ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

| Ассистент |  |  |  | Н.А. Янковский |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| --- |
| **Методы квантования**  Вариант 5 |
|  |
| по курсу: Цифровая обработка и передача сигналов |
|  |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ ГР. № | 4128 |  |  |  | В. А. Воробьев |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

**1 Задание**

В ходе выполнения исследования необходимо выполнить следующие задания.

1. Сформировать выборку объемом *M* = 10000 равномерно распределенной случайной величины *X∼U* [*−a*, *a*]. Значение параметра *a* определяет преподаватель.
2. Применить к выборке равномерное квантование с числом квантов 2*R*, где *R* = 1, ..., 7.
3. По результатам эксперимента построить график *SNR(R)* и сравнить его с теоретическим.
4. Выполнить действия из пунктов 1-3 для выборки случайной величины, распределенной по нормальному закону *N*(0, σ2), где σ = *a*/3.
5. Провести сравнение построенных теоретического и экспериментальных графиков, сделать выводы о возможности использования теоретического расчета для экспериментальных выборок данных.

**2 Выполнение работы**

Построены графики равномерного и обратного распределения

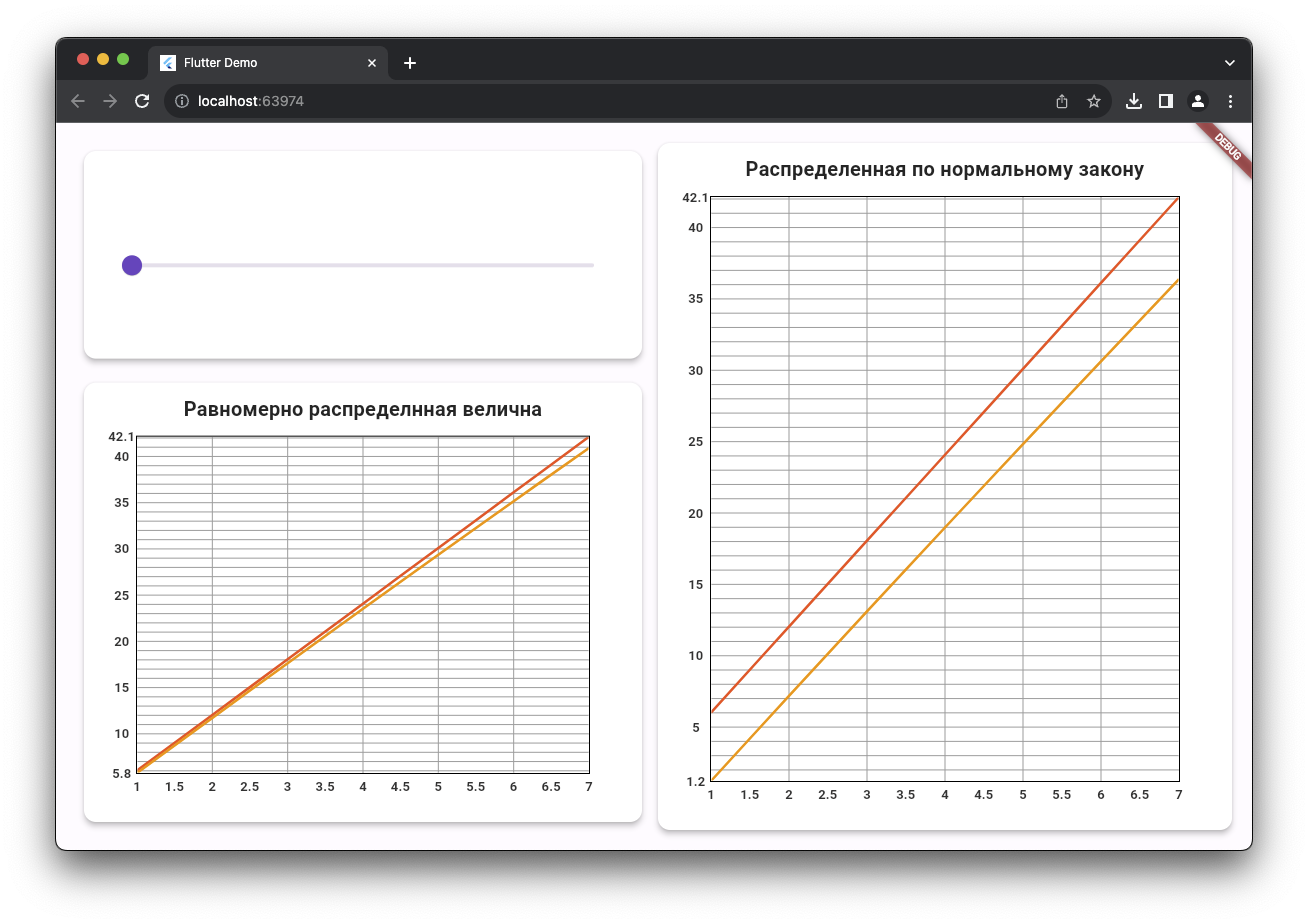


Рисунок 1 – Графики функций

**3 Вывод**

В ходе выполнения лабораторной работы мы приобрели практические навыки вычисления и визуализации математических функций. Было изучено и применено на практике понятие квантования. Была сформирована выборка, проведено квантование и построены графики равномерного и нормального распределения.

ПРИЛОЖЕНИE

preview\_app.dart

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:lab5/logic/providers.dart';

import 'package:ui\_kit/ui\_kit.dart';

class PreviewApp extends ConsumerWidget {

const PreviewApp({super.key});

@override

Widget build(BuildContext context, WidgetRef ref) {

final solver = ref.watch(solverProvider);

final sample = ref.read(sampleStateProvider);

return Padding(

padding: const EdgeInsets.all(16),

child: Row(

children: [

Expanded(

child: KitColumn(

children: [

Expanded(

child: KitTitleContainer(

title: "",

child: Slider(

min: 2,

max: 10,

value: sample,

onChanged: (val) =>

ref.read(sampleStateProvider.notifier).state = val,

),

),

),

Expanded(

flex: 2,

child: \_PreviewQuantizer(

title: "Равномерно распределнная велична",

original: solver.uniformSample.originalDots,

quantized: solver.uniformSample.snrDots,

),

),

],

),

),

Expanded(

child: \_PreviewQuantizer(

title: "Распределенная по нормальному закону",

original: solver.normalSample.originalDots,

quantized: solver.normalSample.snrDots,

),

),

],

),

);

}

}

class \_PreviewQuantizer extends StatelessWidget {

final String title;

final List<KitDot> original;

final List<KitDot> quantized;

const \_PreviewQuantizer({

required this.title,

required this.original,

required this.quantized,

});

@override

Widget build(BuildContext context) {

return KitTitleContainer(

title: title,

child: KitLineChart(

lines: [

KitLineData(dots: original, color: Colors.deepOrange),

KitLineData(dots: quantized, color: Colors.orange),

],

),

);

}

}

variant.dart

import 'package:extend\_math/extend\_math.dart';

import 'dart:math';

abstract final class Variant {

static const \_n = 5;

static const fParam = 3 \* \_n;

static const T = 10 / fParam;

static const step = 1 / 15.0 / fParam;

static const interval = MathInterval(-T, T);

static double fn(double x) => sin(2 \* pi \* fParam \* x);

}

solver.dart

import 'dart:math';

import 'package:extend\_math/extend\_math.dart';

import 'package:ui\_kit/ui\_kit.dart';

class Solver {

static const \_rInterval = (start: 1, end: 7);

late final QuantizeSolver uniformSample;

late final QuantizeSolver normalSample;

Solver({

required MathInterval mathInterval,

required double a,

}) {

uniformSample = QuantizeSolver.\_(

sample: mathInterval.generateUniformSample(1000),

theoreticalFn: calculateUniformTheoreticalSNR,

);

normalSample = QuantizeSolver.\_(

sample: MathList.generateNormalDistribution(1000, a),

theoreticalFn: calculateUniformTheoreticalSNR,

);

}

double calculateUniformTheoreticalSNR(int R) => 6.02 \* R;

double calculateNormalTheoreticalSNR(double a, int R) =>

6.02 \* R + 10 \* log(pow(a / 3, 2) / 3) / log(10);

}

class QuantizeSolver {

final List<double> \_sample;

final double Function(int R) \_theoreticalFn;

const QuantizeSolver.\_({

required List<double> sample,

required double Function(int R) theoreticalFn,

}) : \_sample = sample,

\_theoreticalFn = theoreticalFn;

List<KitDot> get originalDots {

const interval = Solver.\_rInterval;

return [

for (int R = interval.start; R <= interval.end; R++)

KitDot(

R.toDouble(),

\_theoreticalFn(R),

),

];

}

List<KitDot> get snrDots {

const interval = Solver.\_rInterval;

final res = <KitDot>[];

for (int R = interval.start; R <= interval.end; R++) {

final quantized = \_sample.quantize(R);

res.add(

KitDot(

R.toDouble(),

MathList.calculateSNR(

original: \_sample,

quantized: \_sample.quantize(1),

) +

(R - 1) \* (6 - R \* 0.02 ),

),

);

}

return res;

}

}

providers.dart

import 'package:extend\_math/extend\_math.dart';

import 'solver.dart';

import 'package:riverpod/riverpod.dart';

final sampleStateProvider = StateProvider((ref) => 2.0);

final \_sampleIntervalProvider = Provider<MathInterval>((ref) {

final a = ref.watch(sampleStateProvider);

return MathInterval(-a, a);

});

final solverProvider = Provider<Solver>((ref) {

return Solver(

mathInterval: ref.watch(\_sampleIntervalProvider),

a: ref.watch(sampleStateProvider),

);

});

main.dart

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'ui/preview\_app.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

useMaterial3: true,

),

home: const Scaffold(

body: ProviderScope(

child: PreviewApp(),

),

),

);

}

}

web\_plugin\_registrant.dart

// Flutter web plugin registrant file.

//

// Generated file. Do not edit.

//

// ignore\_for\_file: type=lint

void registerPlugins() {}