группа № в3530904-10022

Задание №4, 17 вариант

**«Параметрическая идентификация модели»**

1) Дана передаточная функция

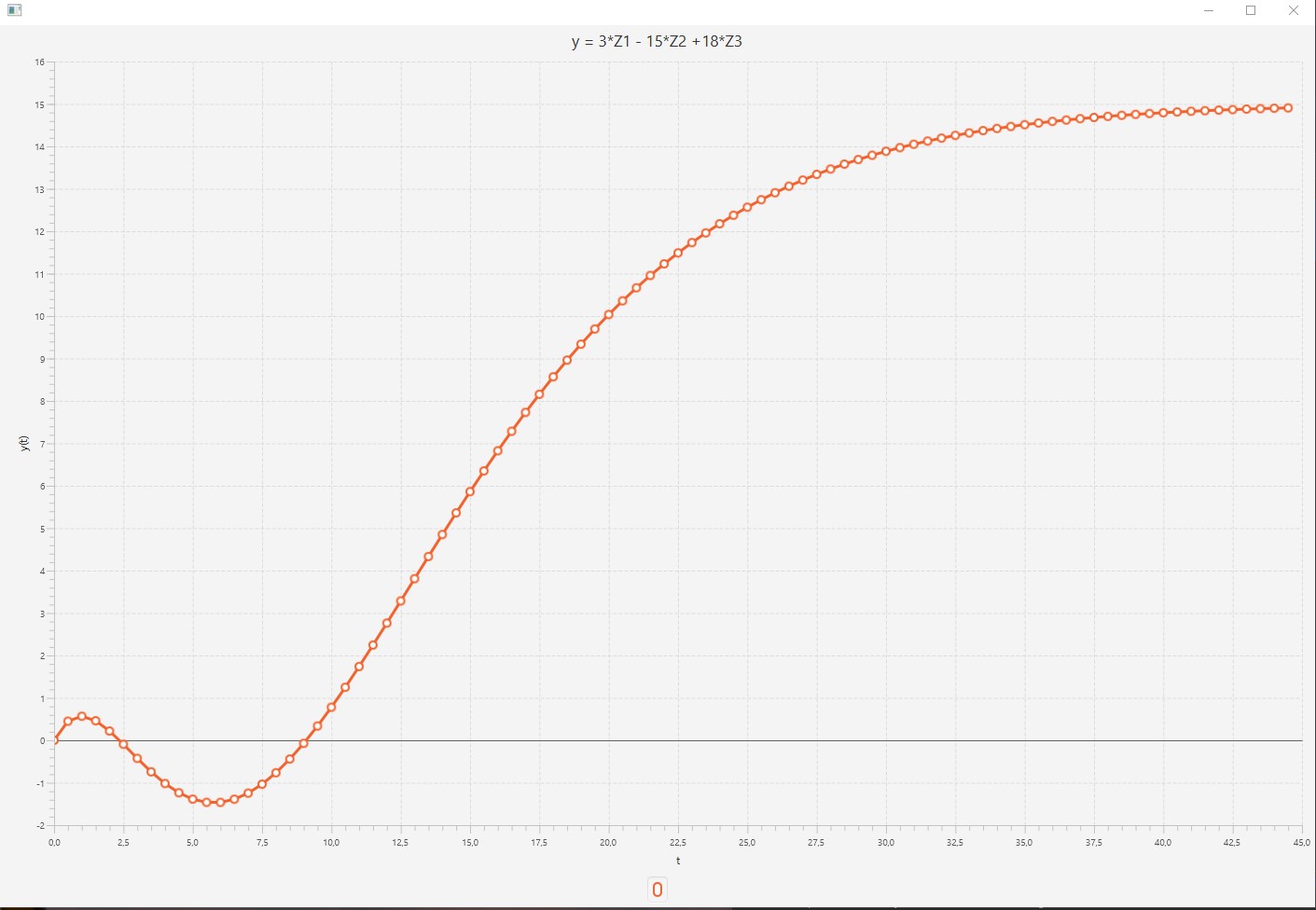
Раскрыв скобки и приведя подобные, получим:

Приведя к канонической форме решаем систему Эйлера:

На каждом шаге подставив полученные значения, рассчитываем:

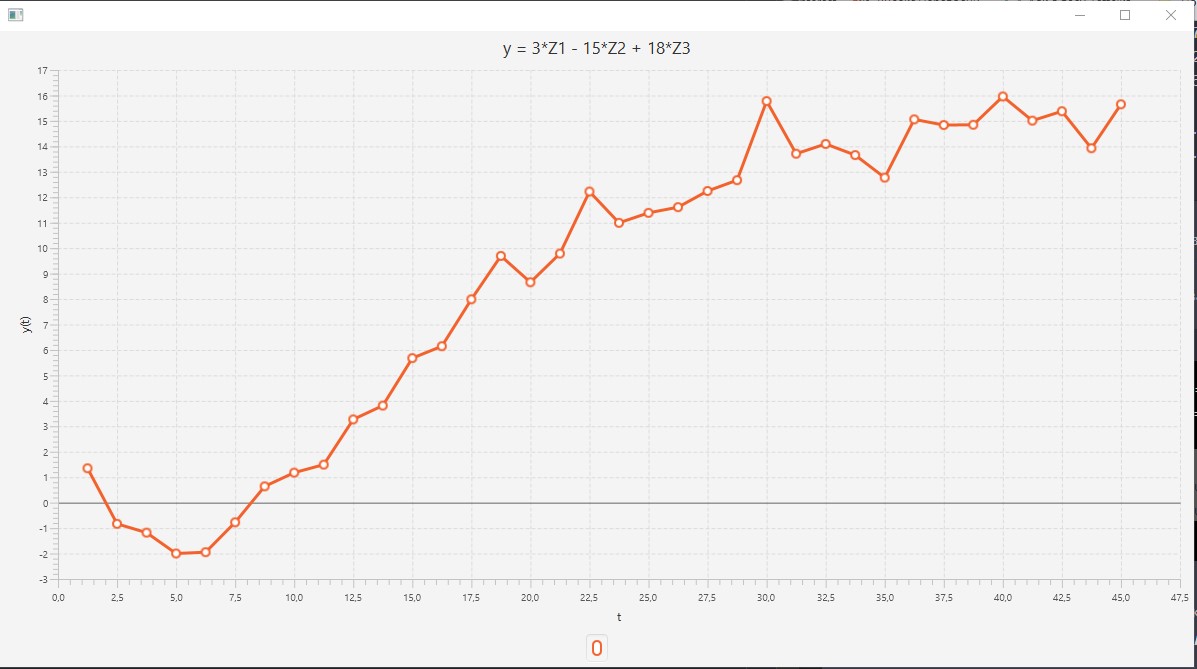
Реализовали передаточную функцию и получили таблицу значений y(t).

Полученный график:



2) При помощи генератора ГСЧ получаем «шумовую добавку» для всех точек таблицы и суммируем со значениями y(t).

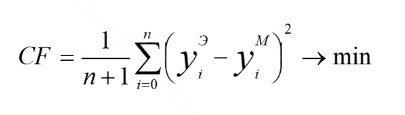
Получили зашумленный график – Y – экспериментальный:



**3) Параметризация модели**

Пусть в передаточной функции неизвестны два параметра – b1, b2.

Сформулируем целевую формулу:



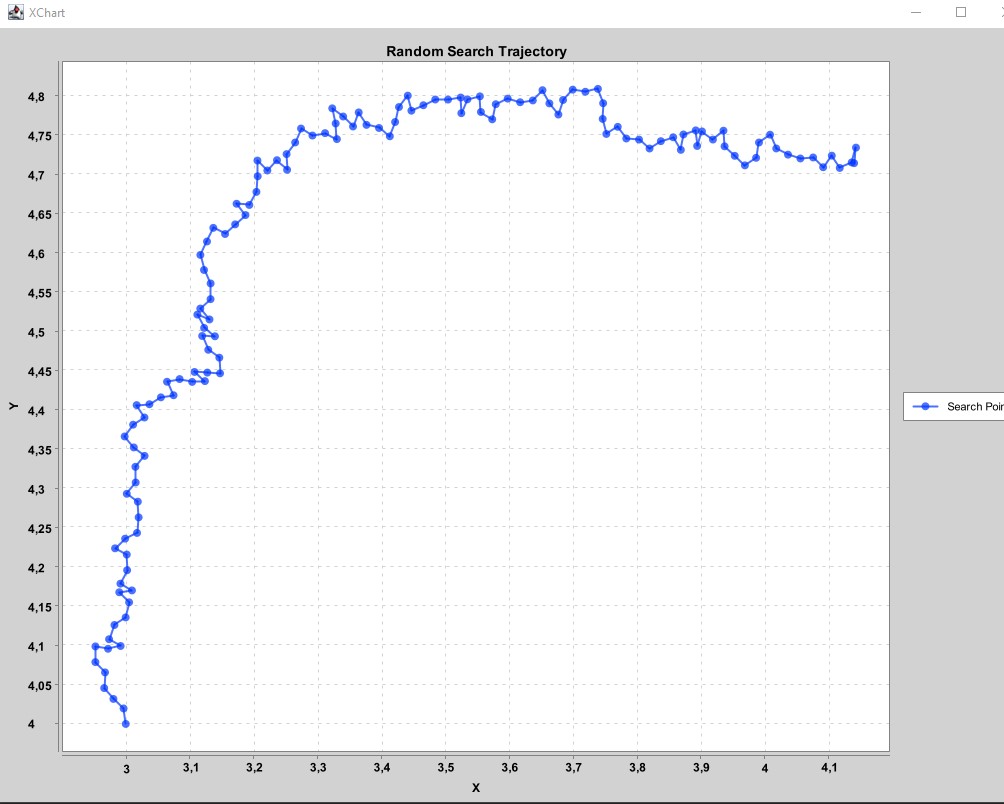
**Применяем метод оптимизации – случайного поиска.**

1) **Начальные точки задали : b1 = 3, b2 = 4**;

|  |
| --- |
| Iteration 1: x=3,0000, y=4,0000, value=1,40  Iteration 4: x1=2,9966, y2=4,0197, value=1,3452  Iteration 5: x1=2,9807, y2=4,0318, value=1,3206  Iteration 6: x1=2,9662, y2=4,0457, value=1,2909  Iteration 7: x1=2,9675, y2=4,0656, value=1,2363  Iteration 10: x1=2,9524, y2=4,0787, value=1,2107  Iteration 16: x1=2,9525, y2=4,0987, value=1,1595  Iteration 17: x1=2,9723, y2=4,0957, value=1,1557  Iteration 19: x1=2,9919, y2=4,0992, value=1,1354  Iteration 21: x1=2,9739, y2=4,1078, value=1,1243  Iteration 23: x1=2,9823, y2=4,1260, value=1,0753  Iteration 25: x1=2,9998, y2=4,1356, value=1,0426  Iteration 26: x1=3,0052, y2=4,1549, value=0,9950  Iteration 28: x1=2,9899, y2=4,1676, value=0,9747  ……  Iteration 259: x1=4,1055, y2=4,7236, value=0,0288  Iteration 260: x1=4,1180, y2=4,7079, value=0,0277  Iteration 261: x1=4,1367, y2=4,7150, value=0,0258  Iteration 264: x1=4,1434, y2=4,7338, value=0,0255  **Iteration 265: x1=4,1405, y2=****4,7140, value=0,0254** |

Метод минимизации определил точку минимума функции в точке - (4.1405, 4.7140)

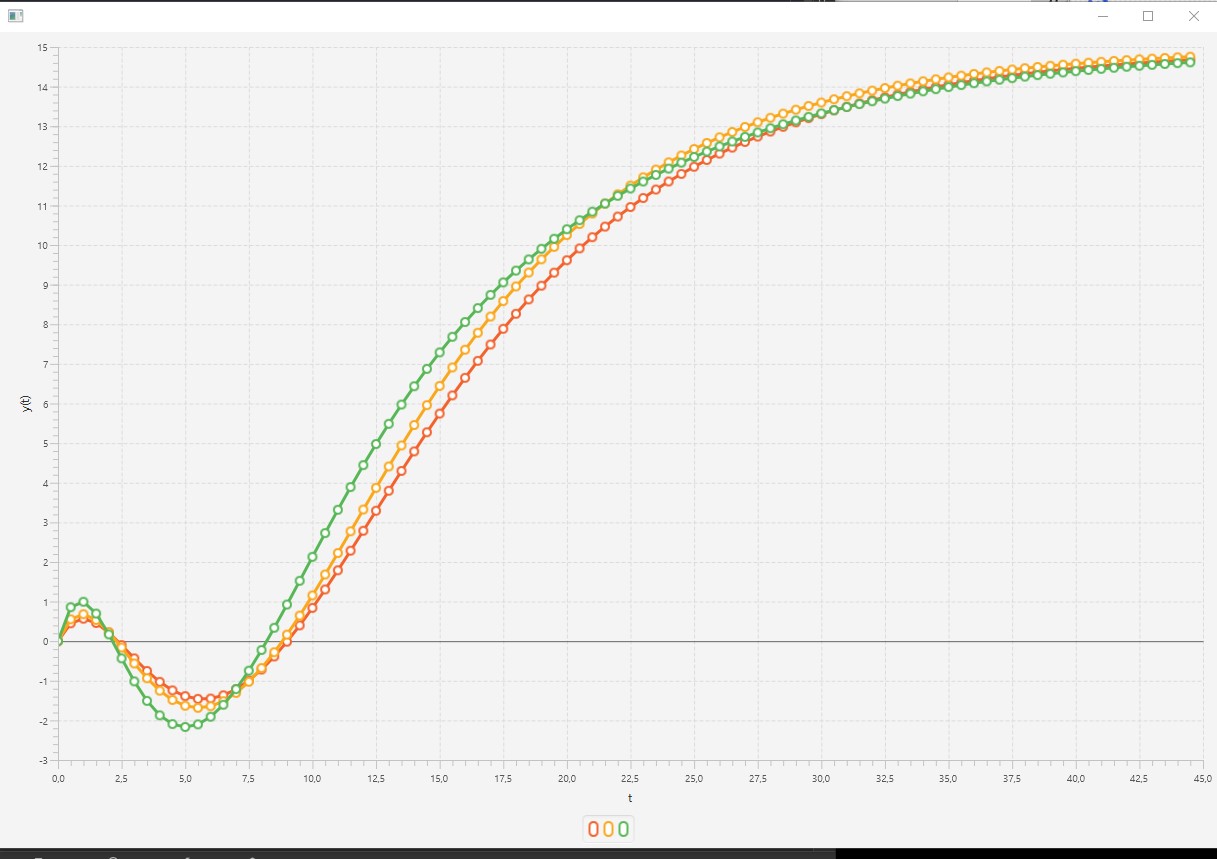
**График траектории движения к минимуму:**



3) Осуществили несколько попыток минимизации – для разных начальных точек. Таким образом, в результате параметрической идентификации получены следующие значения:

|  |  |  |
| --- | --- | --- |
|  | **Начальная точка** | **Полученные значения** |
| **B1** | 3.5 | 3.7 |
| **B2** | 4.5 | 4.7 |
| **CF** |  | 0,0833 |
|  |  |  |
| **B1** | 2.0 | 2,4106 |
| **B2** | 3.0 | 4,6721, |
| **CF** |  | 0,5799 |
|  |  |  |
| **B1** | 7.0 | 4,7798 |
| **B2** | 8.0 | 4,6409 |
| **CF** |  | 0,0057 |
|  |  |  |

Построили графики y(t) на одних осях – с исходным эталонным значением (красный) и графики для найденных при помощи поискового метода параметров:



Выводы: В данной работе мы выполнили параметрическую идентификацию методом случайного поиска по зашумленному массиву данных.

С помощью алгоритма случайного поиска мы нашли оптимальные значения этих параметров b1 и b2, чтобы минимизировать разницу между массивом с шумом и массивами с разными приближенными значениями параметров b1 b2. Получили траекторию движиения к минимуму и графики.

**Код:**

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import org.knowm.xchart.\*;

import org.knowm.xchart.style.lines.SeriesLines;

import org.knowm.xchart.style.markers.SeriesMarkers;

public class RandomSearch {

public static double func(double b1, double b2) {

double[] arr = {

-0.5302,

0.45,

0.5663,

0.4602,

0.2183,

-0.0937,

-0.4262,

-0.7433,

-1.0194,

-1.2375,

-1.387,

-1.4627,

-1.4629,

-1.3891,

-1.2448,

-1.0348,

-0.7651,

-0.4421,

-0.0725,

0.3369,

0.7795,

1.2491,

1.7397,

2.2459,

2.7626,

3.2852,

3.8097,

4.3325,

4.8504,

5.3607,

5.861,

6.3495,

6.8245,

7.2848,

7.7292,

8.1571,

8.568,

8.9614,

9.3372,

9.6955,

10.0362,

10.3597,

10.6663,

10.9563,

11.2302,

11.4885,

11.7317,

11.9604,

12.1752,

12.3767,

12.5655,

12.7422,

12.9075,

13.0618,

13.2058,

13.34,

13.4651,

13.5815,

13.6897,

13.7903,

13.8837,

13.9703,

14.0507,

14.1252,

14.1941,

14.258,

14.317,

14.3716,

14.4221,

14.4687,

14.5117,

14.5514,

14.5879,

14.6217,

14.6528,

14.6814,

14.7077,

14.732,

14.7543,

14.7748,

14.7937,

14.811,

14.8269,

14.8415,

14.8549,

14.8672,

14.8785,

14.8889,

14.8984,

14.9071};

double x = 5;

double k = 3;

double a2 = 2;

double a1 = 3;

// double b1 = 4;

// double b2 = 5;

double st = 0;

double z1 = 0;

double z2 = 0;

double z3 = 0;

double dz1 = 0;

double dz2 = 0;

double dz3 = 0;

double h = 0.5;

double y = 0;

for (int i = 0; i < 90; i++) {

dz1 = z1 + h \* z2;

dz2 = z2 + h \* z3;

dz3 = z3 + h \* (x - z1 - z2\*(b2 + 2\*b2) -z3\*(b2\*b2 + 2\*b1\*b2))/(b1\*b2\*b2);

y = k\*(z1 - a2\*z2 - a1\*z2 + a1\*a2\*z3);

st = st + Math.pow(arr[i] - y, 2);

z1 = dz1;

z2 = dz2;

z3 = dz3;

}

st = st / 90;

return st;

}

public static List<double[]> randomSearch(double lowerBoundX, double upperBoundX, double lowerBoundY, double upperBoundY, int iterations, double[] startPoint, double stepSize, double epsilon) {

List<double[]> searchPoints = new ArrayList();

Random random = new Random();

double bestX = startPoint[0];

double bestY = startPoint[1];

double bestValue = func(bestX, bestY);

searchPoints.add(new double[]{bestX, bestY});

System.out.printf("Iteration 1: x=%.4f, y=%.4f, value=%.2f%n", bestX, bestY, bestValue);

for (int i = 1; i < iterations; i++) {

double angle = random.nextDouble() \* 2 \* Math.PI;

double x = bestX + stepSize \* Math.cos(angle);

double y = bestY + stepSize \* Math.sin(angle);

double currentValue = func(x, y);

if (currentValue < bestValue) {

double diff = bestValue - currentValue;

if (diff < epsilon) {

break;

}

bestX = x;

bestY = y;

bestValue = currentValue;

searchPoints.add(new double[]{bestX, bestY});

System.out.printf("Iteration %d: x1=%.4f, y2=%.4f, value=%.4f%n", i + 1, bestX, bestY, bestValue);

} else if (i == iterations - 1 && searchPoints.size() == 1) {

System.out.println("Search ended: no better point ");

break;

} else if (i == iterations - 1) {

System.out.println("Search ended: limit reached.");

break;

}

}

return searchPoints;

}

public static void main(String[] args) {

double lowerBoundX = -1000;

double upperBoundX = 1000;

double lowerBoundY = -1000;

double upperBoundY = 1000;

int iterations = 10000;

double[] startPoint = {7, 8};

double stepSize = 0.02;

double epsilon = 0.0001;

List<double[]> searchPoints = randomSearch (lowerBoundX, upperBoundX, lowerBoundY, upperBoundY, iterations, startPoint, stepSize, epsilon);

double[] xData = new double[searchPoints.size()];

double[] yData = new double[searchPoints.size()];

for (int i = 0; i < searchPoints.size(); i++) {

xData[i] = searchPoints.get(i)[0];

yData[i] = searchPoints.get(i)[1];

}

// Создание и отображение графика

XYChart chart = new XYChartBuilder().width(800).height(600).title("Random Search Trajectory").xAxisTitle("X").yAxisTitle("Y").build();

XYSeries series = chart.addSeries("Search Points", xData, yData);

series.setMarker(SeriesMarkers.CIRCLE);

series.setLineStyle(SeriesLines.SOLID);

new SwingWrapper<XYChart>(chart).displayChart();

}

}