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 № 4**

Дисциплина: Системный анализ и обработка информации

Тема: «Метод максимального правдоподобия»

Выполнил: студент группы ВТ-32

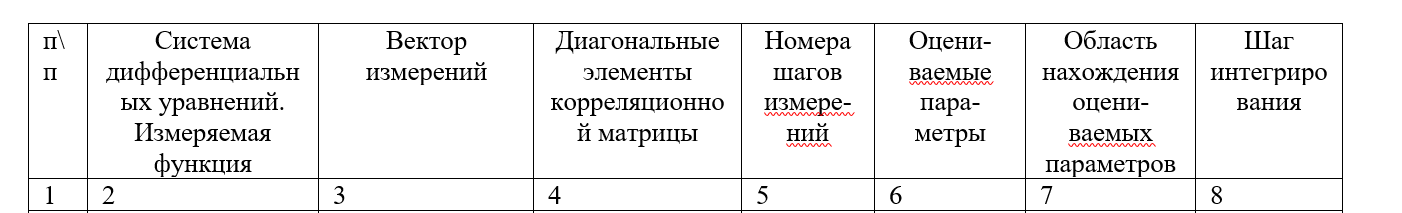
Воскобойников И. С.

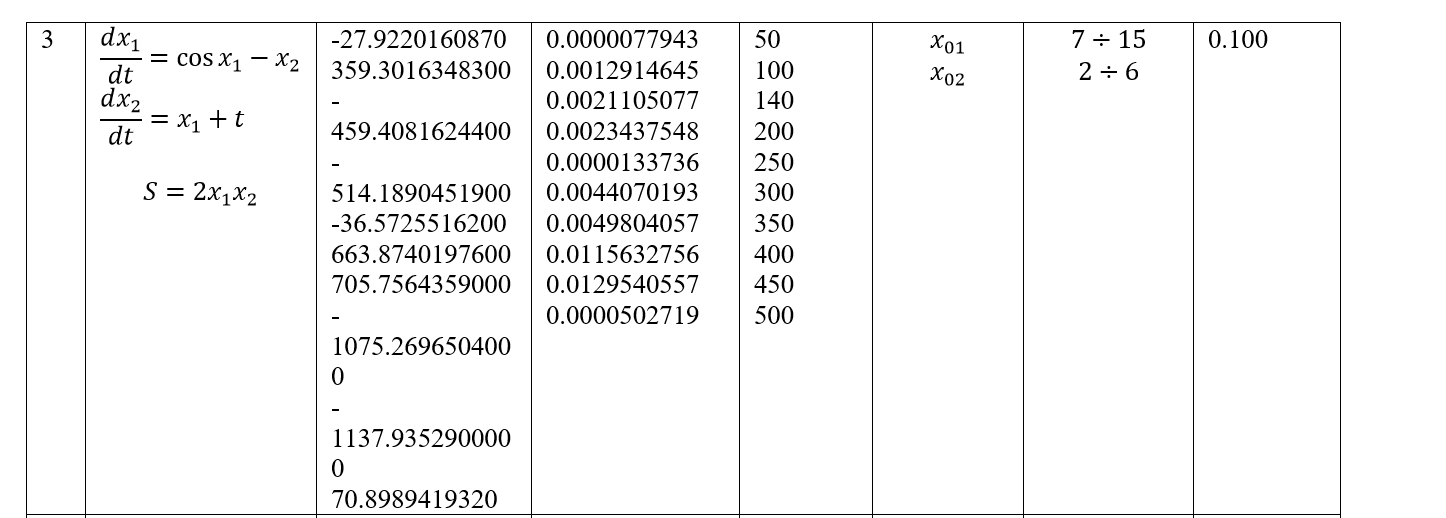
Проверил:

Полунин А. И.

Белгород 2021

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

**Задание: **

****

**Выполнение работы**

1. В первую очередь задаем начальные значение вектора оцениваемых параметров и вычисляем обратную матрицу
2. Найдем матрицу частных производных L, воспользовавшись методом конечных разностей.
3. Вычисляем вектор .
4. Вычисляем вектор подшагивания .
5. Находим новое значение вектора оцениваемых параметров, прибавив к нему новый вектор подшагивания.
6. В случае, если не абсолютное значение вектора подшагивания меньше, чем заданная точность , повторяем шаги 3-5, пока условие не будет выполнено.
7. Получив значения параметров процесса, вычисляем корреляционную матрицу погрешностей оценки неизвестных параметров: .

Результат работы программы:

****

**Вывод:** в процессе выполнения лабораторной работы был изучен метод максимального правдоподобия, который применим в случае, если существует необходимость найти математическую модель некоторого процесса, для которой известен вид самой модели, но неизвестны ее коэффициенты.

Листинг программы:

import numpy as np  
from scipy.integrate import ode  
from scipy.integrate import odeint  
import matplotlib.pyplot as plt  
from math import log, pow  
from scipy.integrate import solve\_ivp  
  
  
def dydt2(t, y):  
 y1, y2 = y  
 return [np.cos(y1) - y2,  
 y1 + t]  
  
  
dt = 0.100  
  
dv = 0.0001  
  
E = 0.0001  
  
  
def dydt(y, t):  
 y1, y2 = y  
 return [np.cos(y1) - y2,  
 y1 + t]  
  
  
def func(x1, x2):  
 return 2 \* x1 \* x2  
  
  
def minus(a: list, b: list, n: int):  
 res = []  
 for i in range(n):  
 res.append(a[i] - b[i])  
 return res  
  
  
def main():  
 *# t = np.linspace(0,5.01,501) # вектор моментов времени* t\_i = [50 \* dt,  
 100 \* dt,  
 140 \* dt,  
 200 \* dt,  
 250 \* dt,  
 300 \* dt,  
 350 \* dt,  
 400 \* dt,  
 450 \* dt,  
 500 \* dt]  
 Sx = [0 for i in range(10)]  
 y0 = [15, 7] *# начальное значение* R = [-2.7922016097E+01,  
 3.5930163485E+02,  
 -4.5940816254E+02,  
 -5.1418904539E+02,  
 -3.6572551705E+01,  
 6.6387402976E+02,  
 7.0575663100E+02,  
 -1.0752643306E+03,  
 -1.1381561796E+03,  
 7.0931662092E+01]  
 Kv = np.zeros([10, 10])  
 M = np.zeros([2, 10])  
 Kv[0, 0] = 1 / 7.7942838469E-06  
 Kv[1, 1] = 1 / 1.2914644653E-03  
 Kv[2, 2] = 1 / 2.1105077014E-03  
 Kv[3, 3] = 1 / 2.6437548373E-03  
 Kv[4, 4] = 1 / 1.3373627322E-05  
 Kv[5, 5] = 1 / 4.4070193469E-03  
 Kv[6, 6] = 1 / 4.9804057235E-03  
 Kv[7, 7] = 1 / 1.1563275556E-02  
 Kv[8, 8] = 1 / 1.2954055706E-02  
 Kv[9, 9] = 1 / 5.0271926642E-05  
 dR = [0 for i in range(10)]  
 buf1 = [0, 0]  
 buf2 = [0, 0]  
 buft = [0, 0]  
 res = np.array([100, 100])  
 buf = []  
 n = 0  
 while ((abs(res[0]) > E) and (abs(res[1]) > E) and (n < 100)):  
 ivp = solve\_ivp(dydt2, (0, 50.2), y0, t\_eval=t\_i, max\_step=dt) *# решение уравнения* for i in range(10):  
 Sx[i] = func(ivp.y[0, i], ivp.y[1, i])  
 print(Sx)  
 *# print(R)* dR = minus(R, Sx, 10)  
 print(dR)  
 buf2[0] = y0[0]  
 buf2[1] = y0[1]  
 for k in range(2):  
 for j in range(10):  
 buft[1] = t\_i[j]  
 *# print('t',j,'=',buft[1])* buf2[k] += dv  
 buf = odeint(dydt, buf2, buft)  
 *# print(buf)* buf1[0] = func(buf[1, 0], buf[1, 1])  
 buf2[k] -= 2 \* dv  
 buf = odeint(dydt, buf2, buft)  
 buf1[1] = func(buf[1, 0], buf[1, 1])  
 *# print(j,k,buf1)* buf2[k] += dv  
 M[k, j] = ((buf1[0] - buf1[1]) / (2 \* dv))  
 *# print(M[k,j])  
 # print('M',M)* buf3 = np.dot(M, Kv)  
 *# print('MKv',buf3)* buf4 = M.transpose()  
 buf3 = np.dot(buf3, buf4)  
 *# print('MKvM',buf3)* buf3 = np.linalg.inv(buf3)  
 *# print(buf3)* buf4 = np.dot(buf3, M)  
 *# print(buf4)* buf4 = np.dot(buf4, Kv)  
 *# print('ahfKV',buf4)* bR = np.array([dR])  
 bR = bR.transpose()  
  
 *# print(bR)* res = np.dot(buf4, dR)  
 print(res)  
 y0[0] += res[0]  
 y0[1] += res[1]  
 n += 1  
 print(n)  
 print(y0[0], y0[1])  
  
  
main()