**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 7**

Тема: Построение плоских полиномиальных кривых.

Студент: Ильин Илья Олегович

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

**Вариант 11:**  B-сплайн. n = 5, k = 3. Узловой вектор равномерный.

1. Описание программы

Программа написана на Swift и CoreGraphics. Функция countCoefficients считает коэффициенты для B-сплайна по формуле Кокса-де-Бура. Функция bSpline считает точки итоговой кривой на основе массива контрольных точек, степени кривой и шага апроксимации.

1. Набор тестов

Тест 1: все точки лежат на одной прямой

Тест 2: конечные точки лежат в разных полуплоскостях

Тест 3: средний перепад ординат у точек

Тест 4: максимальный перепад ординат у точек

1. Результаты выполнения тестов

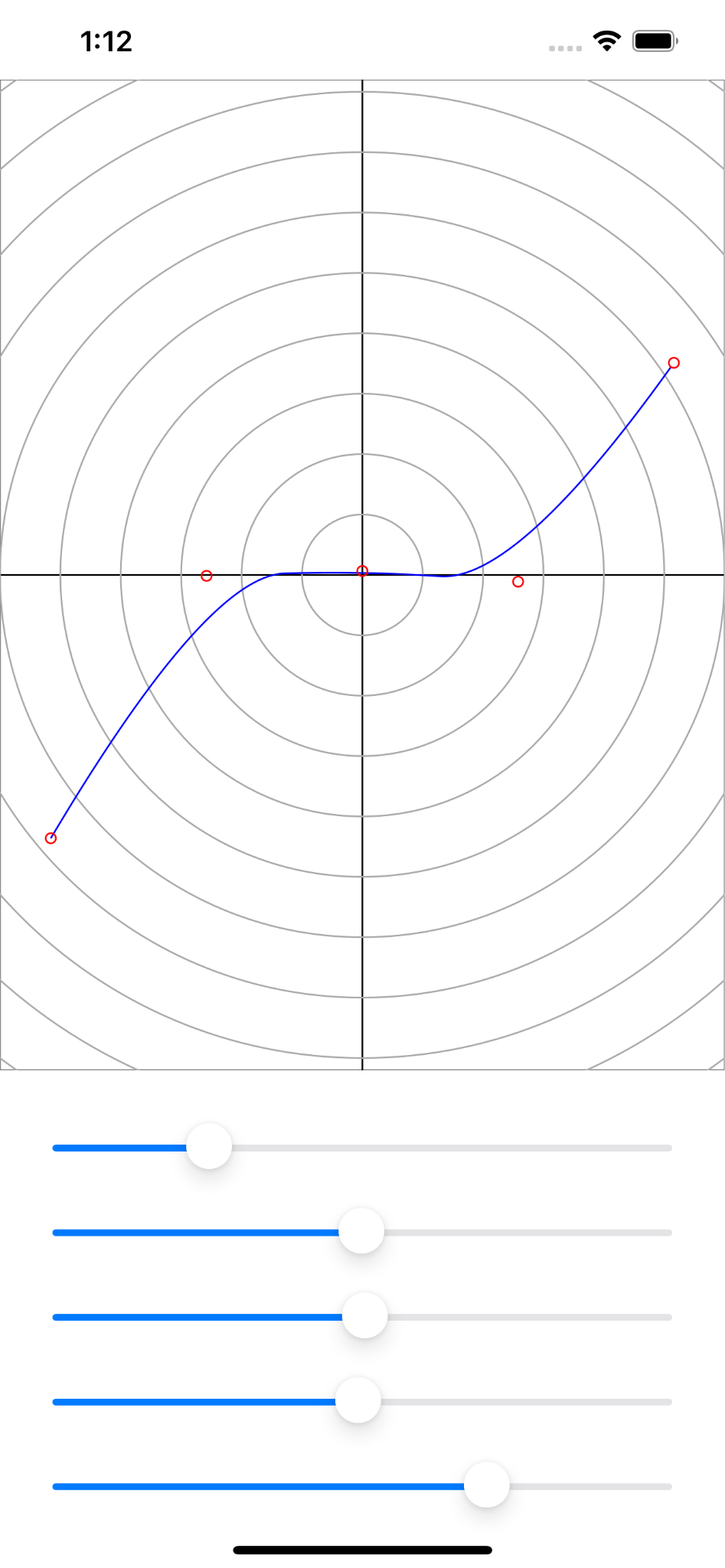
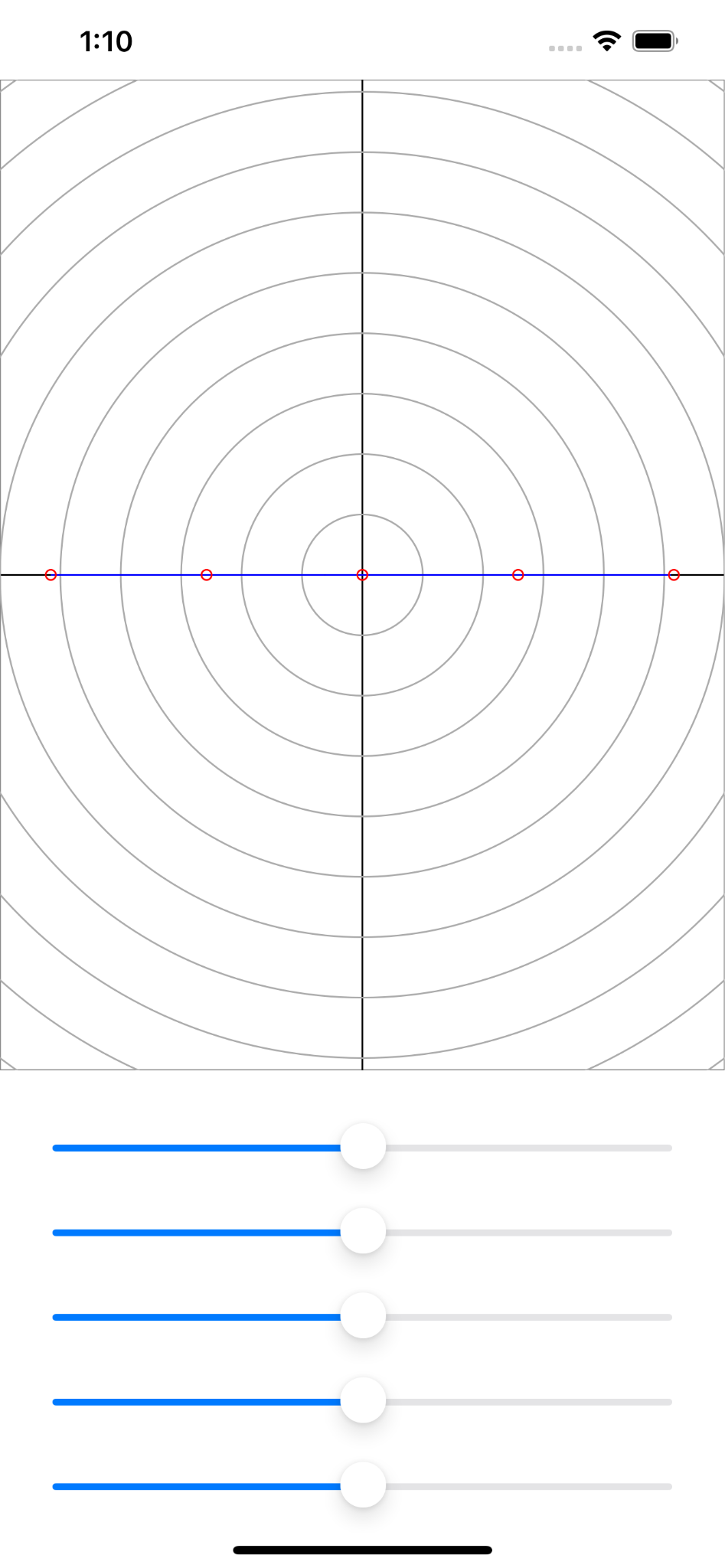


Рис. 1. Тест 1. Рис. 2. Тест 2.

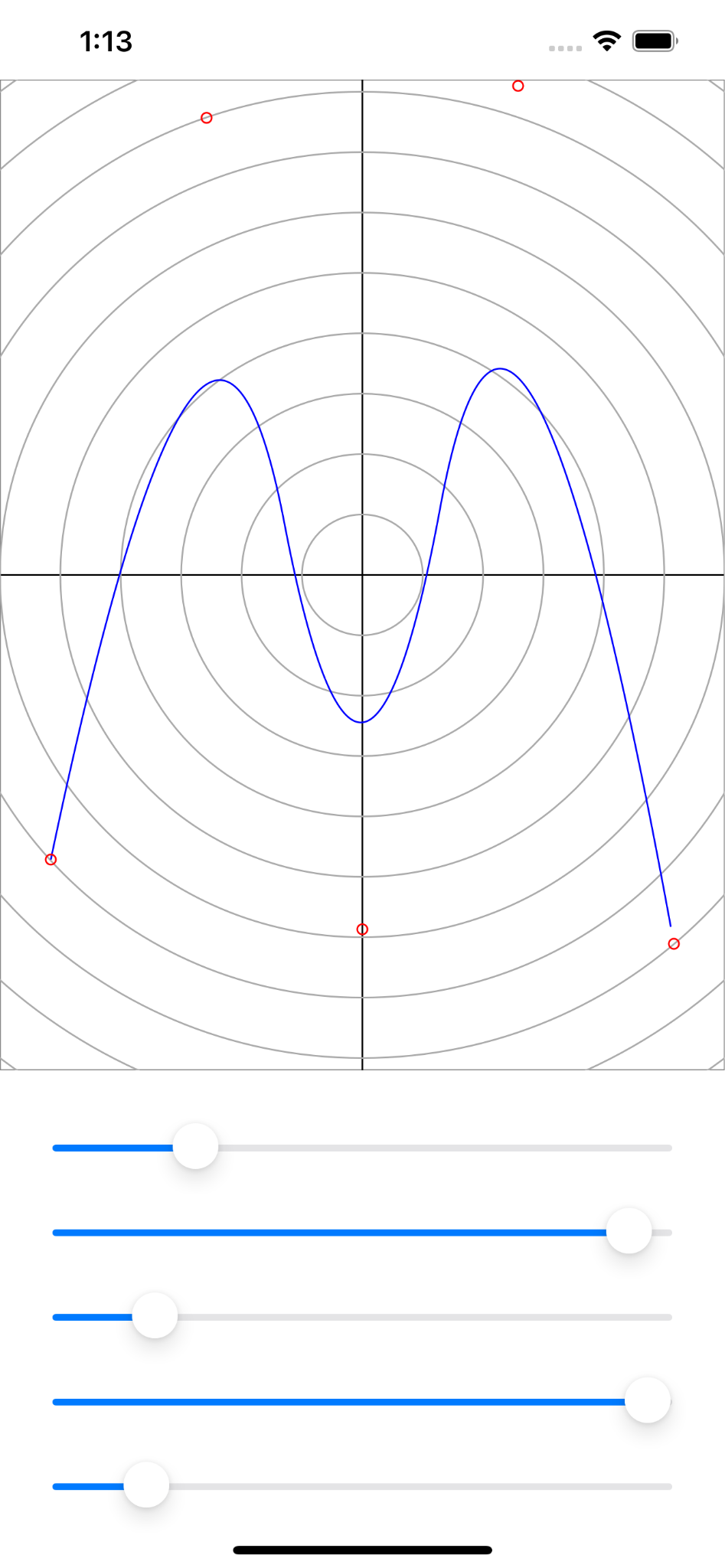
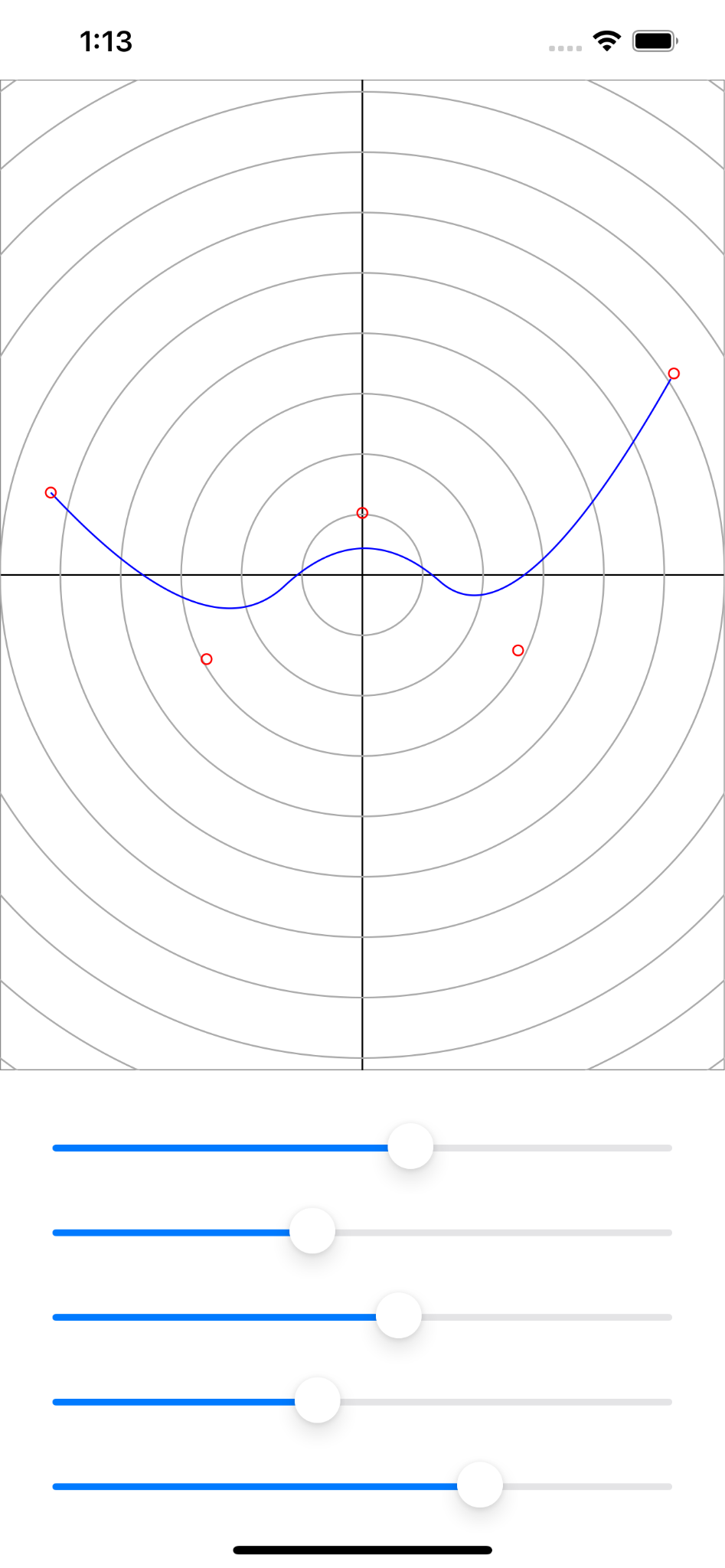


Рис. 3. Тест 3. Рис. 4. Тест 4.

1. Листинг программы

ViewController.swift

//

// ViewController.swift

// CG7

//

// Created by Илья Ильин on 23.12.2021.

//

/\*

Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

Вариант 11: B-сплайн. n = 5, k = 3. Узловой вектор равномерный.

\*/

**import** UIKit

**import** simd

**class** ViewController: UIViewController {

**var** graphView: GraphView!

**var** stackView: UIStackView!

**var** sliders = [UISlider]()

**let** degree = 3

**var** points = [simd\_double3]()

**let** pointsCount = 5

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

setupViews()

countPoints()

drawBSpline()

}

**override** **func** viewDidLayoutSubviews() {

setupSliders()

}

// **MARK: Setup views**

**func** setupViews() {

graphView = GraphView()

graphView.backgroundColor = .white

graphView.translatesAutoresizingMaskIntoConstraints = **false**

view.addSubview(graphView)

stackView = UIStackView()

stackView.translatesAutoresizingMaskIntoConstraints = **false**

stackView.axis = .vertical

stackView.alignment = .fill

stackView.spacing = 20

stackView.distribution = .fillEqually

**for** i **in** 0..<pointsCount {

**let** slider = UISlider()

slider.tag = i

slider.addTarget(**self**, action: **#selector**(sliderValueChanged(\_:)), **for**: .valueChanged)

stackView.addArrangedSubview(slider)

sliders.append(slider)

}

view.addSubview(stackView)

addConstraints()

}

**func** addConstraints() {

graphView.setContentHuggingPriority(.defaultHigh, **for**: .horizontal)

NSLayoutConstraint.activate([

graphView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor),

graphView.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor),

graphView.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor),

graphView.bottomAnchor.constraint(equalTo: stackView.topAnchor, constant: -30),

stackView.centerXAnchor.constraint(equalTo: view.centerXAnchor),

stackView.widthAnchor.constraint(equalTo: view.safeAreaLayoutGuide.widthAnchor, constant: -60),

stackView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor),

])

}

**func** setupSliders() {

**let** height = graphView.frame.height

**for** slider **in** sliders {

slider.minimumValue = -1 \* Float(height/2)

slider.maximumValue = Float(height/2)

}

}

**@objc** **func** sliderValueChanged(\_ sender: UISlider) {

**let** pointIdx = sender.tag

points[pointIdx].y = Double(sender.value)

drawBSpline()

}

// **MARK: B-Spline**

**func** drawBSpline() {

**let** spline = bSpline(degree: degree, points: points, step: 0.01)

**let** graphPoints = spline.map{ CGPoint(x: $0.x, y: $0.y) }

graphView.linePoints = graphPoints

**let** dotPoints = points.map{ CGPoint(x: $0.x, y: $0.y) }

graphView.dotPoints = dotPoints

}

**func** countPoints() {

**let** padding: CGFloat = 30

**let** viewWidth = **self**.view.frame.**self**.width

**let** width: CGFloat = ( viewWidth - padding\*2 ) / CGFloat(pointsCount - 1)

**let** start: CGFloat = -1 \* viewWidth / 2 + padding

print(">>> \(start), \(width)")

**for** i **in** 0..<pointsCount {

points.append(simd\_double3(x: start + width \* CGFloat(i),

y: 0,

z: 0))

}

}

**func** bSpline(degree: Int, points: [simd\_double3], step: Double) -> [simd\_double3] {

**guard** 1 <= degree && degree <= points.count - 1 **else** {

NSLog("Invalid dimension")

**return** []

}

**var** result = [simd\_double3]()

**let** knots = countKnots(degree: degree, pointsNum: points.count)

**let** range = [ Double(knots[degree-1]), Double(knots[points.count]) ]

**for** u **in** stride(from: range[0], to: range[1], by: step) {

**let** U = simd\_double4(u\*u\*u, u\*u, u, 1)

**var** num = Int(floor(u - range[0]))

**var** point = simd\_double3(0, 0, 0)

**if** u == range[1] {

num -= 1

}

**for** i **in** num..<num + degree {

**let** coeff = countCoefficients(i: i, degree: degree, u: u, knots: knots)

point += simd\_dot(coeff, U) \* points[i]

}

result.append(point)

}

**return** result

}

**func** countKnots(degree: Int, pointsNum: Int) -> [Int] {

**var** knots = [Int]()

**for** \_ **in** 0..<degree-1 {

knots.append(0)

}

**for** i **in** 0..<degree + pointsNum - 2\*degree + 2 {

knots.append(i)

}

**for** \_ **in** 0..<degree-1 {

knots.append(knots.last ?? 0)

}

**return** knots

}

**func** countCoefficients(i: Int, degree: Int, u: Double, knots: [Int]) -> simd\_double4 {

**let** result: simd\_double4

**if** degree == 1 {

**if** (u >= Double(knots[i]) && u < Double(knots[i+1])) || abs( Double(knots.last ?? 0) - u) < 1e-8 {

result = simd\_make\_double4(0, 0, 0, 1)

} **else** {

result = simd\_make\_double4(0, 0, 0, 0)

}

} **else** {

**var** length1: Double = Double( knots[i + degree - 1] - knots[i] )

**var** length2: Double = Double( knots[i + degree] - knots[i + 1] )

**if** length1 == 0.0 {

length1 = 1.0

}

**if** length2 == 0.0 {

length2 = 1.0

}

**var** tmp1 = simd\_double4()

**var** tmp2 = simd\_double4()

tmp1[0] = -1 \* countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[0] \* Double( knots[i] ) / length1 + countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[1] / length1

tmp1[1] = -1 \* countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[1] \* Double( knots[i] ) / length1 + countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[2] / length1

tmp1[2] = -1 \* countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[2] \* Double( knots[i] ) / length1 + countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[3] / length1

tmp1[3] = -1 \* countCoefficients(i: i, degree: degree - 1, u: u, knots: knots)[3] \* Double( knots[i] ) / length1

tmp2[0] = countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[0] \* Double( knots[i + degree] ) / length2 - countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[1] / length2

tmp2[1] = countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[1] \* Double( knots[i + degree] ) / length2 - countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[2] / length2

tmp2[2] = countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[2] \* Double( knots[i + degree] ) / length2 - countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[3] / length2

tmp2[3] = countCoefficients(i: i + 1, degree: degree - 1, u: u, knots: knots)[3] \* Double( knots[i + degree] ) / length2

result = tmp1 + tmp2

}

**return** result

}

}

1. Выводы

В ходе выполнения данной ЛР я познакомился со сплайн функциями, а конкретно с B-сплайном.

B-кривые разрабатывались как замена существующим кривым Безье, главным недостатком которых являлась невозможность их локальной модификации и жесткая привязка степени полинома к количеству точек. B-кривые избавлены от всех этих недостатков: степень может быть задана пользователем, также присутствует локальная модификация кривой (чем ниже степень, тем более локальными получаются модификации).

ЛИТЕРАТУРА

1. Документация Swift [Электронный ресурс] URL: https://swift.org/documentation/ (Дата обращения: 20.12.2021).