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-31

Ковалёв И. Д.

Проверил:

Полунин А. И.

Белгород 2020

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

**Задание:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 6.6602020976E+00  7.5303301045E+00  8.5136259224E+00  9.6223224315E+00  1.2352396030E+01  2.3935782994E+01  4.7279362361E+01  9.4361804386E+01  1.6478350915E+02  2.5100574732E+02 | 1.7752908369E-06  2.2699495265E-06  2.8994128950E-06  3.7039807870E-06  6.1049911047E-06  2.2914082111E-05  8.9394910333E-05  3.5608336131E-04  1.0861339974E-03  2.5160425473E-03 | 50  60  70  80  100  150  200  250  290  320 |  | 4  8 | 1.0E-2 |

**Выполнение работы**

1. В первую очередь задаем начальные значение вектора оцениваемых параметров (4, 8) 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:

Количество итераций = 6

Оцениваемые параметры:

x\_01 = 1.9677543560940551 ; x\_02 = 4.959599584706487

Вектор подшагивания Δϴ = 3.5910279597516334e-09

Корреляционная матрицу погрешностей оценки неизвестных параметров K\_ϴ:

[[ 5.31656252e-08 -4.52675736e-07]

[-4.52675736e-07 8.02346058e-06]]

**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

import matplotlib.pyplot as plt

import math

import scipy

import scipy.optimize as opt

import scipy.integrate as integrate

R=np.array([

            [6.6602020976E+00],

            [7.5303301045E+00],

            [8.5136259224E+00],

            [9.6223224315E+00],

            [1.2352396030E+01],

            [2.3935782994E+01],

            [4.7279362361E+01],

            [9.4361804386E+01],

            [1.6478350915E+02],

            [2.5100574732E+02]])

k11 =   1.7752908369E-06

k22 =   2.2699495265E-06

k33 =   2.8994128950E-06

k44 =   3.7039807870E-06

k55 =   6.1049911047E-06

k66 =   2.2914082111E-05

k77 =   8.9394910333E-05

k88 =   3.5608336131E-04

k99 =   1.0861339974E-03

k1010 = 2.5160425473E-03

Kv = np.zeros((10, 10))

di = np.diag\_indices\_from(Kv)

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

delta = 0.1

x1 = 11

x2 = 15

x1add = x1+delta

x2add = x2+delta

x1sub = x1-delta

x2sub = x2-delta

def dx1\_dt(t, x1,x2):

  return math.sqrt(2 \* (x1 \*\* 2) + 3)

def dx2\_dt(t, x1,x2):

  return np.cos(x2) + x1 + t

def s(x1,x2):

  return x1 + x2 / 3

step = 1.0E-2

arr\_N = [50, 60, 70, 80, 100, 150, 200, 250, 290, 320]

def RK45(f1, f2, x10, x20, step, 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)

  a4 = np.dot(a3,L)

  a5 = np.dot(a4,Kv)

  dq = a5.dot(dR)

  return  dq, a3

Kv = np.linalg.inv(Kv)

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)

  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):

    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)