**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

Лабораторная работа №4

Дисциплина: Системный анализ

по теме Метод максимального правдоподобия(ММП)

Выполнил: ст. группы ВТ-31  
Новожен Н.В

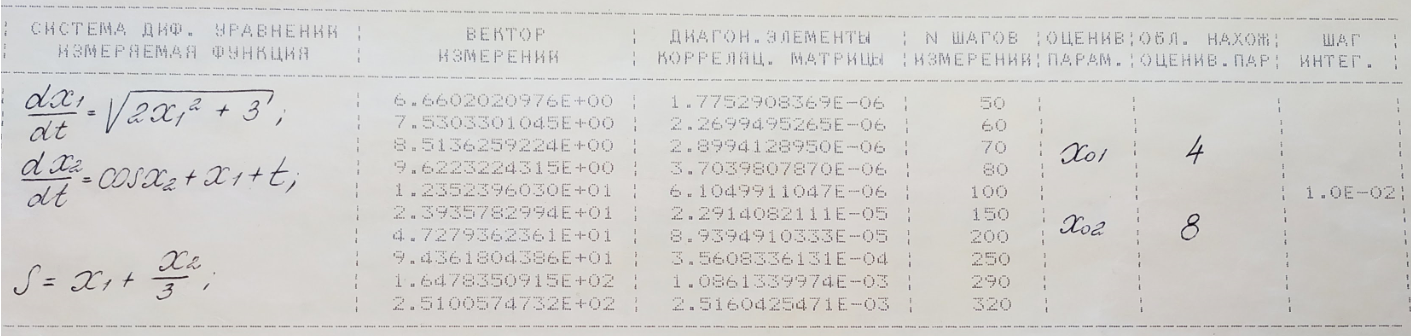
Проверил: Полунин А.И

**Белгород 2020**

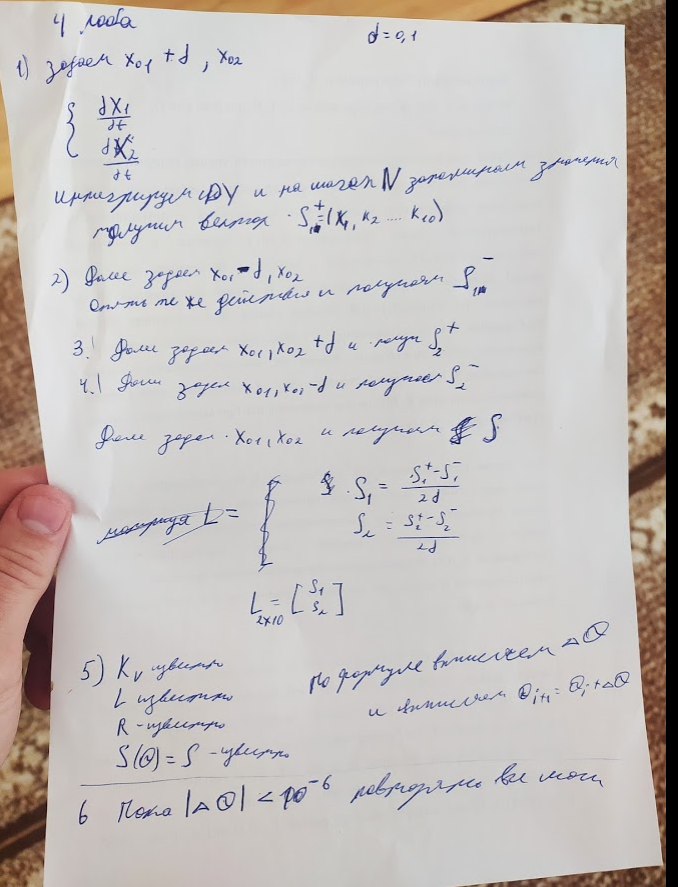
**Цель работы:** необходимо найти математическую модель процесса или системы, для которых известен вид математической модели, но неизвестны ее коэффициенты с помощью ММП

В качестве вектора наблюдений *Ri* (*i=1,2,…,N*) быть значения одной или нескольких выходных координат системы в известные моменты времени *ti*, либо некоторая функция от них. Математическую модель вектора измерений обозначим Z(ϴ), где ϴ - вектор оцениваемых параметров, состоящий из К элементов

**ВАРИАНТ 7**



**Алгоритм**

****

**Результат работы программы**

**md=== 4.23163481677644**

**x1= 6.15 x2= 6.15**

**md=== 0.23021045750033572**

**x1= 2.0340809017660186 x2= 5.167175906203282**

**md=== 0.012925341394151807**

**x1= 1.96799164185813 x2= 4.946655919654803**

**md=== 4.188995895716141e-05**

**x1= 1.9677515791613114 x2= 4.959579031511355**

**md=== 6.329538635821194e-09**

**x1= 1.9677516203420868 x2= 4.95962092145007**

**all itterate = 50 real itterate=5**

**x1= 1.9677516208597063 x2= 4.959620927758408**

**md=== 6.329538635821194e-09**

**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**

*##вектор измерений*

R=np.array([

[6.6602020976E+00],

[7.5303301045E+00],

[8.5136259224E+00],

[9.6223224315E+00],

[1.2352396030E+01],

[2.3935782934E+01],

[4.7279362361E+01],

[9.4361304386E+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=2.5160425471E-03

*##Совместил*

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

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

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

*###дельта*

d=0.1

*##цениваемые параметры*

x1=6.15

x2=6.15

*###*

x1add=x1+d

x2add=x2+d

x1sub=x1-d

x2sub=x2-d

*## система ДУ*

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

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

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

**return** np.cos(x2)+x1+t

*###*

*##измеряемая ф*

**def** s(x1,x2):

**return** x1+x2/3

*##шаг интегрирования*

step=1.0E-2

*###нужные шаги*

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

**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\*d)

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

*#print( tx1,tx2)*

**return** L

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

a1=np.dot(L,Kv)

*#print(np.linalg.det(Kv))*

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

a3=np.linalg.inv(a2)*#K0 -кор матр*

*#a3=np.linalg.pinv(a2)#K0 -кор матр*

a4=np.dot(a3,L)

a5=np.dot(a4,Kv)

dq=a5.dot(dR)

*#dq=((np.linalg.inv((L.dot(Kv)).dot(L.transpose())).dot(L)).dot(Kv)).dot(dR)*

**return** dq

**# MAIN**

*###*

*### Вперед решать*

*###*

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)*#L матрица производных*

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

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

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

*#### Находим вектор коэфициентов - a*

a=Get\_a(Kv,L,dR)

*### a найден*

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

print('md===',md)

print('x1=',x1,'x2=',x2)

x1=x1+a[0][0]*#x1+delta a*

x2=x2+a[1][0]*#x2+delta a*

x1add=x1+d

x2add=x2+d

x1sub=x1-d

x2sub=x2-d

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

**break**

print("all itterate = **{0}** real itterate=**{1}**".format(coun,k))

print('x1=',x1,'x2=',x2)

print('md===',md)

**md=== 4.23163481677644**

**x1= 6.15 x2= 6.15**

**md=== 0.23021045750033572**

**x1= 2.0340809017660186 x2= 5.167175906203282**

**md=== 0.012925341394151807**

**x1= 1.96799164185813 x2= 4.946655919654803**

**md=== 4.188995895716141e-05**

**x1= 1.9677515791613114 x2= 4.959579031511355**

**md=== 6.329538635821194e-09**

**x1= 1.9677516203420868 x2= 4.95962092145007**

**all itterate = 50 real itterate=5**

**x1= 1.9677516208597063 x2= 4.959620927758408**

**md=== 6.329538635821194e-09**