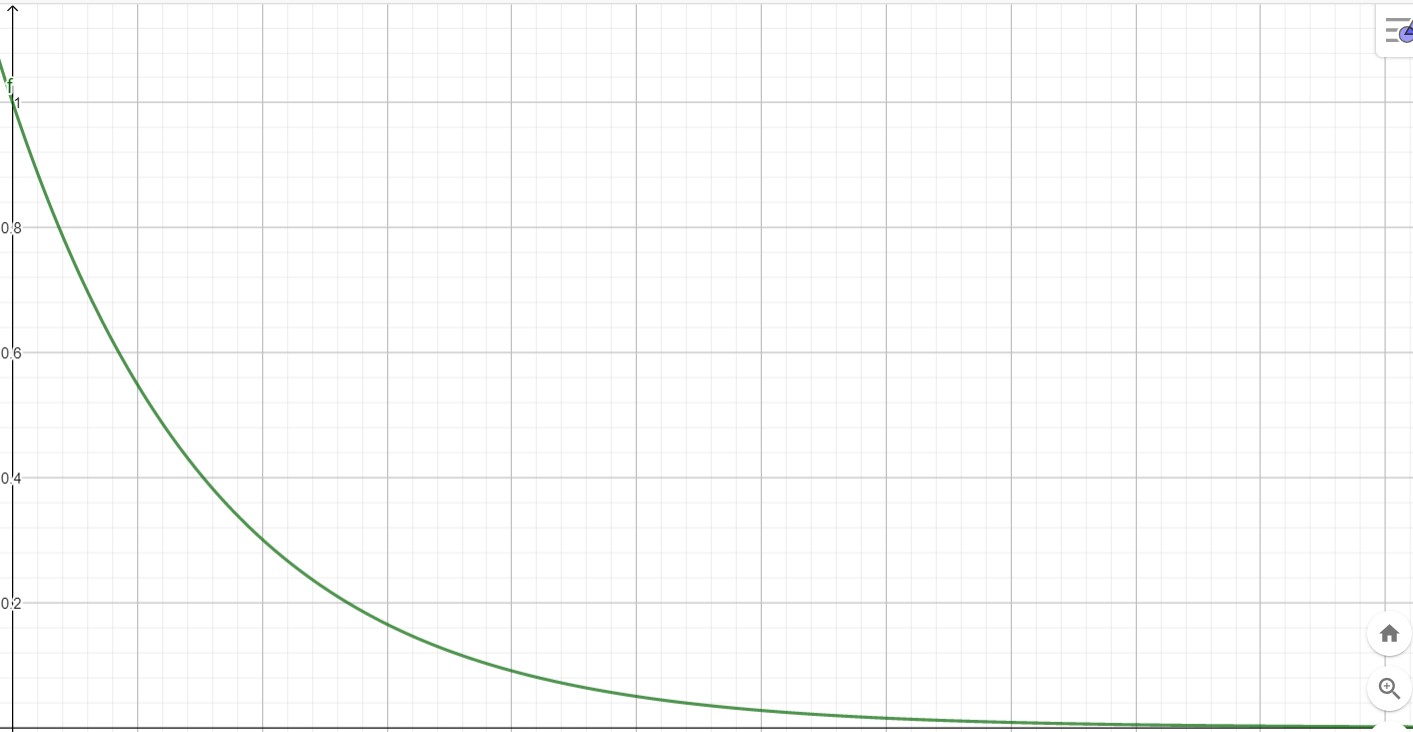
Графіки гістограм частот були побудовані за допомогою Excel:



Зображення графіків:



Графік експоненційної функції e-3x (функція взята приблизно для наглядності):



Таким чином, розподіл чисел ГВЧ набуває форми експоненійного закону розподілу.

3. Критерій згоди хі-квадрат

Згідно із теоретичними відомостями, наведеними у розділі 2.1, рівень значимості становить 0.05, а кількість степенів свободи рівна кількості інтервалів мінус 1 мінус кількість параметрів закону розподілу (згідно із завданням, їх всього один: “лямбда”).

Програма обчислила критерій згоди хі-квадрат та вивела результати тесту на екран:

=============================

Лямбда: 0.05

Значення тесту хі-квадрат: 13.90981416848492

Теоретично допустиме значення тесту хі-квадрат: 26.3

Тест пройдено: true

=============================

Лямбда: 0.1

Значення тесту хі-квадрат: 7.847460502975533

Теоретично допустиме значення тесту хі-квадрат: 13.7

Тест пройдено: true

=============================

Лямбда: 0.2

Значення тесту хі-квадрат: 9.35744100878377

Теоретично допустиме значення тесту хі-квадрат: 26.3

Тест пройдено: true

=============================

Лямбда: 0.25

Значення тесту хі-квадрат: 13.38909718980412

Теоретично допустиме значення тесту хі-квадрат: 13.7

Тест пройдено: true

В усіх випадках генератор випадкових чисел відповідає критерію згоди хі-квадрат відносно наведеного експоненційного закону розподілу.

**Висновки про відповідність закону розподілу**

Тест на середнє та дисперсію було пройдено із похибкою, що не перевищує 1.014%.

Графік гістограми розподілу чисел ГВЧ має форму, що нагадує експоненційний закон розподілу.

Критерій згоди хі-квадрат показав, що поточний закон розподілу відповідає наведеному експоненційному закону розподілу.

Вищенаведені фактори свідчать про те, що закон розподілу ГВЧ відповідає експоненційному закону розподілу.

**Висновки**

Під час виконання даної роботи було проведено аналіз закону розподілу ГВЧ. Усі критерії підтверджують припущення відповідності закону розподілу ГВЧ експоненційному. Для роботи було використано компільовану мову програмування Nim та програму для роботи із табличними даними Excel.