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)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Системный анализ и обработка информации

Лабораторная работа №4

**Метод максимального правдоподобия**

Выполнил:

ст. гр. ВТ-31

Клесов М.И.

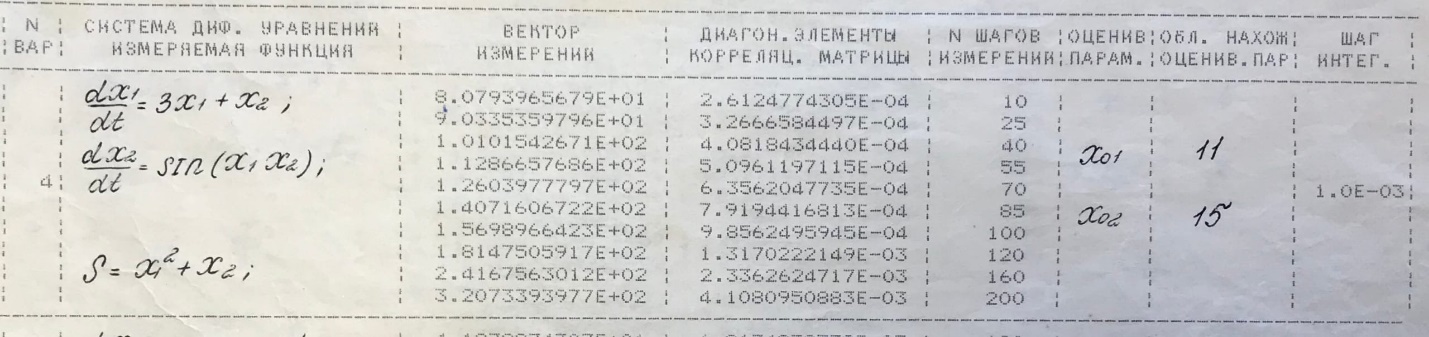
Проверил:

Полунин А.И.

Белгород, 2020

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

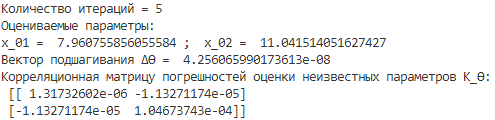
Задание варианта № 4



Алгоритм выполнения

1. Задаем начальные значения вектора оцениваемых параметров;
2. Вычисляем обратную матрицу
3. Вычисляем матрицу L частных производных методом конечных разностей;
4. Вычисляем вектор ;
5. Вычисляем вектор подшагивания ;
6. Вычисляем новое значение вектора оцениваемых параметров ;
7. Повторяем шаги 3 – 5 до тех пор, пока не выполнится условие ;
8. есть вектор оцениваемых параметров;
9. По имеющимся значениям параметров процесса вычисляем корреляционную матрицу погрешностей оценки неизвестных параметров .

Результат работы программы



Текст программы

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([

            [8.0793965679E+01],

            [9.0335359796E+01],

            [1.0101542671E+02],

            [1.1286657686E+02],

            [1.2603977797E+02],

            [1.4071606722E+02],

            [1.5698966423E+02],

            [1.8147505917E+02],

            [2.4167563012E+02],

            [3.2073393977E+02],])

# диагональные элементы корреляционной матрицы

k11 =   2.6124774305E-04

k22 =   3.2666584497E-04

k33 =   4.0818434440E-04

k44 =   5.0961197115E-04

k55 =   6.3562047735E-04

k66 =   7.9194416813E-04

k77 =   9.8562495945E-04

k88 =   1.3170222149E-03

k99 =   2.3362624717E-03

k1010 = 4.1080950883E-03

# 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 = 0.1

# оцениваемые параметры

x1 = 11

x2 = 15

x1add = x1+delta

x2add = x2+delta

x1sub = x1-delta

x2sub = x2-delta

# система ДУ

def dx1\_dt(t, x1,x2):

  return 3\*x1 + x2

def dx2\_dt(t, x1,x2):

  return np.sin(x1\*x2)

# измеряемая функция

def s(x1,x2):

  return x1\*\*2 + x2

# шаг интегрирования

step = 1.0E-3

# количество шагов измерений

arr\_N = [10, 25, 40, 55, 70, 85, 100, 120, 160, 200]

# 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)