Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия задач вычислительной математики»

Программирование численных методов обработки табличной информации

Пояснительная записка

ОГУ 09.03.04.3022.187 ПЗ

Руководитель
старший преподаватель
_____ Е.А. Шнякина
«27» декабря 2022 г.

Студент группы 20ПИнж(б)РПиС
Захаров А.В. Захаров
«27» декабря 2022 г.

Оренбург 2022

Утверждаю
Заведующий кафедрой
программного обеспечения вычислительной
техники и автоматизированных систем

_____ Д.В.Горбачев
подпись инициалы, фамилия

« 4 » октября 2022 г.

ЗАДАНИЕ

на выполнение курсовой работы

студенту Захарову Андрею Владимировичу
фамилия, имя, отчество

по направлению подготовки 09.03.04 Программная инженерия
код, наименование

по дисциплине Программная инженерия задач вычислительной математики
наименование дисциплины

1 **Тема работы**: Программирование численных методов обработки табличной информации.

2 **Срок сдачи студентом работы** «27» декабря 2022 г

3 **Цель и задачи работы**

Целью курсовой работы является систематизация, закрепление, расширение теоретических и практических знаний по основам вычислительной математики в области интерполирования и восстановления функции,

Курсовая работа включает в себя решение следующих задач:

Для заданной последовательности точек, используя нелокальный кубический сплайн (I тип граничных условий), вычислить значения в заданных точках x^*.

При реализации алгоритма разработать 2-3 набора тестовых данных и оценить на них правильность реализации алгоритма.

4 Исходные данные к работе: таблично заданная функция; точки x^*, в которых необходимо вычислить значения функции - вариант № 4.

5 Перечень вопросов, подлежащих разработке:

1) систематизация теоретических знаний в области интерполирования функций;
2) программная реализация численного метода интерполирования функции; 3) разработка тестовых данных; 4) анализ и интерпретация результатов решения поставленной задачи

6 Перечень иллюстративного материала презентация доклада.

Дата выдачи и получения задания

Руководитель «4» октября 2022 г. _____ Е.А. Шнякина
подпись инициалы, фамилия

Студент «4» октября 2022 г. Захаров Л.В. Захаров
подпись инициалы, фамилия

Аннотация

Курсовая работа представляет собой решение задачи интерполирования функций на основе использования численного метода: нелокальный кубический сплайн (I тип граничных условий).

В работе излагаются теоретические аспекты интерполирования функции методом нелокального кубического сплайна (I тип граничных условий). Представлены выводы расчётных формул и погрешности метода. Выполнено сопоставление метода с другими методами, решающими задачу интерполирования функции. Приведён алгоритм разработанного программного средства, реализующего заданный метод. Разработаны и описаны наборы тестовых данных.

Выполнено решение варианта № 4. Результаты вычисленных значений в указанных точках представлены таблично и графически. (Наилучшее приближение таблично заданной функции представлено в аналитическом и графическом видах).

Работа содержит 32 листа текста, 10 рисунков, 29 таблиц, 1 приложение.

					ОГУ 09.03.04.3022.187 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Захаров А.В.	Захаров	27.12	Программирование численных методов обработки табличной информации	Лит.	Лист	Листов
Провер.		Шнякина Е.А.		27.12		К Р	3	32
Н.контр.						ФМИТ, 20ПИнж(б)РПиС		
Зав. Каф.		Горбачев Д.В.						

Содержание

Введение

Целью курсовой работы является систематизация, закрепление, расширение теоретических и практических знаний по основам вычислительной математики в области интерполирования и восстановления функции.

Курсовая работа включает в себя решение следующих задач:

– Для заданной последовательности точек, используя нелокальный кубический сплайн (I тип граничных условий), вычислить значения в заданных точках x^*.

– При реализации алгоритма разработать 2-3 набора тестовых данных и оценить на них правильность реализации алгоритма.

Поставленная задача актуальна, и может использоваться, например в морской навигации. При помощи визуализации сплайновой изоповерхности [1], можно мгновенно исключить ошибки оценки глубины дна и предотвратить повреждение судна.

Формулировка задания курсовой работы согласно выданному варианту

На отрезке $x \in [a; b]$ дана функция $y = f(x)$ в табличной форме* (см. таблицу 1):

Таблица 1 – пример табличной формы функции

x_i	x_0	x_1	…	x_9
y_i	y_0	y_1	…	y_9

*где $x_0 = a$, $x_9 = b$.

Необходимо вычислить значения функции в точках x_k^*, $k = 1, \dots, 5$, используя интерполяцию нелокальными кубическими сплайнами (1-ый тип граничных условий).

В таблице 2 представлен вариант функции в табличной форме.

Таблица 2 – вариант функции в табличной форме

i	0	1	2	3	4
x_i	0,02	0,04	0,06	0,08	0,10
y_i	1,414216	1,414236	1,414290	1,414395	1,414567

Продолжение таблицы 2

i	5	6	7	8	9
x_i	0,12	0,14	0,16	0,18	0,20
y_i	1,414824	1,415183	1,415661	1,416274	1,417039

Граничные условия для построения нелокальных кубических сплайнов приведены в формуле (1).

$$S'(0{,}02) = 0{,}00085,\ S'(0{,}20) = 0{,}08468 \qquad (1)$$

Значения x_k^*, в которых необходимо вычислить значения соответствующих таблично заданных функций, даны в таблице 3.

Таблица 3 - значения x_k^*

x_1^*	x_2^*	x_3^*	x_4^*	x_5^*
0,03	0,05	0,09	0,15	0,19

Полученные значения y_k^* изобразить графически, построив в одной системе координат и на одном рисунке заданные точки (x_i, y_i), $i = 0, \dots, 9$ и полученные (x_k^*, y_k^*), $k = 1, \dots, 5$.

1 Теоретическое обоснование решения задачи

1.1 Постановка задачи интерполирования

Пусть на отрезке $[a, b]$ определена некоторая функция $y = f(x)$, однако полная информация о ней недоступна. Известны лишь её значения в конечном числе точек $x_0, x_1, \dots, x_n$ этого отрезка, которые будем считать занумерованными в порядке возрастания:

$$a \le x_0 < x_1 < \cdots < x_i < x_{i+1} < \cdots < x_n \le b \qquad (2)$$

Требуется по известным значениям

$$y_i = f(x_i), i = 0, 1, \dots, n \qquad (3)$$

интерполировать, хотя бы приближённо, исходную функцию $y = f(x)$, т.е. построить на отрезке $[a, b]$ функцию $F(x)$, достаточно близкую к $f(x)$. Функцию $F(x)$ принято называть интерполирующей, точки $x = x_0, x = x_1, \dots, x = x_n$ - узлами интерполяции.

Подобные задачи часто возникают на практике, например при обработке экспериментальных данных, когда значение переменной y, зависящей от x, измеряется в конечном числе точек x_i: $y_i = f(x_i), i = 0, 1, \dots, n$, или при работе с табличными функциями, если требуется вычислить $y = f(x)$ при значениях аргумента, не совпадающего ни с одним из табличных x_i.

Поставленный выше в общей форме вопрос о приближении функций является достаточно сложным. Существует не один подход к его решению.

1.2 Описание метода

Сплайном называется функция, которая вместе с несколькими производными непрерывна на всем заданном отрезке $[a, b]$, а на каждом частичном отрезке $[x_i, x_{i+1}]$; в отдельности является некоторым алгебраическим многочленом. [2]

Пусть $h_i = x_{i+1} - x_i$.

Кубический сплайн $S_3(x)$, принимающий в узлах x_i, x_{i+1}соответственно значения y_i, y_{i+1}, имеет на частичном отрезке $[x_i, x_{i+1}]$ вид (формула 4).

$$S_3(x) = \frac{(x_{i+1}-x)^2(2(x-x_i)+h)}{h_i^3} y_i +$$

$$+ \frac{(x-x_i)^2(2(x_{i+1}-x)+h)}{h_i^3} y_{i+1} +$$

$$+ \frac{(x_{i+1}-x)^2(x-x_i)}{h_i^2} m_i + \frac{(x-x_i)^2(x-x_{i+1})}{h_i^2} m_{i+1} \qquad (4)$$

Простые вычисления показывают, что

$$S_3'(x_i) = m_i. \qquad (5)$$

Если отыскать $S_3''(x)$ и воспользоваться непрерывностью сплайна,

$$S_3''(x_i - 0) = S_3''(x_i + 0), \quad i = 1, 2, \dots, n-1 \qquad (6)$$

то можно перейти к следующей системе линейных алгебраических уравнений относительно наклонов:

$$m_{i-1} + 4m_i + m_{i+1} = \frac{3(y_{i+1} - y_{i-1})}{h_i}, \quad i = 1, 2, \dots, n-1 \qquad (7)$$

Так как неизвестных $n+1$, а СЛАУ состоит из $n-1$ уравнений, то для его решения методом трёхточечной прогонки необходимо знать m_0и m_{n+1}. Используя формулу (5), получается первый тип граничных условий:

$$m_0 = y_0, m_{n+1} = y_{n+1}. \qquad (8)$$

Сплайн имеет преимущество перед другими методами интерполяции, так как для его построения требуется знать только первую производную искомой функции. Сплайн может восстановить почти любую функцию, имеющую первую производную на заданном отрезке.

2 Описание программного средства

2.1 Укрупнённая схема алгоритма программного средства

Разработанное программное средство предназначено для решения поставленной задачи курсовой работы:

– Составление интерполирующей функции по заданным табличным точкам;
– Определение значений этой функции в указанных точках;
– Построение графика составленной функции.

Программа написана на языке программирования Swift. Swift — это надёжный и интуитивно понятный язык программирования от Apple, при помощи которого можно создавать приложения для iOS, Mac, Apple TV и Apple Watch [3].

В программе для построения пользовательского интерфейса использовался SwiftUI. SwiftUI — это набор инструментов для создания пользовательского интерфейса, который позволяет декларативно разрабатывать приложения.

Системные требования:

– iOS 16.0+ или iPadOS 16.0+ или macOS 13.0+;
– Минимум 50 МБ свободного пространства в хранилище.

Входные данные: файл JSON в виде массива объектов (см. таблицы 4.1, 4.2 и 4.3). Пример такого объекта изображён на рисунке 1.


```
{
    "name":"Вариант № 4",
    "boundaryConditions":{
        "a":0.00085,
        "b":0.08468
    },
    "findIn":[
        0.03,
        0.05,
        0.09,
        0.15,
        0.19
    ],
    "table":[
        {"x": 0.02, "y": 1.414216},
        {"x": 0.04, "y": 1.414236},
        {"x": 0.06, "y": 1.414290},
        {"x": 0.08, "y": 1.414395},
        {"x": 0.10, "y": 1.414567},
        {"x": 0.12, "y": 1.414824},
        {"x": 0.14, "y": 1.415183},
        {"x": 0.16, "y": 1.415661},
        {"x": 0.18, "y": 1.416274},
        {"x": 0.20, "y": 1.417039},
    ]
},
```


Рисунок 1 – Пример входных данных в формате JSON

Таблица 4.1 – описание объекта массива входных данных

Название	Тип данных	Назначение
name	строка	Название набора данных
boundaryConditions	объект BoundaryConditions	Граничные условия
findIn	массив чисел	Значения x_k^*, в которых необходимо вычислить значения соответствующих таблично заданных функций
table	массив объектов Table	Список узлов интерполяции

Таблица 4.2 – описание объекта BoundaryConditions

Название	Тип данных	Назначение
a	число	Значение производной функции в левой граничной точке
b	число	Значение производной функции в правой граничной точке

Таблица 4.3 – описание объекта Table

Название	Тип данных	Назначение
x	число	Значение x_i
y	число	Значение y_i

Укрупнённая схема алгоритма приложения представлена на рисунке 3.

При запуске программы появляется окно с выбором входных данных (см. рисунок 2).

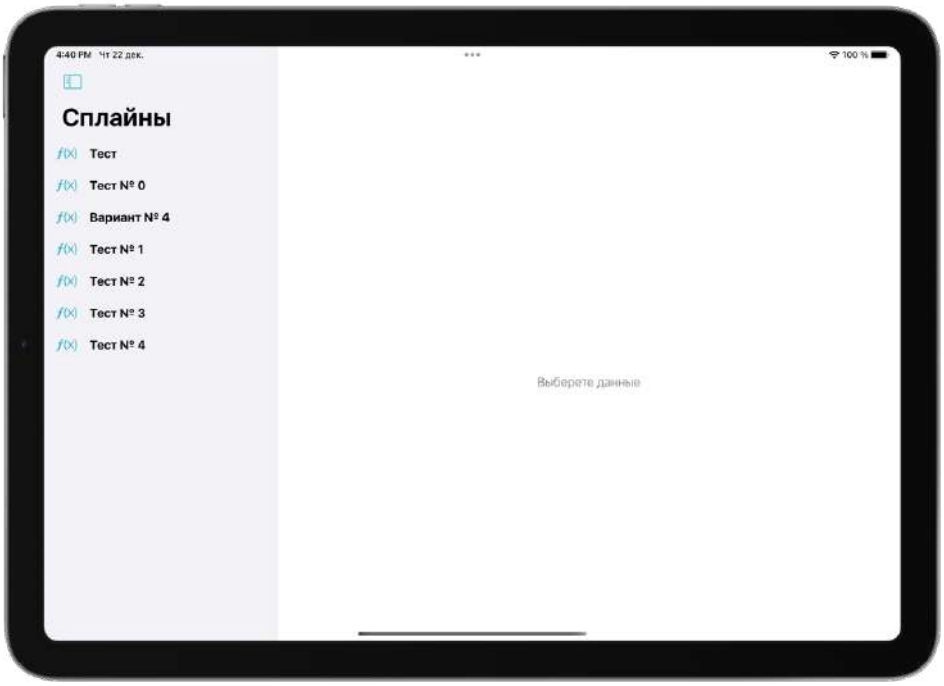

Рисунок 2 – Старт программы

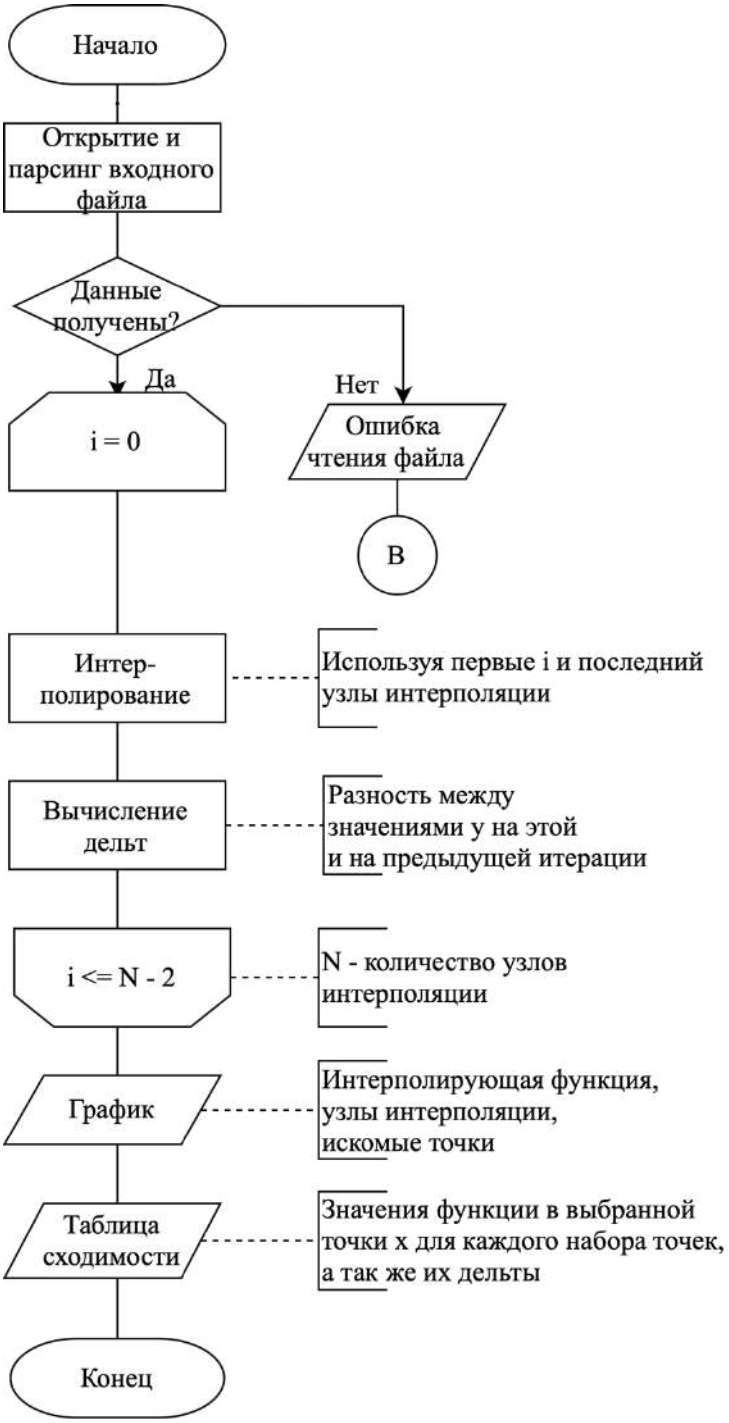

Рисунок 3 - Укрупнённая схема алгоритма приложения

В левом боковом меню можно выбрать набор данных по его имени. При нажатии на него, откроется график, с интерполирующей функцией (выделена зелёным), узлами интерполяции (синие точки) и искомыми точками (жёлтые). Боковое меню можно закрыть (см. рисунки 4 и 5).

Нажатие на кнопку с изображением английской строчной буквы i, вписанной в окружность, в правом нижнем углу откроет правое боковое меню (см. рисунок 6). В нём можно:

– выбрать количество заданных точек для построения графика;

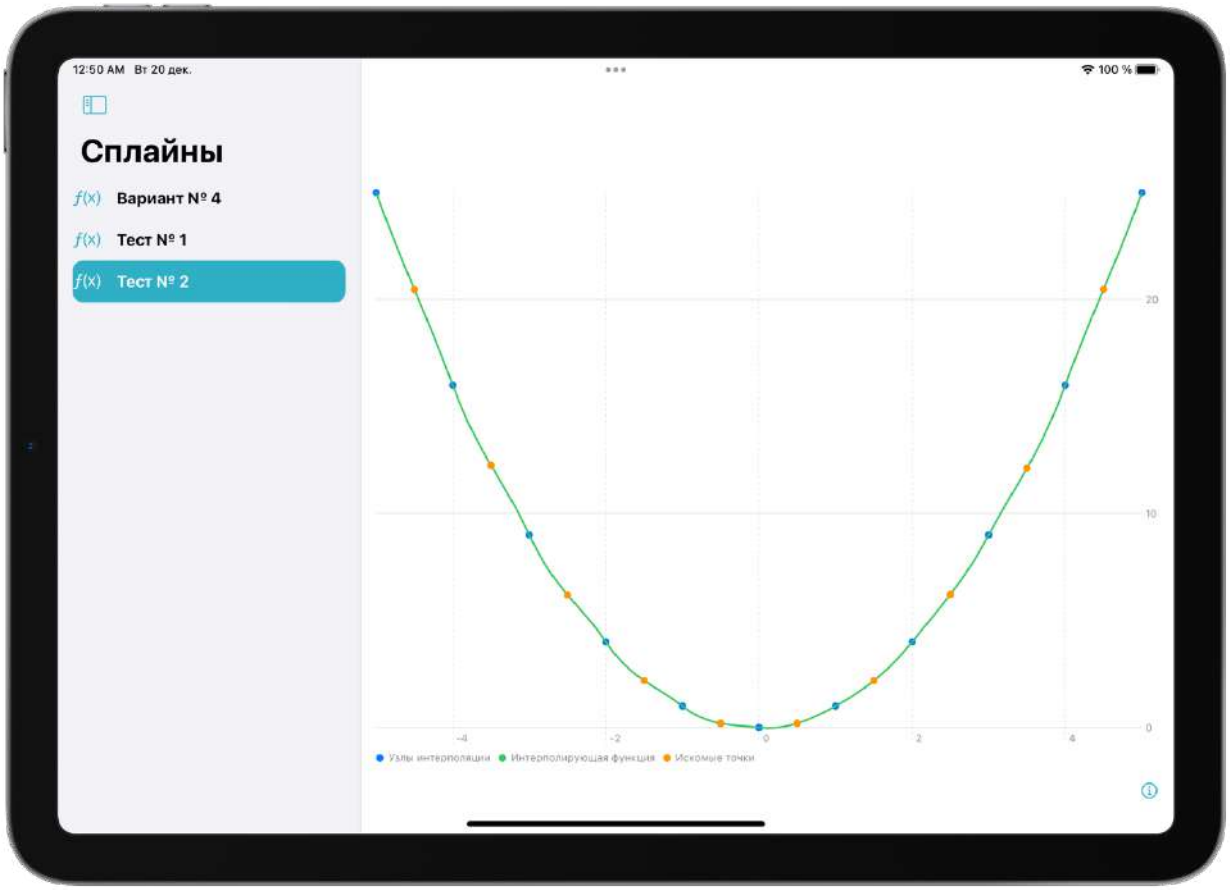

Рисунок 4 – Выбор входных данных

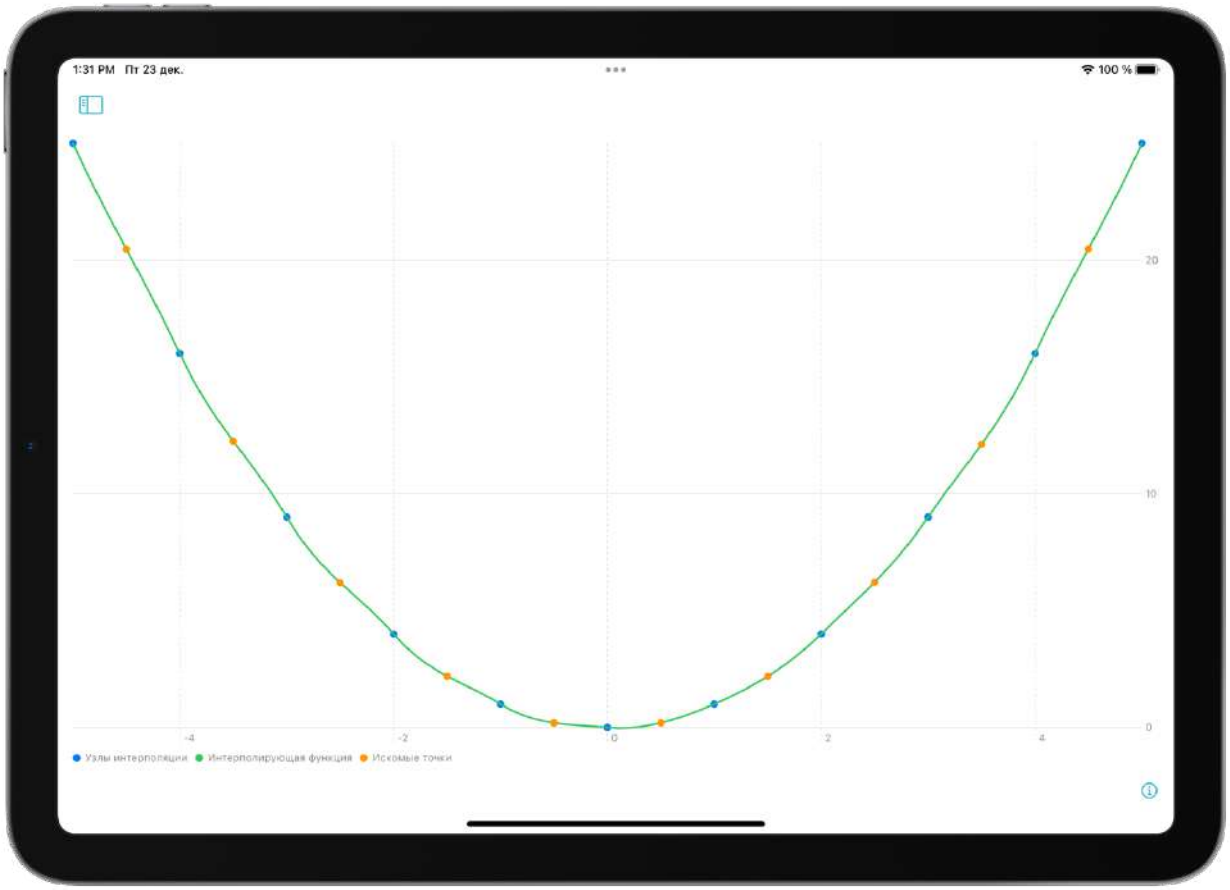

Рисунок 5 – График для выбранного набора данных

– Узнать значения функции в искомых точках для каждого набора заданных точек, а также разносность между значением функции в данном наборе и предыдущем наборе (кроме первого).

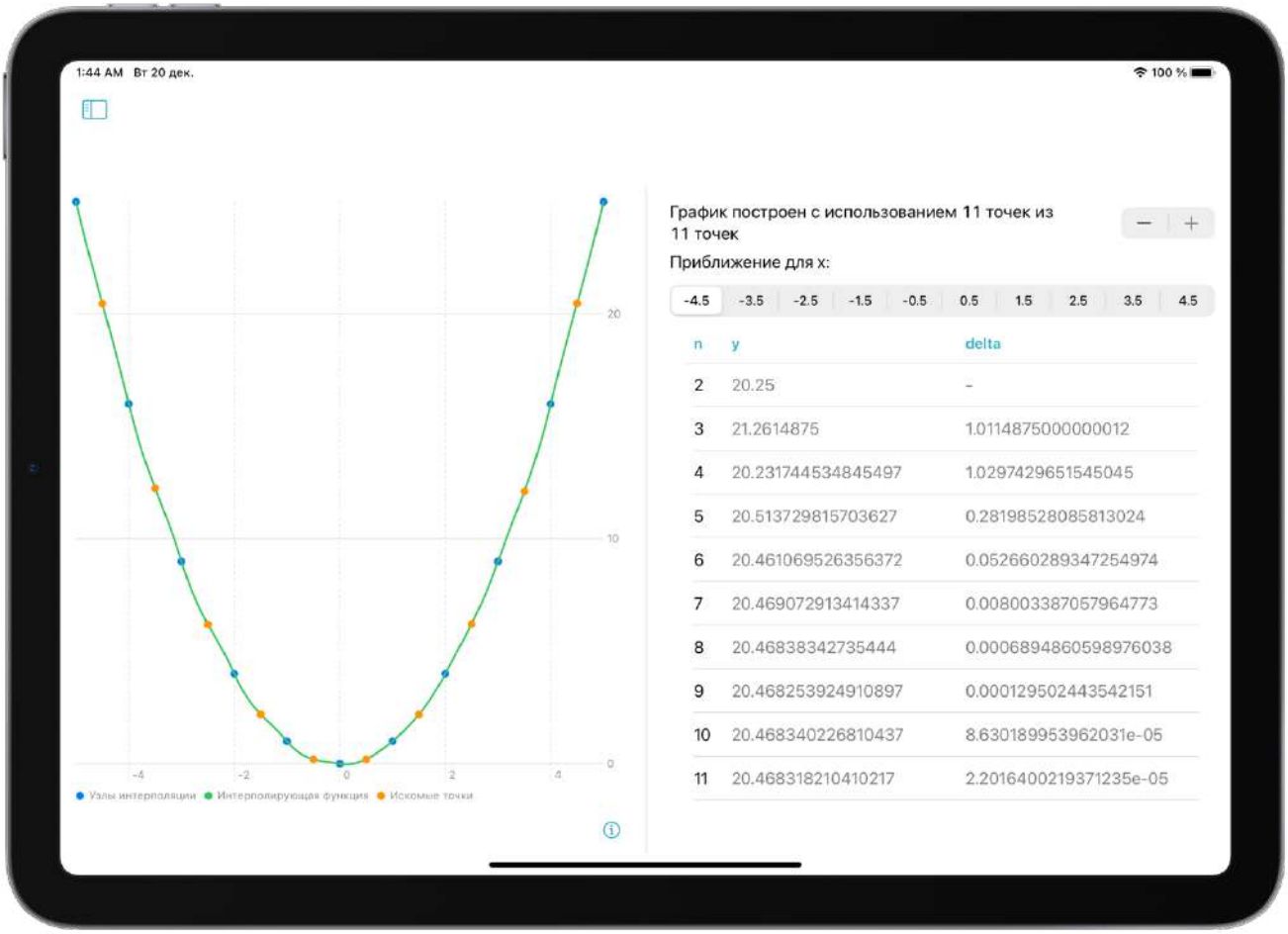

Рисунок 6 – Правое боковое меню

2.2 Описание модулей/подпрограмм

Проект под названием PIZVM.swiftpm состоит из 6 папок, которые разбивают проект на 6 логических модулей (см. таблицу 5).

Таблица 5 – Модули проекта

Название модуля	Файлы внутри	Назначение
Data	DataIn.swift, DataProvider.swift	Получение, хранение входных наборов данных
Logic	Calculator.swift, Compairing.swift, Interpolate.swift	Интерполяция и подготовка результатов для вывода на экран
Main	MyApp.swift	Точка входа программы
Resources	data.json	Ресурсы программы (входные параметры)

Продолжение таблицы 5

Название модуля	Файлы внутри	Назначение
Utils	DecodableIgnore.swift DecodableKey.swift Utils.swift	Вспомогательные методы
Views	ChartView.swift ContentView.swift DataRow.swift DataView.swift DynamicStack.swift MoreInfoView.swift	Показ пользовательского интерфейса

2.2.1 Модуль Data

Структуры и классы файлов модуля Data представлены в таблицах 6 и 7.

Таблица 6 – Структуры и классы в файле DataIn.swift

Тип	Название	Назначение
struct	DataIn	Хранение входного набора данных (см. таблицу 4.1)
struct	BoundaryConditions	Граничные условия (см. таблицу 4.2)
struct	PairXY	Пара значений x и y (см. таблицу 4.3)

Таблица 7 – Структуры и классы в файле DataProvider.swift

Тип	Название	Назначение
class	DataProvider	Парсинг входного файла и сохранение его результатов

2.2.2 Модуль Logic

Структуры и классы файлов модуля Logic представлены в таблицах 8, 9 и 10.

Таблица 8 – Структуры и классы в файле Calculator.swift

Тип	Название	Назначение
class	Calculator	Проведение интерполирования для разного числа узлов интерполяции, хранение этих результатов

Таблица 9 – Структуры и классы в файле Compairing.swift

Тип	Название	Назначение
struct	CompairingTable	Таблица значений функции в искомой точке
struct	CompairingRow	Строка таблицы CompairingTable

Таблица 10 – Структуры и классы в файле Interpolate.swift

Тип	Название	Назначение
class	Interpolate	Составление интерполирующей функции для входного набора узлов интерполяции, определение значений функции в искомых точках

2.2.3 Модуль Main

Структуры и классы файла модуля Main представлены в таблице 11.

Таблица 11 – Структуры и классы в файле MyApp.swift

Тип	Название	Назначение
struct	MyApp	Точка входа программы. Указывает окно при запуске программы

2.2.4 Модуль Resources

Содержит файл входных данных (см. пункт 2.1).

2.2.5 Модуль Utils

Файлы этого модуля содержат вспомогательные struct и extension, необходимые для упрощения работы некоторых модулей проекта.

2.2.6 Модуль Views

Структуры и классы файлов модуля Views представлены в таблицах 12, 13, 14, 15, 16 и 17.

Таблица 12 – Структуры и классы в файле ChartView.swift

Тип	Название	Назначение
struct	ChartView	Отрисовка графика

Таблица 13 – Структуры и классы в файле ContentView.swift

Тип	Название	Назначение
struct	ContentView	Меню выбора входных данных

Таблица 14 – Структуры и классы в файле DataRow.swift

Тип	Название	Назначение
struct	DataRow	Строка меню выбора входных данных

Таблица 15 – Структуры и классы в файле DataView.swift

Тип	Название	Назначение
struct	DataView	Строка меню выбора входных данных

Таблица 16 – Структуры и классы в файле DynamicStack.swift

Тип	Название	Назначение
struct	DynamicStack	Располагает элементы согласно текущей конфигурации или ориентации устройства.

Таблица 17 – Структуры и классы в файле MoreInfoView.swift

Тип	Название	Назначение
struct	MoreInfoView	Правое боковое меню (см. рисунок 6)

3 Тестирование программного средства

3.1 Описание тестовых наборов данных

Для проверки корректности результатов работы разработанного программного средства были заданы следующие тестовые наборы данных (см. таблицы 18, 19, 20 и 21).

Тестирование заключается в сравнении теоретической и абсолютной (практической) погрешностей решений. Теоретическая погрешность E в точках x^* вычислена при составлении тестового примера. Абсолютная погрешность вычисляется по формуле:

$$\varepsilon = |f(x^*) - \varphi(x^*)|, \quad (9)$$

где $f(x^*)$ - точное значение функции в точке x^*,

$\varphi(x^*)$ - значение интерполяционной функции в точке x^*, вычисленное в программе.

Таблица 18 – Описание тестовых наборов данных

Тестовый набор данных	Характеристики тестового набора данных								
	количество точек сетки	степень гладкости функции	отрезок, $[x_0;\ x_n]$	$M_{n+1} = \max_{[x_0;\ x_n]} f^{(n+1)}(x)$	граничные условия		значение функции в точках x^*		теоретическая оценка погрешности, E
					$S'(x_0)$	$S'(x_n)$	x_1^*	x_2^*	
$f_1(x) = \sin x$	11	12	[0; 3,14]	1	1	−0,99	0,197	0,479	0,00972117
$f_2(x) = \sin x$	11	12	[0; 0,2]	0,1987	1	0,98	0,009	0,029	$3{,}18 \cdot 10^{-4}$
$f_3(x) = e^x$	11	12	[0; 0,02]	1,0202	1	1,02	1,001	1,003	$1{,}63 \cdot 10^{-5}$

Таблица 19 – Табличные значения функции $f_1(x) = \sin x$

i	0	1	2	3	4	5
x_i	0	0,314	0,628	0,942	1,256	1,57
y_i	0	0,308866	0,587528	0,808736	0,950859	1

Продолжение таблицы 19

i	6	7	8	9	10
x_i	1,884	2,198	2,512	2,826	3,14
y_i	0,951351	0,809672	0,588816	0,310380	0,001593

Таблица 20 – Табличные значения функции $f_2(x) = \sin x$

i	0	1	2	3	4	5
x_i	0	0,02	0,04	0,06	0,08	0,1
y_i	0	0,019998667	0,039989334	0,059964006	0,079914694	0,099833417

Продолжение таблицы 20

i	6	7	8	9	10
x_i	0,12	0,14	0,16	0,18	0,2
y_i	0,119712207	0,139543115	0,159318207	0,179029573	0,198669331

Таблица 21 – Табличные значения функции $f_3(x) = e^x$

i	0	1	2	3	4	5
x_i	0	0,002	0,004	0,006	0,008	0,01
y_i	1	1,002002001	1,004008011	1,006018036	1,008032086	1,010050167

Продолжение таблицы 21

i	6	7	8	9	10
x_i	0,012	0,014	0,016	0,018	0,02
y_i	1,012072289	1,014098459	1,016128685	1,018162976	1,020201340

Данные функции обладают достаточной степенью гладкости и имеют разные максимальные значения производных ($M_{n+1} = \max\limits_{[x_0;\, x_n]} f^{(n+1)}(x)$) на заданном отрезке.

Ожидаемый результат работы программы: значения интерполирующей функции в искомых точках достаточно приближено к точным значениям, то есть абсолютная погрешность не больше теоретической.

3.2 Счёт, анализ и интерпретация результатов тестирования

3.2.1 Первый тестовый набор

После выполнения программой первого тестового набора был получен следующий график интерполирующей функции (см. рисунок 7). Анализ значений интерполирующей в искомых точках приведён в таблице 22.

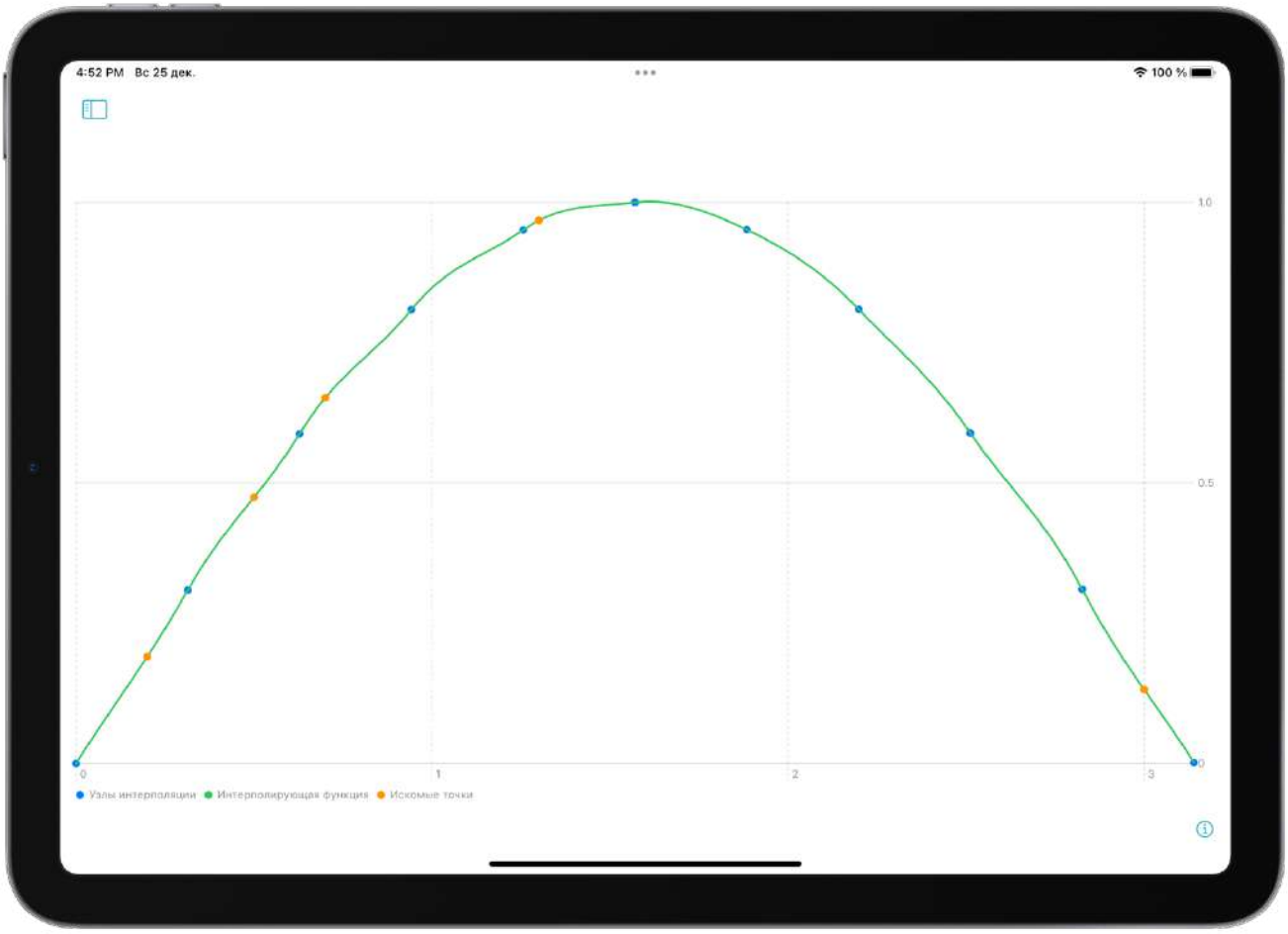

Рисунок 7 – График интерполирующей функции первого тествого набора

Таблица 22 - Анализ результатов работы программы

№	x^*	$f(x^*)$	$\varphi(x^*)$	$E(x^*)$	ε
1	0,2	0,19866933	0,19030550	0,00972117	0,00836383
2	0,5	0,47942554	0,47434195	0,00972117	0,00508359

$f(x^*)$ - точное значение функции в точке x*;

$\varphi(x^*)$ - значение функции, вычисленное с помощью интерполирующей функции;

$E(x^*)$ - теоретическое значение погрешности;

ε – значение абсолютной погрешности в точке, вычисляемое по формуле (9).

3.2.2 Второй тестовый набор

После выполнения программой второго тестового набора был получен следующий график интерполирующей функции (см. рисунок 8). Анализ значений интерполирующей в искомых точках приведён в таблице 23.

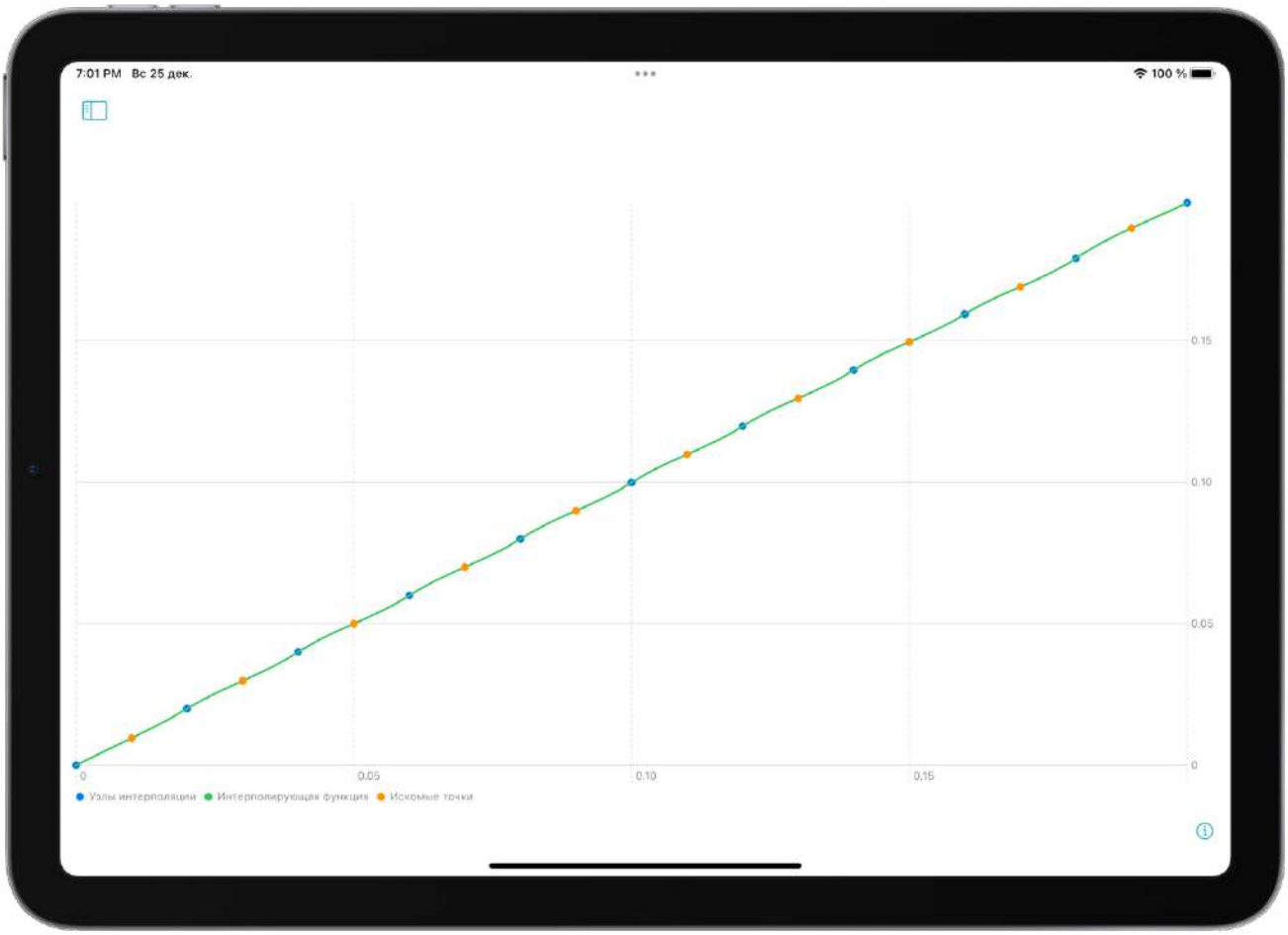

Рисунок 8 – График интерполирующей функции второго тествого набора

Таблица 23 - Анализ результатов работы программы

№	x^*	$f(x^*)$	$\varphi(x^*)$	$E(x^*)$	ε
1	0,01	0,00999983	0,00953959	$3{,}18 \cdot 10^{-4}$	0,00046024
2	0,03	0,02999550	0,02980296	$3{,}18 \cdot 10^{-4}$	0,00019254

$f(x^*)$ - точное значение функции в точке x*;

$\varphi(x^*)$ - значение функции, вычисленное с помощью интерполирующей функции;

$E(x^*)$ - теоретическое значение погрешности;

ε – значение абсолютной погрешности в точке, вычисляемое по формуле (9).

3.2.3 Третий тестовый набор

После выполнения программой третьего тестового набора был получен следующий график интерполирующей функции (см. рисунок 9). Анализ значений интерполирующей в искомых точках приведён в таблице 24.

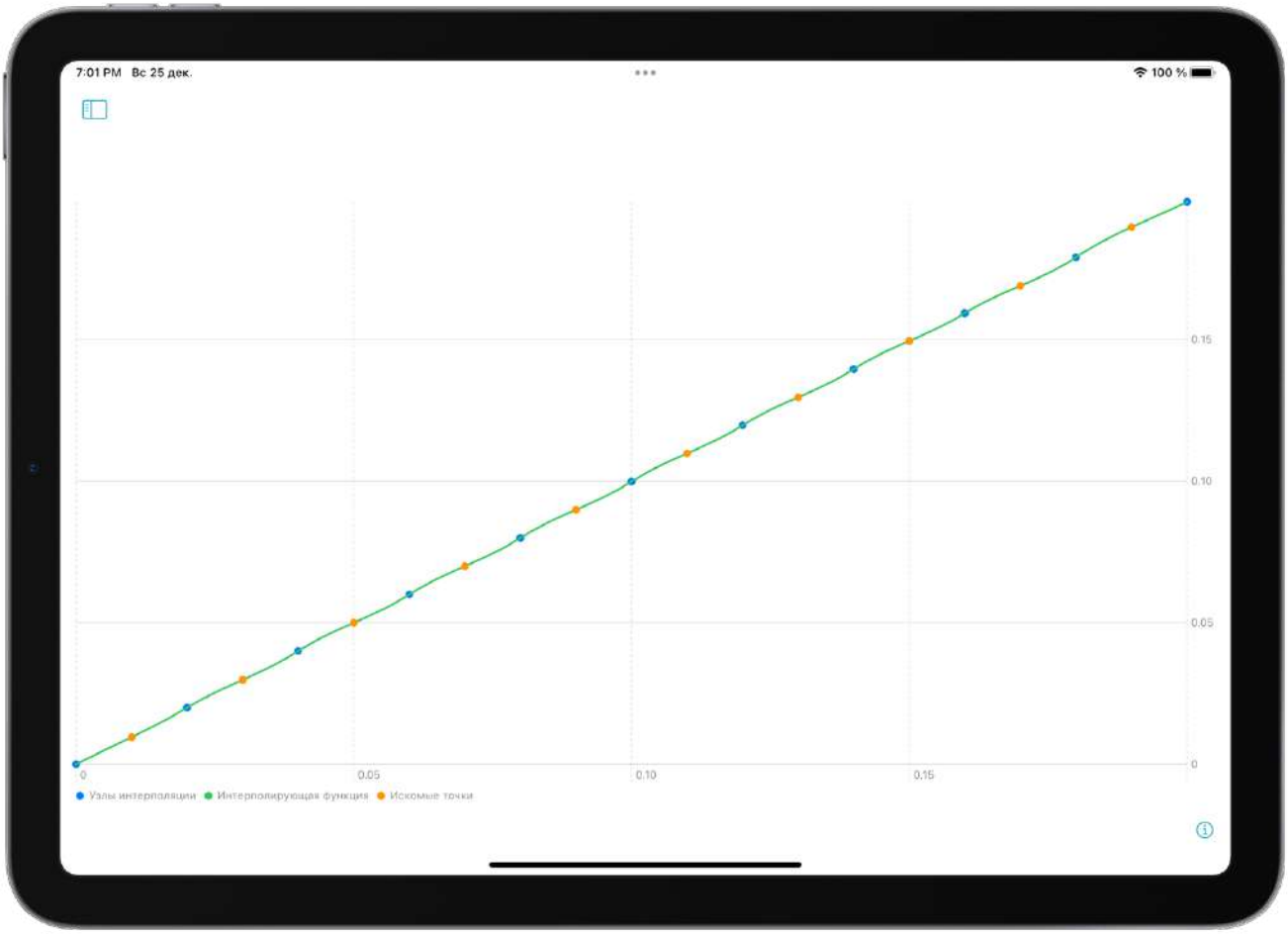

Рисунок 9 – График интерполирующей функции третьего тествого набора

Таблица 24 - Анализ результатов работы программы

№	x^*	$f(x^*)$	$\varphi(x^*)$	$E(x^*)$	ε
1	0,001	1,0010005	1,00095442	$1{,}63 \cdot 10^{-5}$	0,00004608
2	0,003	1,0030045	1,00298498	$1{,}63 \cdot 10^{-5}$	0,00001952

$f(x^*)$ - точное значение функции в точке x*;

$\varphi(x^*)$ - значение функции, вычисленное с помощью интерполирующей функции;

$E(x^*)$ - теоретическое значение погрешности;

ε – значение абсолютной погрешности в точке, вычисляемое по формуле (9).

В результате анализа проведённого теста, можно заключить, что полученная абсолютная погрешность имеет тот же порядок, что теоретическая. Следовательно разработанное программное средство работает корректно.

4 Анализ и интерпретация результатов поставленной задачи

После выполнения программой заданного по варианту набора был получен следующий график интерполирующей функции (см. рисунок 10). Результаты вычислений искомых точек см. в таблицах 25, 26, 27, 28 и 29.

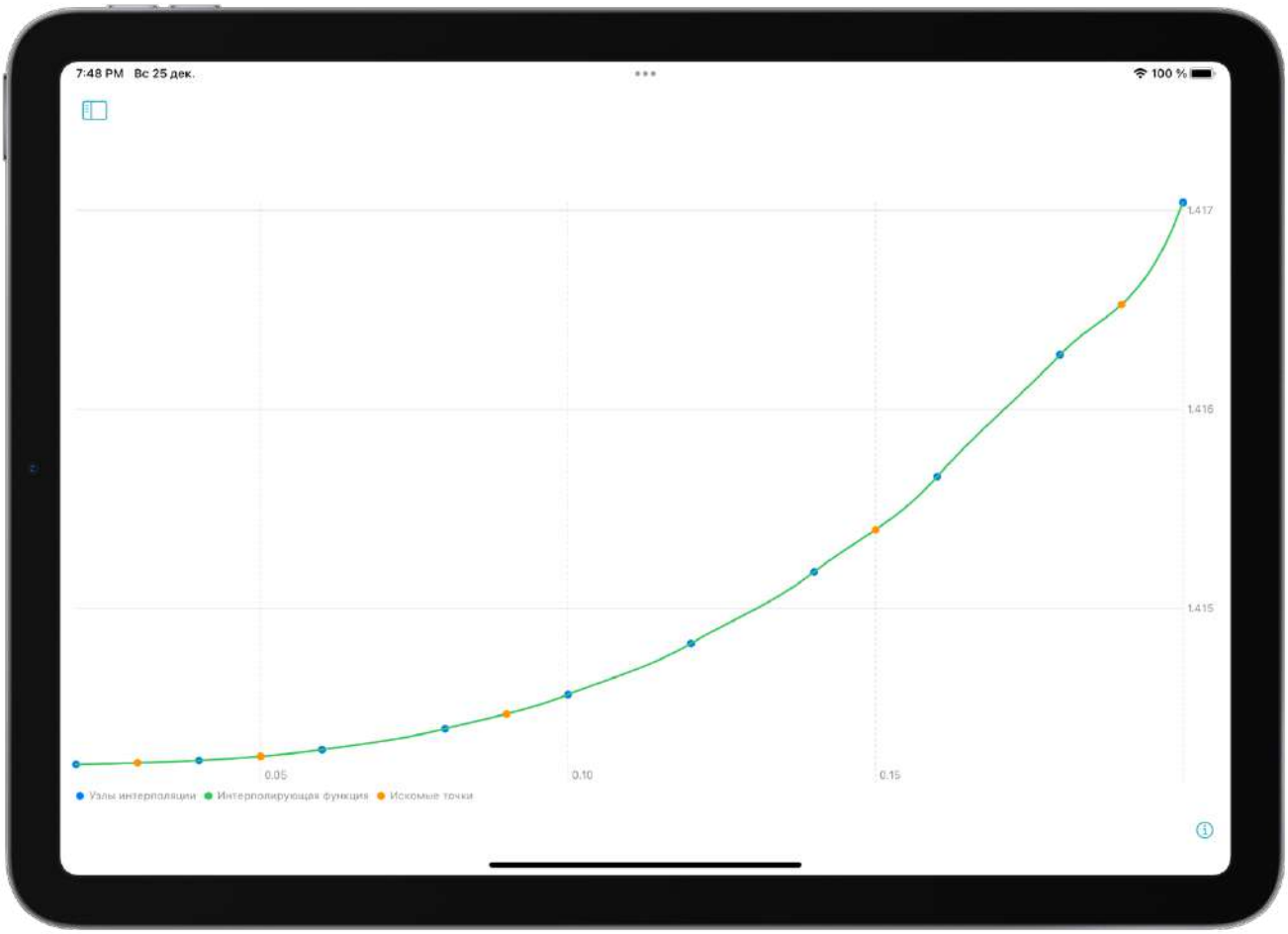

Рисунок 10 – График интерполирующей функции заданного набора

Таблица 25 – Результаты вычислений в точке $x_1^* = 0{,}03$

Количество точек сетки i	$\varphi_i(x_1^*)$	$\Delta\lvert\varphi_i(x_1^*) - \varphi_{i-1}(x_1^*)\rvert$
2	1,4142043217078188	–
3	1,4142243063989441	$1{,}998469112529655 \cdot 10^{-5}$
4	1,4142246504508424	$3{,}440518983044427 \cdot 10^{-7}$
5	1,4142233472756665	$1{,}3031751759928767 \cdot 10^{-6}$
6	1,4142240817459866	$7{,}344703201184188 \cdot 10^{-7}$
7	1,4142237842341299	$2{,}97511856706123 \cdot 10^{-7}$

Продолжение таблицы 25

Количество точек сетки i	$\varphi_i(x_1^*)$	$\Delta\lvert\varphi_i(x_1^*) - \varphi_{i-1}(x_1^*)\rvert$
8	1,4142238826256588	$9{,}839152892432423 \cdot 10^{-8}$
9	1,4142238587165135	$2{,}390914533023647 \cdot 10^{-8}$
10	1,4142238594643215	$7{,}478080377154583 \cdot 10^{-10}$

Таблица 26 – Результаты вычислений в точке $x_2^* = 0{,}05$

Количество точек сетки i	$\varphi_i(x_2^*)$	$\Delta\lvert\varphi_i(x_2^*) - \varphi_{i-1}(x_2^*)\rvert$
2	1,414089986111111	–
3	1,4142312865857438	0,0001413004746328994
4	1,4142526280582888	$2{,}1341472544955664 \cdot 10^{-5}$
5	1,4142591439341685	$6{,}515875879742339 \cdot 10^{-6}$
6	1,4142554715825677	$3{,}672351600814139 \cdot 10^{-6}$
7	1,4142569591418512	$1{,}487559283530615 \cdot 10^{-6}$
8	1,4142564671842068	$4{,}919576443995766 \cdot 10^{-7}$
9	1,4142565867299333	$1{,}1954572642913774 \cdot 10^{-7}$
10	1,4142565829908933	$3{,}739039966532687 \cdot 10^{-9}$

Таблица 27 – Результаты вычислений в точке $x_3^* = 0{,}09$

Количество точек сетки i	$\varphi_i(x_3^*)$	$\Delta\lvert\varphi_i(x_3^*) - \varphi_{i-1}(x_3^*)\rvert$
2	1,413778245781893	–
3	1,4140125591388881	0,00023431335699508615
4	1,4142894470358547	0,00027688789696656
5	1,4144738593881783	0,00018441235232358189
6	1,414453907858007	$1{,}995153017131379 \cdot 10^{-5}$
7	1,4144750311998326	$2{,}1123341825690645 \cdot 10^{-5}$
8	1,414468045401283	$6{,}985798549630218 \cdot 10^{-6}$

Продолжение таблицы 27

Количество точек сетки i	$\varphi_i(x_3^*)$	$\Delta\lvert\varphi_i(x_3^*) - \varphi_{i-1}(x_3^*)\rvert$
9	1,4144697429505972	$1{,}6975493142279419 \cdot 10^{-6}$
10	1,414469689856228	$5{,}309436934552991 \cdot 10^{-8}$

Таблица 28 – Результаты вычислений в точке $x_4^* = 0{,}15$

Количество точек сетки i	$\varphi_i(x_4^*)$	$\Delta\lvert\varphi_i(x_4^*) - \varphi_{i-1}(x_4^*)\rvert$
2	1,4143065920781892	–
3	1,414404071625631	$9{,}747954744177001 \cdot 10^{-5}$
4	1,4145513577933488	0,00014728616771786385
5	1,414736070740478	0,00018471294712907316
6	1,4149610243880626	0,00022495364758468028
7	1,4152170607852068	0,00025603639714422144
8	1,4154212846986522	0,00020422391344543378
9	1,4153921193209282	$2{,}9165377724016395 \cdot 10^{-5}$
10	1,4153948794803106	$2{,}7601593823867177 \cdot 10^{-6}$

Таблица 29 – Результаты вычислений в точке $x_5^* = 0{,}19$

Количество точек сетки i	$\varphi_i(x_5^*)$	$\Delta\lvert\varphi_i(x_5^*) - \varphi_{i-1}(x_5^*)\rvert$
2	1,4162589505144032	–
3	1,416264158168216	$5{,}207653812844271 \cdot 10^{-6}$
4	1,4162724516929126	$8{,}293524696556531 \cdot 10^{-6}$
5	1,4162837498433696	$1{,}1298150456973133 \cdot 10^{-5}$
6	1,4162994657559407	$1{,}5715912571101498 \cdot 10^{-5}$
7	1,416322338173286	$2{,}2872417345221052 \cdot 10^{-5}$
8	1,4163571458286193	$3{,}4807655333368714 \cdot 10^{-5}$
9	1,4164147580279576	$5{,}76121993383083 \cdot 10^{-5}$

Продолжение таблицы 29

Количество точек сетки i	$\varphi_i(x_5^*)$	$\Delta\lvert\varphi_i(x_5^*) - \varphi_{i-1}(x_5^*)\rvert$
10	1,4165264592374256	0,00011170120946801809

Узлы интерполяции добавлялись на каждом приближении по очереди:

– Первая сетка: первый и последний узлы интерполяции;

– Вторая сетка: первый, второй и последний узлы интерполяции;

– Третья сетка: первый, второй, третий и последний узлы интерполяции;

– …

– $N - 1$ сетка (последняя): все узлы сетки.

Изучив приведённые выше таблицы, можно сделать следующий вывод.

Для точек, расположенных близко к правой границе отрезка ($x_1^* = 0{,}03$ и $x_2^* = 0{,}05$) наблюдается постоянное уменьшение модуля разности значений интерполирующей функции в этих точках на двух соседних сетках. И наоборот, для точек, отдалённых от правой границы отрезка ($x_4^* = 0{,}15$ и $x_5^* = 0{,}19$) наблюдается постоянное увеличение модуля разности значений интерполирующей функции в этих точках на двух соседних сетках.

Сходимость данного метода наблюдается только в точках, стоящих близко к правой границе отрезка, из-за особенности выбора точек для сетки.

Заключение

Рассмотрим результаты выполнения задания, целью которого являлась разработка программного средства для проведения сплайн-интерполяции.

Было получено наилучшее приближение значений интерполирующей функции в искомых точках:

– $x_1^* = 0{,}03, y_1^* = 1{,}4142238594643215$;
– $x_2^* = 0{,}05, y_1^* = 1{,}4142565829908933$;
– $x_3^* = 0{,}09, y_1^* = 1{,}4144696898562280$;
– $x_4^* = 0{,}15, y_1^* = 1{,}4153948794803106$;
– $x_5^* = 0{,}19, y_1^* = 1{,}4165264592374256$.

Для достижения цели работы были решены следующие инженерные задачи:

– Изучение алгоритма построения сплайна первыми граничными условиями;
– Написание программного средства на языке программирования Swift;
– Проведение тестовых вычислений;
– Получение наилучшего приближения значений интерполирующей функции в искомых точках.

Тестирование программного средства показало, что программа успешно выполняет поставленную задачу.

Таким образом, поставленные инженерные задачи выполнены, цель курсовой работы достигнута.

Список использованных источников

1 Ююкин И.В. ПРИМЕНЕНИЕ МЕТОДА СПЛАЙН-ФУНКЦИЙ ПРИ КОМПЬЮТЕРНОЙ ВИЗУАЛИЗАЦИИ ПОДВОДНОГО РЕЛЬЕФА. Вестник Государственного университета морского и речного флота имени адмирала С. О. Макарова. 2021;13(1):64-79. https://doi.org/10.21821/2309-5180-2021-13-1-64-79 (дата обращения 26.12.2022).

2 Волков Е. А. Численные методы: Учеб. пособие для вузов. - 2-е изд., испр. - М.: Наука. Гл. ред. физ.-мат. лит., 1987.- 248 с.

3 Swift. [Электронный ресурс]. Режим доступа: https://www.apple.com/ru/swift/ (дата обращения 26.12.2022).

Приложение А

(обязательное)

Фрагмент программы

```
import Foundation
/*
 Составление интерполирующей функции для входного набора узлов интерполяции, определение значений функции в искомых точках
 */
class Interpolate {
    var resTables: [PairXY] = []
    var findTables: [PairXY] = []
    var allPoints: [PairXY] = []

    var accuracyLenth: Int = 0

    private let boundaryConditions: BoundaryConditions
    private let findIn: [Double]
    var table: [PairXY]

    private let pointsNumber = 100

    init(dataIn: DataIn, n: Int) {
        self.boundaryConditions = dataIn.boundaryConditions
        self.findIn = dataIn.findIn
        self.table = [dataIn.table.first!]
        if n > 0 {
            for i in 1...n {
                self.table.append(dataIn.table[i])
            }
        }
        self.table.append(dataIn.table.last!)
        calculate()
    }

    private func calculate() {

        let xs = table.map { $0.x }
        let ys = table.map { $0.y }
        let n = xs.count

        let h = Array<Double>.create(n - 1) { i in
            xs[i + 1] - xs[i]
        }

        let sigma = Array<Double>.create(n - 2) { i in
            ((-h[i + 1] / 3) + (h[i] / 3)) / 2
        }

        let A = Array<Double>.create(n - 2) { i in
```

```
    (-2 / h[i]) + sigma[i] * (6 / h[i].power(2))
}

let B = Array<Double>.create(n - 2) { i in
    let s1 = sigma[i] * (6 / h[i].power(2))
    let s2 = sigma[i] * (6 / h[i + 1].power(2))
    return (-4 / h[i + 1]) - (4 / h[i]) - s2 + s1
}

let C = Array<Double>.create(n - 2) { i in
    (-2 / h[i + 1]) - sigma[i] * (6 / h[i + 1].power(2))
}

let D = Array<Double>.create(n - 2) { i in
    (-6 / h[i + 1]) - sigma[i] * (12 / h[i + 1].power(2))
}

let E = Array<Double>.create(n - 2) { i in
    (-6 / h[i]) + sigma[i] * (12 / h[i].power(2))
}

var alphaLast = 0.0
let alpha = Array<Double>.create(n - 1) { i in
    if (i != 0) {
        alphaLast = -C[i - 1] / (A[i - 1] * alphaLast + B[i - 1])
    }
    return alphaLast
}

var betaLast = boundaryConditions.a
let beta = Array<Double>.create(n - 1) { i in
    if (i != 0) {
        let s1 = (A[i - 1] * alpha[i - 1] + B[i - 1])
        let s2 = D[i - 1] * (ys[i + 1] - ys[i])
        let s3 = E[i - 1] * (ys[i] - ys[i - 1]) / h[i - 1]
        betaLast =
        ((s2 / h[i] + s3) - alpha[i] * betaLast) / s1
    }
    return betaLast
}

var mLast = boundaryConditions.b
var m = Array<Double>.create(n) { it in
    let i = n - it - 1
    if (it != 0) {
        mLast = alpha[i] * mLast + beta[i]
    }
    return mLast
}
m.reverse()

let xStart = xs.first!
```

```
        let xLast = xs.last!
        let accuracy = abs(xLast - xStart) / Double(pointsNumber)

        let accuracySecondPart = accuracy.description.split(separator: ".")[1]
        if let index = accuracySecondPart.index(of: "000000") ?? accuracySecondPart.index(of:
"999999") {
            accuracyLenth = accuracySecondPart[..<index].count
        } else {
            accuracyLenth = accuracySecondPart.count
        }

        var i = 0

        func f(x: Double, i _i: Int = -1) -> PairXY {

            var i = _i

            if (i == -1) {
                i = 0
                while !(xs[i]...xs[i + 1] ~= x) {
                    i += 1
                }
            }
            let t = (x - xs[i]) / h[i]
            let s1 = 1 - 3 * t.power(2) + 2 * t.power(3)
            let s2 = 3 * t.power(2) - 2 * t.power(3)
            let s31 = 1 - 2 * t + t.power(2)
            let s3 = m[i] * h[i] * t * s31
            let s41 = t.power(2) * (t - 1)
            let s4 = m[i + 1] * h[i] * s41
            let Sc = ys[i] * s1 + ys[i + 1] * s2 + s3 + s4
            return PairXY(x: x, y: Sc)
        }

        for xInter in stride(from: xStart, through: xLast, by: accuracy) {

            if xInter > xs[i + 1] {
                i += 1
            }

            resTables.append(f(x: xInter, i: i))
        }

        resTables.append(table.last!)

        for x in findIn {
            findTables.append(f(x: x))
        }

        allPoints = (table + findTables).sorted(by: {$0.x < $1.x})
    }
}
```