**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 1**

Тема: Построение изображений 2D кривых

Студент: Ильин Илья Олегович

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

Написать и отладить программу, строящую изображение заданной замечательной кривой.

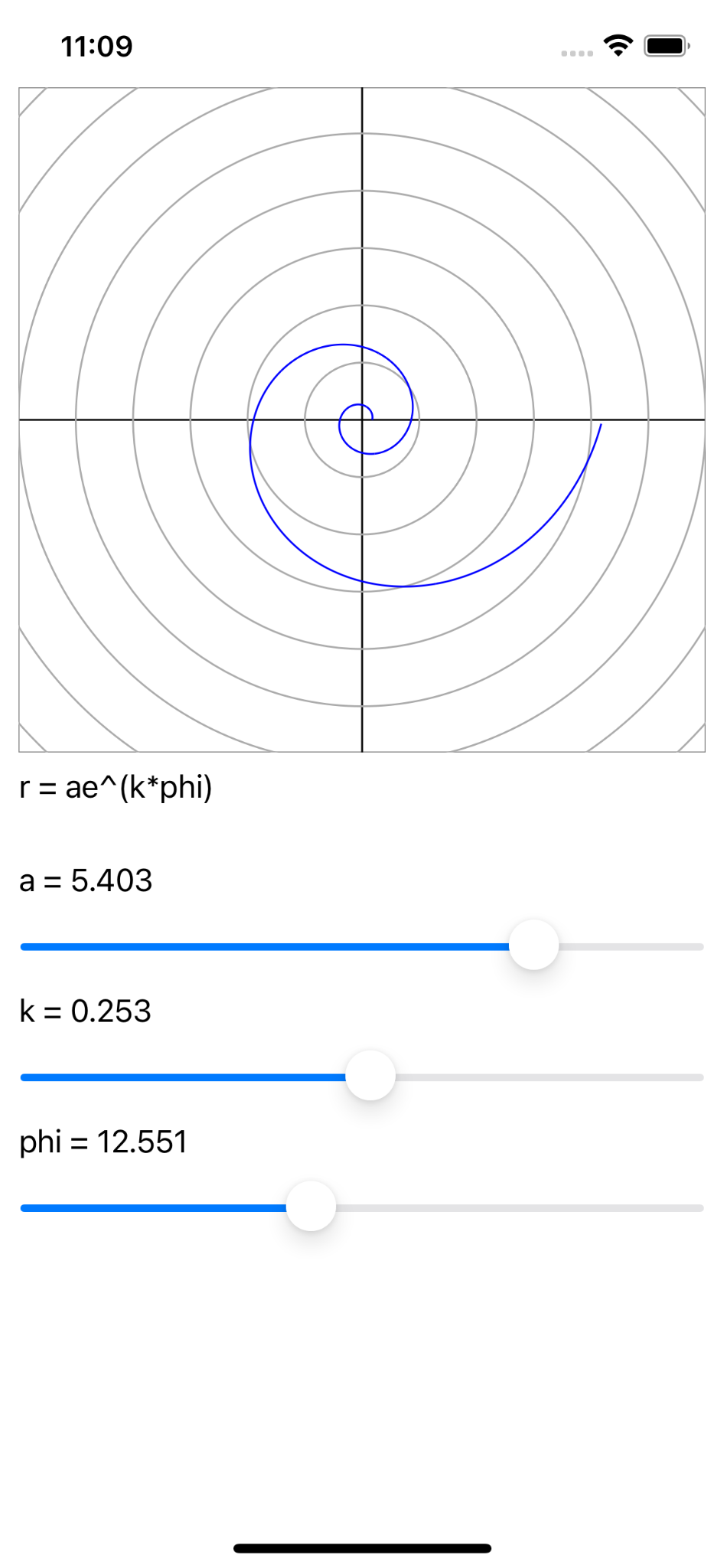
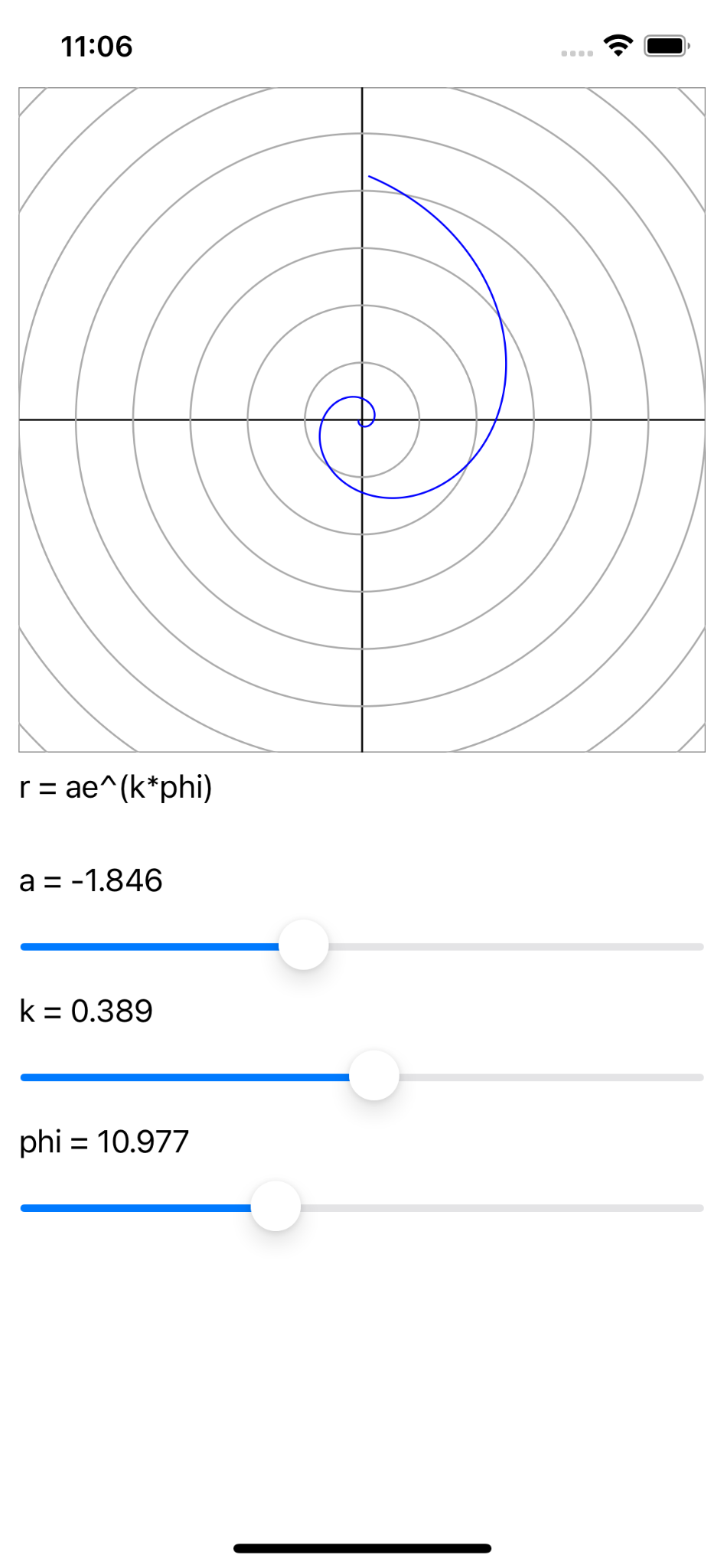
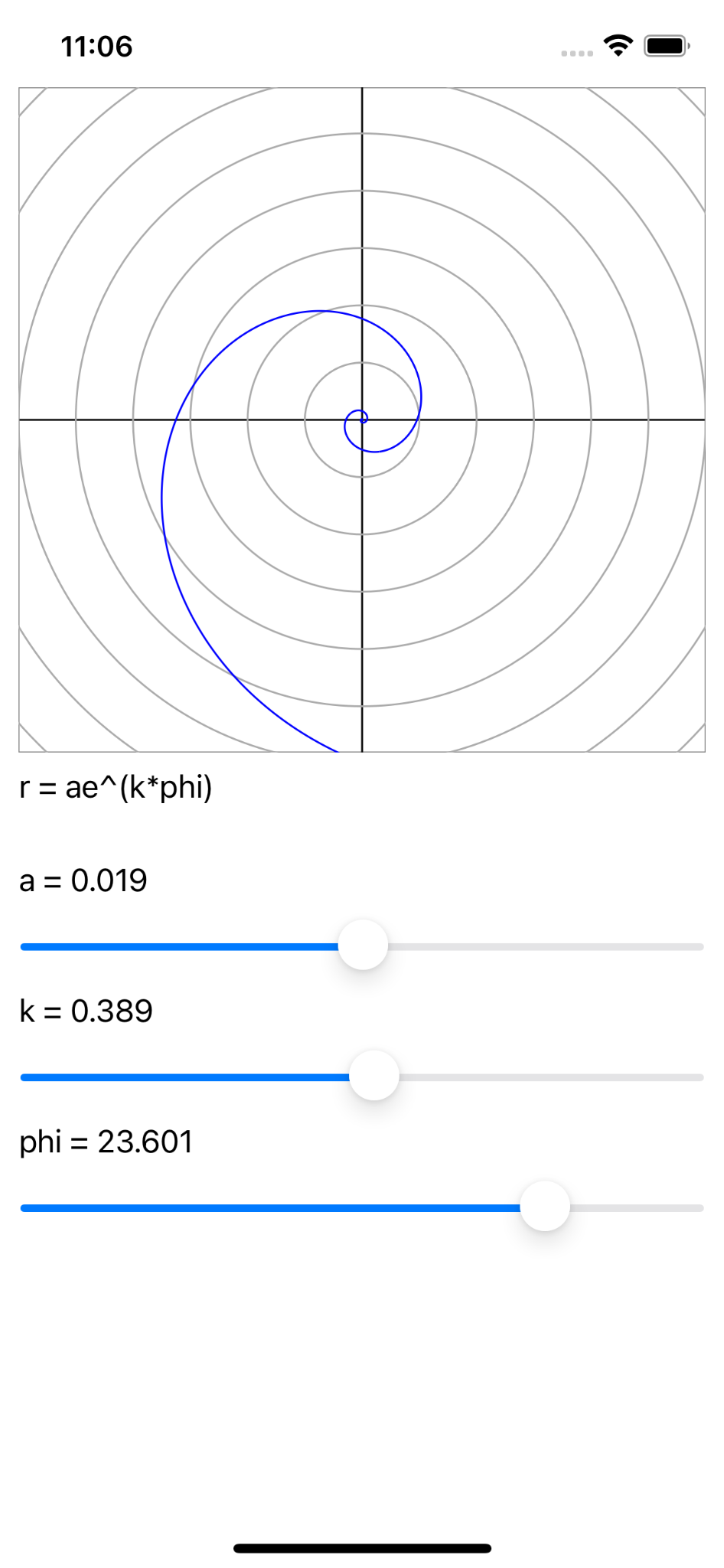
Вариант 10:

1. Описание программы

Программа состоит из UIGraphView, который расположен в верхней половине экрана и отображает график заданной кривой. Ниже, на оставшейся половине, расположены 3 слайдера параметров.

Способ отрисовки: с определенным шагом берем угол, вычисляем значение радиус вектора для этого угла, затем преобразуем полярные координаты и прямоугольным. Полученные точки соединяем между собой линиями, используя UIBezierPath

1. Набор тестов



1. Листинг программы

UIGraphView.swift:

//

// UIGraphView.swift

// CG1

//

// Created by Илья Ильин on 21.09.2021.

//

/\*

- Мб здеь хранить только массив точек, а высчитывать их в контроллере???

\*/

**import** UIKit

**import** Foundation

**@IBDesignable**

**class** UIGraphView: UIView {

**var** points = [CGPoint]() {

**didSet** {

setNeedsDisplay()

}

}

// Преобразует точку из системы координат где центр находится посередине View в обычную систему координат

**func** getPoint(x: CGFloat, y: CGFloat) -> CGPoint {

**let** viewHeight = **self**.bounds.size.height

**let** viewWidth = **self**.bounds.size.width

**return** CGPoint(x: x + (viewWidth / 2),

y: (-1) \* y + (viewHeight / 2))

}

**func** convertPoint(\_ point: CGPoint) -> CGPoint {

**let** viewHeight = **self**.bounds.size.height

**let** viewWidth = **self**.bounds.size.width

**return** CGPoint(x: point.x + (viewWidth / 2),

y: (-1) \* point.y + (viewHeight / 2))

}

**override** **func** draw(\_ rect: CGRect) {

// Рамка по периметру

**let** viewHeight = **self**.bounds.size.height

**let** viewWidth = **self**.bounds.size.width

**let** borderLine = UIBezierPath()

borderLine.move(to: CGPoint(x: 0, y: 0))

borderLine.addLine(to: CGPoint(x: viewWidth, y: 0))

borderLine.addLine(to: CGPoint(x: viewWidth, y: viewHeight))

borderLine.addLine(to: CGPoint(x: 0, y: viewHeight))

borderLine.addLine(to: CGPoint(x: 0, y: 0))

UIColor.gray.setStroke()

borderLine.stroke()

**let** oX = UIBezierPath()

oX.move(to: CGPoint(x: 0, y: viewHeight / 2))

oX.addLine(to: CGPoint(x: viewWidth, y: viewHeight / 2))

UIColor.black.setStroke()

oX.stroke()

**let** oY = UIBezierPath()

oY.move(to: CGPoint(x: viewWidth / 2, y: viewHeight))

oY.addLine(to: CGPoint(x: viewWidth / 2, y: 0))

oY.stroke()

// линии радиус векторов

**let** centerPoint = getPoint(x: 0, y: 0)

**let** maxCircleRadius = sqrt(viewHeight \* viewHeight + viewWidth \* viewWidth) / 2

**let** circleStep = viewWidth / 12

**for** radius **in** stride(from: circleStep, through: maxCircleRadius, by: circleStep) {

**let** circlePath : UIBezierPath = UIBezierPath(

arcCenter: centerPoint,

radius: CGFloat(radius),

startAngle: 0,

endAngle: CGFloat(Double.pi \* 2),

clockwise: **true**

)

UIColor.lightGray.setStroke()

circlePath.stroke()

}

**let** graphLine = UIBezierPath()

**var** isFirst = **true**

**for** point **in** points {

**if** isFirst {

isFirst = **false**

graphLine.move(to: convertPoint(point))

}

graphLine.addLine(to: convertPoint(point))

}

UIColor.blue.setStroke()

graphLine.stroke()

}

}

ViewController.swift

//

// ViewController.swift

// CG1

//

// Created by Илья Ильин on 21.09.2021.

//

// М8О-206Б-19

// Написать и отладить программу, строящую изображение заданной замечательной кривой.

// r = a\*e^(k\*phi), phi <= B

**import** UIKit

**class** ViewController: UIViewController {

**@IBOutlet** **weak** **var** graphView: UIGraphView!

**@IBOutlet** **weak** **var** functionLabel: UILabel!

**@IBOutlet** **weak** **var** aLabel: UILabel!

**@IBOutlet** **weak** **var** kLabel: UILabel!

**@IBOutlet** **weak** **var** phiLabel: UILabel!

**@IBOutlet** **weak** **var** aSlider: UISlider!

**@IBOutlet** **weak** **var** kSlider: UISlider!

**@IBOutlet** **weak** **var** phiSlider: UISlider!

// параметры функции

**var** points = [CGPoint]()

**var** phiStep: Double = 0.01

**var** a: Double = 1 {

**didSet** {

**if** a != oldValue {

redrawGraph()

}

}

}

**var** k: Double = 1 {

**didSet** {

**if** k != oldValue {

redrawGraph()

}

}

}

**var** phiUpperBound: Double = 16 {

**didSet** {

**if** phiUpperBound != oldValue {

redrawGraph()

}

}

}

**func** countPoints() {

points.removeAll()

**for** phi **in** stride(from: 0, through: phiUpperBound, by: phiStep) {

**let** x = a \* exp(k \* phi) \* cos(phi)

**let** y = a \* exp(k \* phi) \* sin(phi)

points.append(CGPoint(x: x, y: y))

}

}

**func** redrawGraph() {

countPoints()

graphView.points = points

}

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

**super**.viewDidLoad()

functionLabel.text = "r = ae^(k\*phi)"

aSlider.value = Float(a)

aSlider.minimumValue = -10

aSlider.maximumValue = 10

aLabel.text = "a = \(aSlider.value)"

kSlider.value = Float(k)

kSlider.minimumValue = -10

kSlider.maximumValue = 10

kLabel.text = "k = \(k)"

phiSlider.value = Float(phiUpperBound)

phiSlider.minimumValue = 0

phiSlider.maximumValue = 30

phiLabel.text = "phi = \(phiUpperBound)"

redrawGraph()

}

**@IBAction** **func** aSliderValueChanged(\_ sender: UISlider) {

a = Double(sender.value)

aLabel.text = "a = \(Double(round(1000\*a)/1000))"

}

**@IBAction** **func** kSliderValueChanged(\_ sender: UISlider) {

k = Double(sender.value)

kLabel.text = "k = \(Double(round(1000\*k)/1000))"

}

**@IBAction** **func** phiSliderValueChanged(\_ sender: UISlider) {

phiUpperBound = Double(sender.value)

phiLabel.text = "phi = \(Double(round(1000\*phiUpperBound)/1000))"

}

}

1. Вывод

В ходе проделанной ЛР научился использовать графические примитивы для построения 2D кривых, также использовал обновление UI посредством отслеживания изменения параметров кривой: при изменении значения параметра на слайдере идет пересчет точек и график строится заново

ЛИТЕРАТУРА

1. Документация Swift [Электронный ресурс] URL: https://swift.org/documentation/ (Дата обращения: 20.09.2021).