**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Дисциплина: Системный анализ**

**Тема: Метод максимального правдоподобия**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: проф. ПО и ВТАС

Полунин Александр Иванович

**Белгород 2020**

**Цель лабораторной работы**: оценить, по данным измерений, неизвестные параметры системы методом максимального правдоподобия и определить точность этой оценки.

**Ход выполнения лабораторной работы**

**Вариант 9**

**Изображение выглядит как текст, доска

Автоматически созданное описание**

Задаем начальные значения вектора оцениваемых параметров

*.*

Интегрируем систему дифференциальных уравнений исследуемого процесса, получаем значения и в заданные моменты времени запоминаем значения математической модели вектора измерений Z.

Для нахождения величины подшагивания

на 1 шаге итерационного процесса при значении вектора оцениваемых параметров

выполняем следующее:

а) вычисляем обратную матрицу :

б) вычисляем матрицу *L* частных производных (i=1, 2), (j=1, 2,…N) методом конечных разностей. Для этого проводим два интегрирования системы дифференциальных уравнений при

, ,

где -величина вариации оцениваемых параметров равная 0,1. получаем значение функции и (i=1,2,…,N). Проводим еще два интегрирования при

,

.

Снова получаем значение функции и . Используя полученные данные, вычисляем методом конечных разностей элементы матрицы *L*

;

в) вычисляем вектор

,

где R-вектор экспериментальных значений, указанных в конкретном варианте задания; Z - вектор вычисленных значений математической модели при

*;*

г) проводим транспонирование матрицы *L*;

д) используя полученные данные, вычисляем вектор

е) вычисляем новое значение вектора оцениваемых параметров

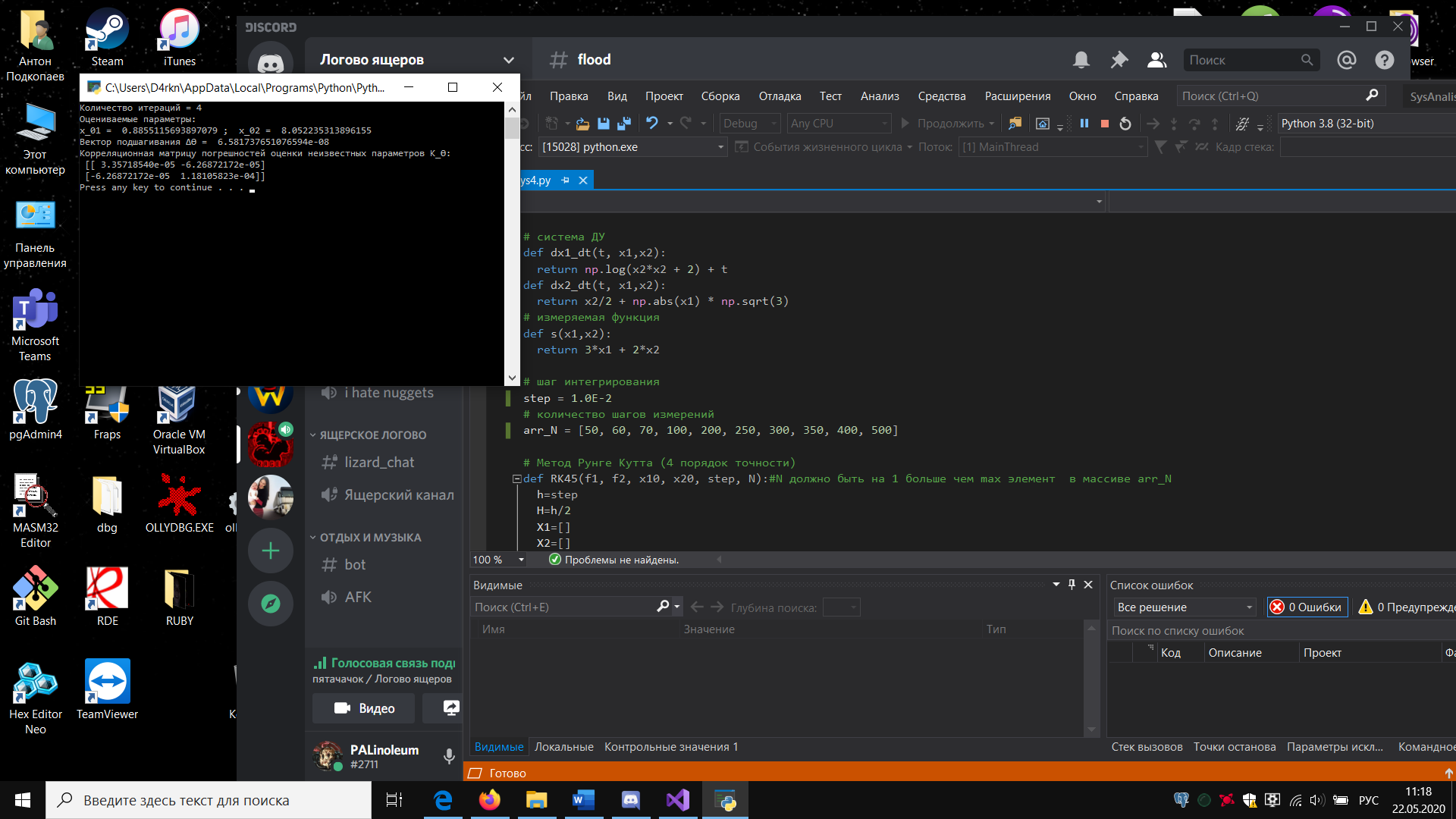
= +

Повторяем рассмотренную последовательность вычислений с новыми значениями вектора . Получаем

Процесс повторяем до тех пор, когда для очередного *j* – го шага для выполнится условие . Полученное значение

и будет вектором оцениваемых параметров. По имеющимся значениям параметров процесса вычисляем корреляционную матрицу погрешностей оценки неизвестных параметров

.



*Приложение*

import numpy as np

import matplotlib.pyplot as plt

import math

import scipy

import scipy.optimize as opt

import scipy.integrate as integrate

# вектор измерений

R=np.array([

[3.4937157447E+01],

[3.9217826917E+01],

[4.3946907873E+01],

[6.0966165241E+01],

[1.5956946183E+02],

[2.4303455087E+02],

[3.5896107983E+02],

[5.1724348631E+02],

[7.3046973513E+02],

[1.3928491165E+03],])

# диагональные элементы корреляционной матрицы

k11 = 4.8850591911E-05

k22 = 6.1568024980E-05

k33 = 7.7256954633E-05

k44 = 1.4869168126E-04

k55 = 1.0187841552E-03

k66 = 2.3623444723E-03

k77 = 5.1530493014E-03

k88 = 1.0699149788E-02

k99 = 2.1343240958E-02

k1010 = 7.7474502401E-02

# корреляционная матрица

Kv = np.zeros((10, 10))

di = np.diag\_indices\_from(Kv)

Kv[di] = [k11, k22, k33, k44, k55, k66, k77, k88, k99, k1010]

# дельта

delta = 0.000001

# оцениваемые параметры

x1 = 3

x2 = 10

x1add = x1+delta

x2add = x2+delta

x1sub = x1-delta

x2sub = x2-delta

# система ДУ

def dx1\_dt(t, x1,x2):

return np.log(x2\*x2 + 2) + t

def dx2\_dt(t, x1,x2):

return x2/2 + np.abs(x1) \* np.sqrt(3)

# измеряемая функция

def s(x1,x2):

return 3\*x1 + 2\*x2

# шаг интегрирования

step = 1.0E-2

# количество шагов измерений

arr\_N = [50, 60, 70, 100, 200, 250, 300, 350, 400, 500]

# Метод Рунге Кутта (4 порядок точности)

def RK45(f1, f2, x10, x20, step, N):#N должно быть на 1 больше чем max элемент в массиве arr\_N

h=step

H=h/2

X1=[]

X2=[]

x1 = [x10]

x2 = [x20]

for i in range(N):

k11 = f1((i+1)\*h, x1[-1], x2[-1])

k12 = f2((i+1)\*h, x1[-1], x2[-1])

k21 = f1((i+1)\*h + H, x1[-1] + H\*k11, x2[-1] + H\*k12)

k22 = f2((i+1)\*h + H, x1[-1] + H\*k11, x2[-1] + H\*k12)

k31 = f1((i+1)\*h + H, x1[-1] + H\*k21, x2[-1] + H\*k22)

k32 = f2((i+1)\*h + H, x1[-1] + H\*k21, x2[-1] + H\*k22)

k41 = f1((i+1)\*h + h, x1[-1] + h\*k31, x2[-1] + h\*k32)

k42 = f2((i+1)\*h + h, x1[-1] + h\*k31, x2[-1] + h\*k32)

x1.append(x1[-1] + (h/6)\*(k11 + 2\*k21 + 2\*k31 + k41))

x2.append(x2[-1] + (h/6)\*(k12 + 2\*k22 + 2\*k32 + k42))

for j in range(len(arr\_N)):

if( arr\_N[j]==(i+1) ):

X1.append(x1[-1] + (h/6)\*(k11 + 2\*k21 + 2\*k31 + k41))

X2.append(x2[-1] + (h/6)\*(k12 + 2\*k22 + 2\*k32 + k42))

return X1, X2

def Get\_vectors2():

NN = max(arr\_N)+1

s1\_add = []

X11,X22 = RK45(dx1\_dt, dx2\_dt, x1add, x2, step, NN)

for i in range(len(X11)):

s1\_add.append(s(X11[i], X22[i]))

s1\_sub = []

X11,X22 = RK45(dx1\_dt, dx2\_dt, x1sub, x2, step, NN)

for i in range(len(X11)):

s1\_sub.append(s(X11[i], X22[i]))

s2\_add = []

X11, X22 = RK45(dx1\_dt, dx2\_dt, x1, x2add, step, NN)

for i in range(len(X11)):

s2\_add.append(s(X11[i], X22[i]))

s2\_sub = []

X11,X22 = RK45(dx1\_dt, dx2\_dt, x1, x2sub, step, NN)

for i in range(len(X11)):

s2\_sub.append(s(X11[i], X22[i]))

ss= []

X11, X22 = RK45(dx1\_dt, dx2\_dt, x1, x2, step, NN)

for i in range(len(X11)):

ss.append(s(X11[i],X22[i]) )

return s1\_add, s1\_sub, s2\_add, s2\_sub, ss

def Get\_L(s1\_add,s1\_sub,s2\_add,s2\_sub):

L = []

ddq = 1/(2\*delta)

L = np.zeros((2, len(s1\_add) ))

for i in range( len(s1\_add) ):

tx1 = s1\_add[i]-s1\_sub[i]

tx2 = s2\_add[i]-s2\_sub[i]

tx1 = tx1\*ddq

tx2 = tx2\*ddq

L[0][i] = tx1

L[1][i] = tx2

return L

def Get\_a(Kv,L,dR):

a1 = np.dot(L,Kv)

a2 = np.dot(a1,L.transpose())

a3 = np.linalg.inv(a2) #K\_тетта - корреляционная матрица оцениваемых параметров

a4 = np.dot(a3,L)

a5 = np.dot(a4,Kv)

dq = a5.dot(dR)

return dq, a3

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

Kv = np.linalg.inv(Kv) # Kv в -1

k = 0

coun = 50

while(k < coun):

k = k + 1

s1\_add, s1\_sub, s2\_add, s2\_sub, ss = Get\_vectors2()

L = Get\_L(s1\_add,s1\_sub,s2\_add,s2\_sub)

dR = np.zeros((10,1))

for i in (range(len(ss))):

dR[i][0] = R[i][0]-ss[i]

a, K\_o = Get\_a(Kv,L,dR) #delta teta - величина подшагивания

md = np.sqrt(a[0][0]\*a[0][0]+a[1][0]\*a[1][0])

x1 = x1 + a[0][0]

x2 = x2 + a[1][0]

x1add = x1 + delta

x2add = x2 + delta

x1sub = x1 - delta

x2sub = x2 - delta

if(md < 10E-6): #если модуль вектора меньше погрешности

break

print("Количество итераций = {}".format(k))

print('Оцениваемые параметры:\nx\_01 = ', x1,'; x\_02 = ', x2)

print('Вектор подшагивания Δϴ = ', md)

print("Корреляционная матрицу погрешностей оценки неизвестных параметров K\_ϴ:\n", K\_o)