Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: «Информатика и системы управления»
КАФЕДРА: «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №9
«Поиск экстремума функции методом покоординатного спуска»

по курсу
«Разработка мобильных приложений»

Выполнил:
студент группы ИУ9-72Б
Караник А.А.

Проверено:
Посевин Д.П.

Москва, 2024

Цель работы

Реализовать поиск экстремума функции методом покоординатного спуска на Flutter с визуализацией графика функции $f(x, y) = (x + a)^2 + (y + b)^2$ с помощью библиотеки Ditredi. Добавить возможность ввода параметров a, b, начальной точки x_0 и y_0 для наглядного отображения траектории спуска.

Реализация

Исходный код:

```
import 'package:ditredi/ditredi.dart';
import 'package:flutter/material.dart';
import 'package:vector_math/vector_math_64.dart' show Vector3;
import 'dart:math';

class SurfacePlot extends StatefulWidget {
  @override
  _SurfacePlotState createState() => _SurfacePlotState();
}

class _SurfacePlotState extends State<SurfacePlot> {
  final diTreDiController = DiTreDiController()
    ..userScale = 2
    ..rotationX = -90
    ..rotationY = 0
    ..rotationZ = 12.5;

  List<Point3D> fullPath = [];
  double sliderValue = 0;

  double x0 = 4.0;
  double y0 = 2.0;
  double eps = 0.01;

  double xn = 0;
  double yn = 0;
  double zn = 0;

  double a = -2;
  double b = -2;

  final TextEditingController x0Controller = TextEditingController();
  final TextEditingController y0Controller = TextEditingController();
  final TextEditingController aController = TextEditingController();
  final TextEditingController bController = TextEditingController();

  @override
  void initState() {
    super.initState();

    x0Controller.text = x0.toString();
    y0Controller.text = y0.toString();
    aController.text = a.toString();
    bController.text = b.toString();

    fullPath = _coordinateDescentPath(
        Vector3(x0, y0, _function(x0, y0, a, b)), a, b, eps);
  }

  void _handleScaleUpdate(ScaleUpdateDetails details) {
    setState(() {
```

```
    if (details.scale != 1.0) {
      diTreDiController.userScale += (details.scale - 1) * 0.01;
      diTreDiController.userScale = diTreDiController.userScale.clamp(0.5, 10.0);
    } else {
      diTreDiController.rotationX += details.focalPointDelta.dy * 0.5;
      diTreDiController.rotationZ += details.focalPointDelta.dx * 0.5;
    }
  });
}

void _updateVariables() {
  setState(() {
    x0 = double.tryParse(x0Controller.text) ?? x0;
    y0 = double.tryParse(y0Controller.text) ?? y0;
    a = double.tryParse(aController.text) ?? a;
    b = double.tryParse(bController.text) ?? b;

    fullPath = _coordinateDescentPath(
        Vector3(x0, y0, _function(x0, y0, a, b)), a, b, eps);
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('lab9'),
    ),
    body: Column(
      children: [
        Expanded(
          child: Center(
            child: GestureDetector(
              onScaleUpdate: _handleScaleUpdate,
              child: DiTreDi(
                controller: diTreDiController,
                figures: [
                  ..._generatePointSurface(),
                  ..._generateAxes(),
                  ..._getPathPoints(),
                  ..._extraLines(),
                  ..._extraPoints(),
                ],
              ),
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            children: [
              Text("Прогресс покоординатного спуска"),
              Slider(
                value: sliderValue,
                min: 0,
                max: fullPath.length.toDouble() - 1,
                divisions: fullPath.length - 1,
                label: sliderValue.toInt().toString(),
                onChanged: (value) {
                  setState(() {
                    sliderValue = value;
                  });
                },
              ),
              SizedBox(height: 20),
```

```
              Row(
                children: [
                  Expanded(
                    child: TextField(
                      controller: x0Controller,
                      keyboardType: TextInputType.number,
                      decoration: InputDecoration(labelText: 'x0'),
                    ),
                  ),
                  SizedBox(width: 10),
                  Expanded(
                    child: TextField(
                      controller: y0Controller,
                      keyboardType: TextInputType.number,
                      decoration: InputDecoration(labelText: 'y0'),
                    ),
                  ),
                ],
              ),
              SizedBox(height: 10),
              Row(
                children: [
                  Expanded(
                    child: TextField(
                      controller: aController,
                      keyboardType: TextInputType.number,
                      decoration: InputDecoration(labelText: 'a'),
                    ),
                  ),
                  SizedBox(width: 10),
                  Expanded(
                    child: TextField(
                      controller: bController,
                      keyboardType: TextInputType.number,
                      decoration: InputDecoration(labelText: 'b'),
                    ),
                  ),
                ],
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: _updateVariables,
                child: Text("Обновить"),
              ),
            ],
          ),
        ),
      ],
    ),
  );
}

List<Point3D> _generatePointSurface() {
  List<Point3D> points = [];
  double step = 0.1;

  double deltaX = 3.0;
  double deltaY = 3.0;
  double maxZ = 5.0;

  for (double x = -a - deltaX; x <= -a + deltaX; x += step) {
    for (double y = -b - deltaY; y <= -b + deltaY; y += step) {
      double z = _function(x, y, a, b);
      if (z <= maxZ) {
        points.add(Point3D(Vector3(x, y, z), color: Colors.blue));
```

```
      }
    }
  }
  return points;
}

List<Line3D> _generateAxes() {
  return [
    Line3D(Vector3(0, 0, 0), Vector3(20, 0, 0),
        color: Colors.red, width: 2.0),
    Line3D(Vector3(0, 0, 0), Vector3(0, 20, 0),
        color: Colors.green, width: 2.0),
    Line3D(Vector3(0, 0, 0), Vector3(0, 0, 20),
        color: Colors.blue, width: 2.0),
  ];
}

double _function(double x, double y, double a, double b) {
  return pow(x + a, 2).toDouble() + pow(y + b, 2).toDouble();
}

List<Point3D> _getPathPoints() {
  int pointCount = sliderValue.toInt();
  return fullPath.sublist(0, pointCount + 1);
}

List<Line3D> _extraLines() {
  int pointCount = sliderValue.toInt();
  return [
    Line3D(
        fullPath[pointCount].position,
        Vector3(fullPath[pointCount].position.x,
            fullPath[pointCount].position.y, 0),
        color: Colors.amber,
        width: 2.0),
  ];
}

List<Point3D> _extraPoints() {
  int pointCount = sliderValue.toInt();
  return [
    Point3D(
        Vector3(fullPath[pointCount].position.x,
            fullPath[pointCount].position.y, 0),
        color: Colors.black,
        width: 3.0),
  ];
}

List<Point3D> _coordinateDescentPath(
    Vector3 start, double a, double b, double epsilon) {
  List<Point3D> path = [];
  Vector3 current = start;
  double currentValue = _function(current.x, current.y, a, b);
  path.add(Point3D(current, color: Colors.red, width: 3.0));

  double step = 0.1;

  while (true) {
    List<Vector3> directions = [
      Vector3(current.x + step, current.y,
          _function(current.x + step, current.y, a, b)),
      Vector3(current.x - step, current.y,
          _function(current.x - step, current.y, a, b)),
      Vector3(current.x, current.y + step,
```

```
            _function(current.x, current.y + step, a, b)),
        Vector3(current.x, current.y - step,
            _function(current.x, current.y - step, a, b)),
      ];

      directions.sort((a, b) => a.z.compareTo(b.z));
      Vector3 next = directions.first;

      double nextValue = _function(next.x, next.y, a, b);
      if ((currentValue - nextValue).abs() < epsilon) {
        path.add(Point3D(next, color: Colors.green, width: 6.0));
        xn = next.x;
        yn = next.y;
        zn = _function(xn, yn, a, b);
        break;
      }

      path.add(Point3D(next, color: Colors.red, width: 3.0));
      current = next;
      currentValue = nextValue;
    }
    return path;
  }
}

void main() {
  runApp(MaterialApp(
    home: SurfacePlot(),
  ));
}
```

Результаты

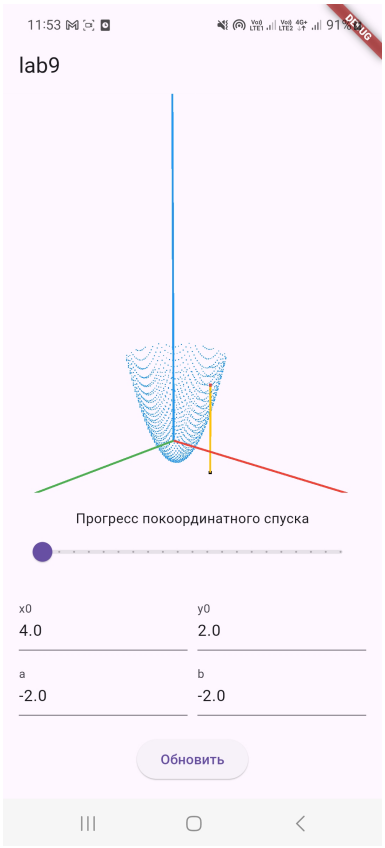

Рис. 1: результаты

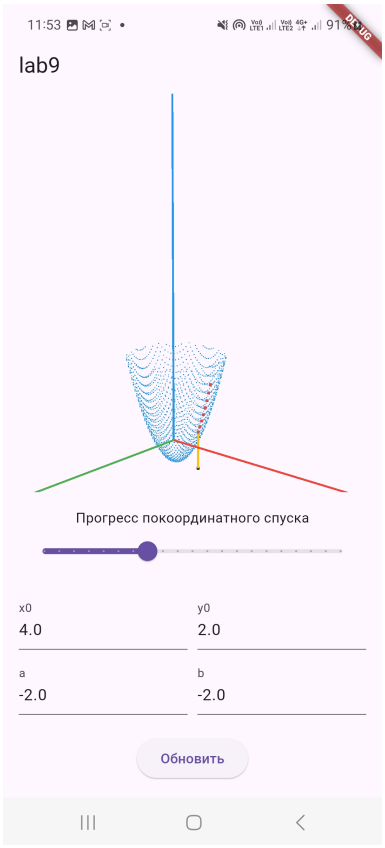

Рис. 2: результаты

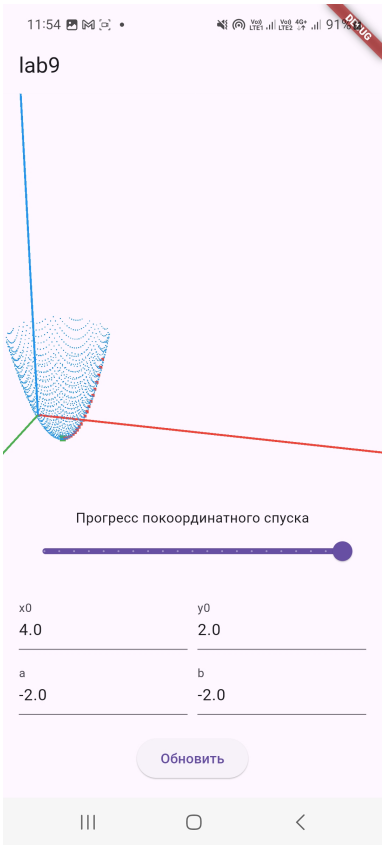

Рис. 3: результаты

Вывод

В ходе выполнения лабораторной работы был успешно реализован метод покоординатного спуска с визуализацией в Flutter. Приложение позволяет изменять параметры функции, что наглядно демонстрирует процесс нахождения минимума и особенности поведения метода на квадратичных функциях.