Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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»

Рубежный контроль № 1

по курсу «Разработка мобильных приложений»

«Работа с библиотекой ditredi»

Студент группы ИУ9-72Б Шемякин В.А.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

Реализовать мобильное приложение, где строится двумерный график функции и ищется минимум методом покоординатного спуска.

Вариант: функция Швефеля.

2 Результаты

Исходный код программы представлен в листинге 1.

```
import 'package:flutter/material.dart';
import 'package:ditredi/ditredi.dart';
import 'package:vector_math/vector_math_64.dart' hide Colors;
import 'dart:math';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Gradient Descent 3D     x  + y  ',
      theme: ThemeData(
        primarySwatch: Colors.deepPurple,
        scaffoldBackgroundColor: const Color(0xFFF5F5F7),
      ),
      home: const GradientDescentApp(),
    );
  }
}

class GradientDescentApp extends StatefulWidget {
  const GradientDescentApp({super.key});

  @override
  State<GradientDescentApp> createState() =>
    _GradientDescentAppState();
```

```
}

class _GradientDescentAppState extends State<GradientDescentApp> {
  final double a1 = -10.0, b1 = 10.0, a2 = -10.0, b2 = 10.0;
  double h = 1.0; //

  double step = 1.0; //

  List<Point3D> points = [];
  List<Point3D> path = [];
  List<Point3D> xyProjection = [];
  Vector3? startPoint;
  bool isDescending = false;
  final modelController = DiTreDiController();

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

  //                f(x, y) = x^2 + y^2
  double function(double x, double y) => x * x + y * y;

  //
  void generateSurface() {
    points.clear();
    for (double x = a1; x <= b1; x += h) {
      for (double y = a2; y <= b2; y += h) {
        points.add(Point3D(
          Vector3(x, y, function(x, y)),
          color: Color.lerp(Colors.blue, Colors.red, function(x, y)
    / 200)!,
          width: 2.0,
        ));
      }
    }
  }

  //

  void findMinimum(Vector3 start) {
    setState(() {
      path = [Point3D(Vector3(start.x, start.y, function(start.x,
   start.y)),
```

```
73            color: Colors.blue, width: 8.0)];
74        xyProjection = [
75          Point3D(Vector3(start.x, start.y, 0),
76              color: Colors.blue.withAlpha(128), width: 6.0)
77        ];
78        isDescending = true;
79      });
80
81      Future<void> stepMove() async {
82        while (true) {
83          final current = path.last.position;
84          final currentValue = function(current.x, current.y);
85
86          final neighbors = [
87            Vector3(current.x + step, current.y, function(current.x +
     step, current.y)),
88            Vector3(current.x - step, current.y, function(current.x -
     step, current.y)),
89            Vector3(current.x, current.y + step, function(current.x,
    current.y + step)),
90            Vector3(current.x, current.y - step, function(current.x,
    current.y - step)),
91          ].where((p) =>
92              p.x >= a1 && p.x <= b1 && p.y >= a2 && p.y <= b2).
    toList();
93
94          Vector3? nextPoint;
95          double minValue = currentValue;
96
97          for (var neighbor in neighbors) {
98            final value = function(neighbor.x, neighbor.y);
99            if (value < minValue) {
100             minValue = value;
101             nextPoint = neighbor;
102           }
103         }
104
105         if (nextPoint == null || (currentValue - minValue).abs() <
    1e-6) {
106           setState(() {
107             path.last = Point3D(Vector3(current.x, current.y,
    current.z),
108                 color: Colors.green, width: 8.0);
109             xyProjection.last = Point3D(Vector3(current.x, current.
    y, 0),
```

```
                color: Colors.green.withAlpha(128), width: 6.0);
            isDescending = false;
          });
          break;
        }

        setState(() {
          path.add(Point3D(Vector3(nextPoint!.x, nextPoint.y,
    function(nextPoint.x, nextPoint.y)),
              color: Colors.red, width: 8.0));
          xyProjection.add(Point3D(Vector3(nextPoint.x, nextPoint.y
    , 0),
              color: Colors.red.withAlpha(128), width: 6.0));
        });

        await Future.delayed(const Duration(milliseconds: 400));
      }
    }

    stepMove();
  }

  void updateSurfaceStep(double newValue) {
    setState(() {
      h = newValue;
      generateSurface();
    });
  }

  void updateAlgorithmStep(double newValue) {
    setState(() {
      step = newValue;
    });
  }

  List<Line3D> get axes {
    return [
      Line3D(Vector3(a1, 0, 0), Vector3(b1, 0, 0),
          color: Colors.orange, width: 3),
      Line3D(Vector3(0, a2, 0), Vector3(0, b2, 0),
          color: Colors.teal, width: 3),
      Line3D(Vector3(0, 0, 0), Vector3(0, 0, function(b1, b2)),
          color: Colors.purpleAccent, width: 3),
    ];
  }
```

```
153
154   @override
155   Widget build(BuildContext context) {
156     return Scaffold(
157       appBar: AppBar(
158         title: const Text('f(x, y) = x   + y   '),
159         centerTitle: true,
160         backgroundColor: Colors.deepPurple,
161         elevation: 4,
162       ),
163       body: Column(
164         children: [
165           Expanded(
166             child: Container(
167               decoration: const BoxDecoration(
168                 gradient: LinearGradient(
169                   colors: [Color(0xFFF8F9FA), Color(0xFFEAEAEA)],
170                   begin: Alignment.topLeft,
171                   end: Alignment.bottomRight,
172                 ),
173               ),
174               child: DiTreDiDraggable(
175                 controller: modelController,
176                 child: DiTreDi(
177                   figures: [
178                     ...axes,
179                     ...points,
180                     ...path,
181                     ...xyProjection,
182                   ],
183                   controller: modelController,
184                   config: const DiTreDiConfig(supportZIndex: false)
    ,
185                 ),
186               ),
187             ),
188           ),
189           Padding(
190             padding: const EdgeInsets.all(16.0),
191             child: Column(
192               children: [
193                 //
194                 Row(
195                   children: [
196                     const Text('                    (h):'),
```

```
                    Expanded(
                      child: Slider(
                        value: h,
                        min: 0.5,
                        max: 2.0,
                        divisions: 3,
                        label: h.toStringAsFixed(1),
                        onChanged: updateSurfaceStep,
                      ),
                    ),
                    Text(h.toStringAsFixed(1)),
                  ],
                ),
                const SizedBox(height: 16),

                //
                Row(
                  children: [
                    Expanded(
                      child: TextField(
                        decoration: InputDecoration(
                          labelText: 'X',
                          border: OutlineInputBorder(
                              borderRadius: BorderRadius.circular
(8)),
                          filled: true,
                          fillColor: Colors.white,
                        ),
                        keyboardType:
                            const TextInputType.numberWithOptions(
decimal: true),
                        onChanged: (value) {
                          if (value.isNotEmpty) {
                            final x = double.tryParse(value) ??
0.0;
                            setState(() {
                              startPoint = Vector3(
                                  x,
                                  startPoint?.y ?? 0.0,
                                  function(x, startPoint?.y ?? 0.0)
);
                            });
                          }
                        },
                      ),
```

```
238                     ),
239                     const SizedBox(width: 16),
240                     Expanded(
241                       child: TextField(
242                         decoration: InputDecoration(
243                           labelText: 'Y',
244                           border: OutlineInputBorder(
245                               borderRadius: BorderRadius.circular
    (8)),
246                           filled: true,
247                           fillColor: Colors.white,
248                         ),
249                         keyboardType:
250                             const TextInputType.numberWithOptions(
    decimal: true),
251                         onChanged: (value) {
252                           if (value.isNotEmpty) {
253                             final y = double.tryParse(value) ??
    0.0;
254                             setState(() {
255                               startPoint = Vector3(
256                                   startPoint?.x ?? 0.0,
257                                   y,
258                                   function(startPoint?.x ?? 0.0, y)
    );
259                             });
260                           }
261                         },
262                       ),
263                     ),
264                     const SizedBox(width: 16),
265                     ElevatedButton(
266                       onPressed: startPoint != null && !
    isDescending
267                           ? () => findMinimum(startPoint!)
268                           : null,
269                       child: const Text('          '),
270                     ),
271                   ],
272                 ),
273               ],
274             ),
275           ),
276         ],
277       ),
```

```
278         );
279     }
280 }
```

Результат запуска представлен на рисунке 1.

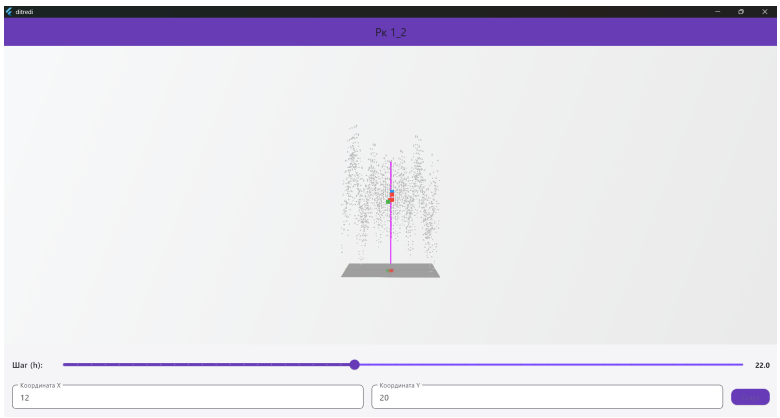

Рис. 1 — Результат

.