МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г.Шухова)

**Лабораторная работа № 2**

Дисциплина: Системный анализ и обработка информации

Тема: универсальный метод получения случайных величин для произвольного закона распределения СВ

Выполнил: студент группы ВТ-31

Ковалёв И. Д.

Проверил:

Полунин А. И.

Белгород 2020

**Цель работы**: изучить универсальный метод получения случайной величины для произвольного закона распределения случайной величины, реализовать датчик случайных чисел по заданному закону распределения.

Пусть

**Ход работы:**

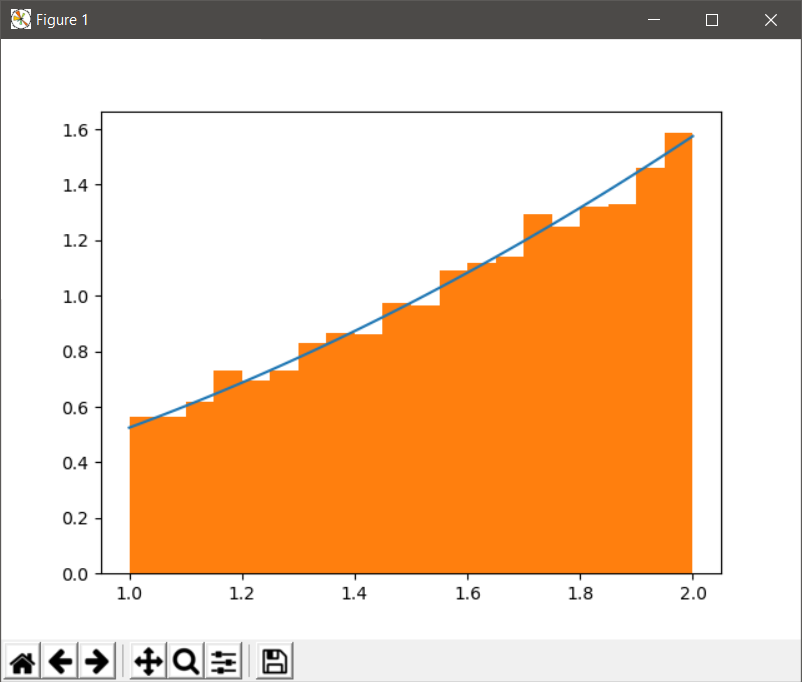
1. Чтобы получить функцию распределения случайной величины из имеющейся функции, нормируем ее по формуле

Тогда функция плотности вероятности будет иметь вид:

1. Для получения случайной величины необходимо решить такое уравнение:

где v – случайная величина, распределенная по равномерному закону, – требуемая случайная величина, распределенная по требуемому закону. Для применения численного метода решения (будет применен метод Ньютона) преобразуем к виду:

**Результаты работы программы:**



Текст программы:

# This Python file uses the following encoding: utf-8

import random

import math

import numpy

import scipy.integrate as integrate

import scipy.optimize as optimize

import matplotlib.pyplot as pyplot

sz = 100

a = 1

b = 2

result = []

arr = []

def fx(x):

return 4 \* x\*\*2 + 2 \* x + 1

def gx(x):

return (4 \* x\*\*2 + 2 \* x + 1) / k

def prob\_num(xi,v):

s1 = integrate.quad(fx, a, xi)[0]

return (s1/k - v)

if \_\_name\_\_ == "\_\_main\_\_":

k = integrate.quad(fx, a, b)[0]

for i in range(0, sz):

arr.append(numpy.random.random\_sample())

for i in arr:

def t(x):

return prob\_num(x,i)

solve = optimize.fsolve(t, b)

result.append(solve[0])

numpy.round(solve, 3, solve)

x\_axis = numpy.linspace(a, b)

fig, ax = pyplot.subplots(1, 1)

pyplot.plot(x\_axis, numpy.apply\_along\_axis(gx, 0, x\_axis))

pyplot.hist(result, 10, density=True)

pyplot.show()