Федеральное агентство по образованию РФ

Государственное образовательное учреждение высшего профессионального образования

Белгородский Государственный Технологический Университет им В. Г. Шухова

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем.

Лабораторная работа №6

«Метод максимального правдоподобия»

**Дискретно-событийное моделирование**

Выполнение: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил:

студент группы ПВ-31

Адаменко И. И.

Проверил:

к. т. н., профессор

Полунин А. И.

Белгород  
2015

Лабораторная работа № 6

«Метод максимального правдоподобия»

**Цель работы:** исследовать точность оценки неизвестных параметров системы, по данным измерений, методом максимального правдоподобия (ММП).

## Ход работы

Исследуемый процесс определяется системой дифференциальных уравнений:

В процессе исследования с помощью аппаратных средств, в моменты времени  (i = 1, 2,…N), где N — число измерений, производятся замеры функции R, зависящей от переменных этой системы:

Измерения производятся со случайной ошибкой V, распределённой по нормальному закону. Параметры закона распределения известны. По этим измерениям необходимо оценить неизвестные параметры процесса. Неизвестными параметрами, вектор , могут быть начальные условия для системы уравнения, описывающих процесс, коэффициенты системы уравнений.

Для оценки неизвестных параметров воспользуемся методом максимального правдоподобия.

Разность вектора измерений и его математической модели S(, вычисленной для некоторого значения вектора :

Является вектором случайных величин, распределённым по нормальному закону:

Где — корреляционная матрица, определяющая взаимосвязь погрешностей измерений Vi.

Необходимо найти такое значение вектора отклонения измеренного вектора от вычисленного имело максимальную плотность вероятности P(). Максимум этой функции достигается при минимальном значении выражения

Воспользовавшись условием нахождения минимума, получаем систему k уравнений для вычисления k компонент вектора оценки .

Это уравнение называется уравнением правдоподобия. Для решения этой системы уравнений наиболее часто используется метод Ньютона, основанный на линеаризации функции S( относительно заданного значения в ряд Тейлора с учетом только линейных членов разложения:

Где N – матрица первых производных, вычисленная при

Подставим эту зависимость в уравнение правдоподобия

Или

Где

Отсюда

Мы получили формулу для вычисления величины подшагивания при нахождении вектора оценки определённых параметров . Вычислив первое , определяем

И вычисляем новые значения матрицы производных M при . Новое значение вектора и новое значение подшагивания Вычисления осуществляют до тех пор, пока величина невязки (т.е. модуль разности правой и левой части уравнения) не станет меньше заданной. Полученное значение и является требуемой оценкой определенных параметров.

**Расчетный алгоритм этой задачи имеет вид:**

1. Задаем начальные значения вектора оцениваемых параметров.
2. Интегрируем систему дифференциальных уравнений исследуемого процесса и в заданные моменты времени запоминаем значения функции S.
3. Для нахождения величины подшагивания где

Выполняем следующее:

* 1. Вычисляем обратную матрицу
  2. Вычисляем матрицу M — матрицу частных производных (k=1,2), (j = 1,2,…) методом конечных разностей. Для этого проводим два интегрирования системы дифференциальных уравнение при

Где  — величина вариации оцениваемых параметров — равна 0.0001. Получаем значение функции и (i = 1,2, …, N). Проводим еще два интегрирования при

Снова получаем значения функции и . Используя полученные данные, вычисляем матрицу M:

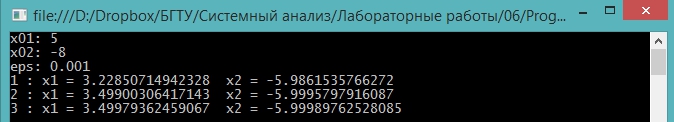
* 1. вычисляем вектор

Где R — вектор опытных значений, указанных в конкретном варианте задания; S — вектор вычисленных значений функции при

* 1. проводим транспонирование матрицы M
  2. используя полученные данные, вычисляем

1. Повторяем пп. 2, 3 с новыми значениями вектора до тех пор, пока .

## Пример работы программы



## Код программы

using System;

namespace Program

{

class Program

{

private static double rungeKuttStep = 0.001;

private static double d = 0.01;

private delegate double function(double t, double x1, double y2);

private static Matrix R, K;

private static double func1(double t, double x1, double x2)

{

return x1 \* x2 - t; // 2.1

}

private static double func2(double t, double x1, double x2)

{

return Math.Exp(-x1); // 2.2

}

private static void rungekuttNextStep(function func1, function func2,

double x0, double y0, int iColumn2Write,

ref Matrix Matr)

{

double t = 0, x1 = x0, y1 = y0;

int j = 0;

double k1, k2, k3, k4, m1, m2, m3, m4;

for (int i = 1; i <= 1000; i++) // max 5

{

t += rungeKuttStep;

k1 = func1(t, x1, y1);

m1 = func2(t, x1, y1);

k2 = func1(t + rungeKuttStep / 2, x1 + rungeKuttStep \* k1 / 2,

y1 + rungeKuttStep \* m1 / 2);

m2 = func2(t + rungeKuttStep / 2, x1 + rungeKuttStep \* k1 / 2,

y1 + rungeKuttStep \* m1 / 2);

k3 = func1(t + rungeKuttStep / 2, x1 + rungeKuttStep \* k2 / 2,

y1 + rungeKuttStep \* m2 / 2);

m3 = func2(t + rungeKuttStep / 2, x1 + rungeKuttStep \* k2 / 2,

y1 + rungeKuttStep \* m2 / 2);

k4 = func1(t + rungeKuttStep, x1 + rungeKuttStep \* k3,

y1 + rungeKuttStep \* m3);

m4 = func2(t + rungeKuttStep, x1 + rungeKuttStep \* k3,

y1 + rungeKuttStep \* m3);

x1 = x1 + rungeKuttStep \* (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;

y1 = y1 + rungeKuttStep \* (m1 + 2 \* m2 + 2 \* m3 + m4) / 6;

if ((i == 100) || (i == 200) || (i == 300) || (i == 400)

|| (i == 500) || (i == 600) || (i == 700)

|| (i == 800) || (i == 900) || (i == 1000)) // 5

{

Matr[j++, iColumn2Write] = x1 + y1; // 2.3

}

}

}

private static void Init()

{

R = new Matrix(10, 1);

K = new Matrix(10, 10);

// 3

R[0, 0] = -4.0748217692;

R[1, 0] = -4.9251244666;

R[2, 0] = -5.3650596829;

R[3, 0] = -5.5725335805;

R[4, 0] = -5.6489380647;

R[5, 0] = -5.6516106094;

R[6, 0] = -5.6127701050;

R[7, 0] = -5.5501597849;

R[8, 0] = -5.4735004615;

R[9, 0] = -5.3884929080;

// 4

K[0, 0] = 0.0000000017;

K[1, 1] = 0.0000000024;

K[2, 2] = 0.0000000029;

K[3, 3] = 0.0000000031;

K[4, 4] = 0.0000000032;

K[5, 5] = 0.0000000032;

K[6, 6] = 0.0000000032;

K[7, 7] = 0.0000000031;

K[8, 8] = 0.0000000030;

K[9, 9] = 0.0000000029;

}

private static void Run(double x01, double x02, double epsilon)

{

Matrix s, S1, S2, k1, M, L, dR, dQ, Tmp;

double d1, d2, x, y, eps;

int kh = 0;

s = new Matrix(10, 1);

S1 = new Matrix(10, 2);

S2 = new Matrix(10, 2);

M = new Matrix(10, 2);

dR = new Matrix(10, 1);

x = x01;

y = x02;

eps = epsilon;

k1 = ~K; kh = 0;

do

{

rungekuttNextStep(func1, func2, x, y, 0, ref s);

rungekuttNextStep(func1, func2, x + d, y, 0, ref S1);

rungekuttNextStep(func1, func2, x - d, y, 0, ref S2);

rungekuttNextStep(func1, func2, x, y + d, 1, ref S1);

rungekuttNextStep(func1, func2, x, y - d, 1, ref S2);

M = (S1 - S2) / (2 \* d);

L = !M;

dR = R - s;

Tmp = L \* k1;

dQ = (~(Tmp \* M)) \* Tmp \* dR;

d1 = dQ[0, 0]; d2 = dQ[1, 0];

x += d1; y += d2;

kh++;

Console.WriteLine(kh + " : x1 = " + x + " x2 = " + y);

} while ((Math.Abs(d1) > eps) && (Math.Abs(d2) > eps));

}

static void Main(string[] args)

{

Init();

Console.Write("x01: ");

double x01 = Convert.ToDouble(Console.ReadLine());

Console.Write("x02: ");

double x02 = Convert.ToDouble(Console.ReadLine());

Console.Write("eps: ");

double eps = Convert.ToDouble(Console.ReadLine());

Run(x01, x02, eps);

Console.ReadKey();

}

}

}