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 №3
дисциплина «Системный анализ»
по теме «Аппроксимация функции по данным измерений методом наименьших квадратов с весовыми коэффицентами»

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Полунин А. И.

Белгород 2020

Лабораторная работа №3

«Аппроксимация функции по данным измерений методом наименьших квадратов с весовыми коэффицентами»

Цель работы:изучить методы аппроксимации при наличии ошибок в измерениях функции.

Задание: Производится измерение значений функции в некоторые заданные моменты времени. Вследствии погрешностей измерительной техники замер значения функции производится с ошибкой. Найти значения аппроксимирующего полинома методом наименьших квадратов и с помощью этого полинома найти значение функции в $X = 4.5$ и $X = 9.5$. сравнить эти значения с точными значениями функции в этих точках. Вычислить дисперсию интерполированных значений функции.

$$t = [1; 2; \ldots; 10; 11]$$

$$\phi_1, \phi_2, \ldots, \phi_k$$

$$\sigma_i = \sigma * i * 0,1$$

Ход работы

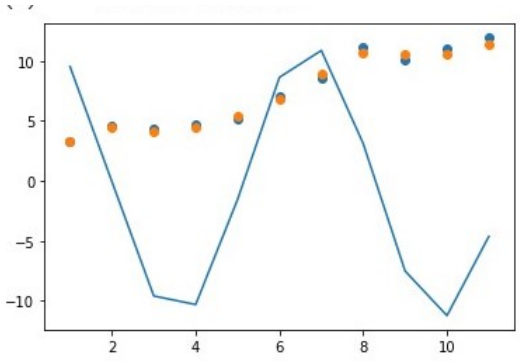

Рис. 1: График функции заданный в варианте 6

Полученные измерения полностью не соответствуют интерполированным значениям, для проверки корректности работы программы используем исходную функцию в качестве апроксимационного.

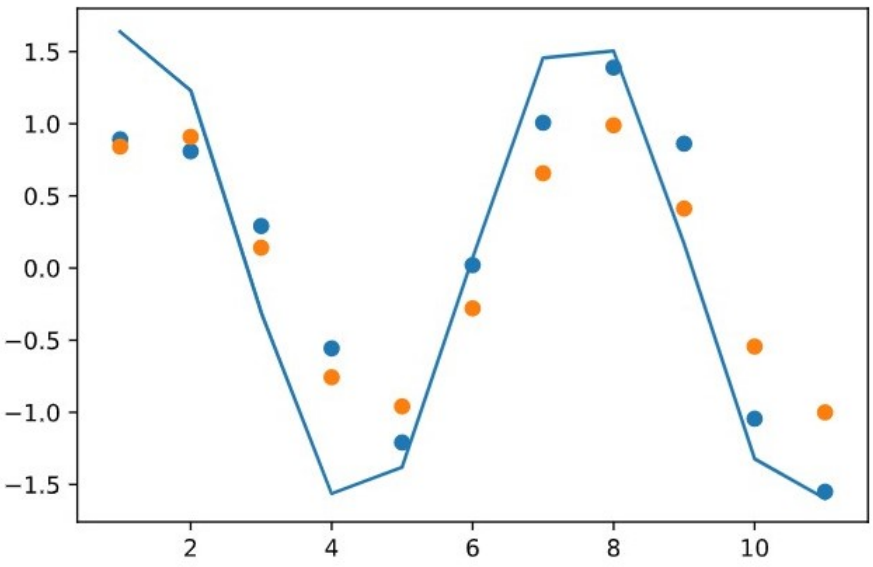

Рис. 2: Полученный график

Приложение

Содержимое файла main.py

```
import numpy as np
import math
import matplotlib.pyplot as pl
import scipy.integrate as integrate
import scipy.optimize as optimize
k = 19.833333333333336
a = 1
b = 2
count_of_num = 2000
arr = []
result_arr = []

def f(x):
    return (x**5+x**2+7)
def f_(x):
    return (x**5+x**2+7)/k

def prob_num(xi,v):
    s1 = integrate.quad(f,a,xi)[0]
    #print('integral =',s1,' v =',v,"int-v =",s1/k-v," xi =",xi)
    return (s1/k - v)


for i in range(0,count_of_num):
    arr.append(np.random.random_sample())

for i in arr:
    def t(x):
        return prob_num(x,i)
    solve = optimize.fsolve(t,b)
    result_arr.append(solve[0])
    #print("xi =",solve)
    np.round(solve,3,solve)
    #print("xi =",solve)
    #print('---')
x_axis = np.linspace(a,b)
fig,ax = pl.subplots(1,1)
pl.plot(x_axis,np.apply_along_axis(f_,0,x_axis))
pl.hist(result_arr,30,density=True)
pl.show()
```