«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 — Программная инженерия
Профиль	Без профиля
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой Лисс А.А.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: РАЗРАБОТКА ИНСТРУМЕНТА МОДЕЛИРОВАНИЯ И ВИЗУАЛИЗАЦИИ ФОРМЫ ПРОСТРАНСТВЕННОЙ КРИВОЙ, ВОССТАНОВЛЕННОЙ ПО ДАННЫМ, ИЗМЕНЯЩИМСЯ В РЕАЛЬНОМ ВРЕМЕНИ.

Студент		_____ *подпись*	Попов Д.С.
Руководитель	д.т.н. *(Уч. степень, уч. звание)*	_____ *подпись*	Лисс А.Р.
Консультант	к.т.н. *(Уч. степень, уч. звание)*	_____ *подпись*	Сергеева Е.И.

Санкт-Петербург

2023

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. кафедрой МО ЭВМ

_____ Лисс А.А.

«___»_____2023 г.

Студент Попов Д.С. Группа 9304

Тема работы: Разработка инструмента моделирования и визуализации формы пространственной кривой, восстановленной по данным, изменяющимся в реальном времени

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МО ЭВМ

Исходные данные (технические требования):

Необходимо разработать инструмент моделирования и визуализации формы пространственной кривой

Содержание ВКР: Введение, Обзор предметной области, Формулировка требований к решению и постановка задачи, Архитектурная программная реализация, .

Перечень отчетных материалов: пояснительная записка, иллюстративный материал

Дополнительные разделы: нормативно-правовое регулирование интеллектуальной деятельности

Дата выдачи задания «___»_____2023 г.

Дата представления ВКР к защите «___»_____2023 г.

Студент _____ Попов Д.С.

Руководитель д.т.н. _____ Лисс А.Р.

(Уч. степень, уч. звание)

Консультант к.т.н. _____ Сергеева Е.И.
(Уч. степень, уч. звание)

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой МО ЭВМ

_____ Лисс А.А.

«___»_____2023 г.

Студент Попов Д.С. Группа 9304

Тема работы: Разработка инструмента моделирования и визуализации формы пространственной кривой, восстановленной по данным, изменяющимся в реальном времени

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	00.00 – 00.00
2	Обзор предметной области	00.00 – 00.00
3	Формулировка требований к решению и постановка задачи	00.00 – 00.00
4	Архитектурная программная реализация	00.00 – 00.00
5	Исследование разработанного инструмента	00.00 – 00.00
6	Оформление иллюстративного материала	00.00 – 00.00
7	Оформление пояснительной записки	00.00 – 00.00
8	Оформление иллюстративного материала	00.00 – 00.00
9	Предзащита	00.00 – 00.00

Студент _____ Попов Д.С.

Руководитель д.т.н _____ Лисс А.Р.

(Уч. степень, уч. звание)

Консультант к.т.н _____ Сергеева Е.И.

(Уч. степень, уч. звание) _____

РЕФЕРАТ

Пояснительная записка 00 стр., 00 рис., 00 табл., 00 ист., 00 прил.

БИБЛИОТЕКА, АЛГОРИТМ, МЕТОД ВОССТАНОВЛЕНИЯ

Объектом исследования являются буксируемые гибкие антенны с системой ориентации.

Предметом исследования методы оценки формы антенны.

Цель работы: выборе оптимального существующего метода восстановления формы пространственной кривой по данным, изменяющимся в реальном времени, и создании соответствующего инструмента для моделирования и визуализации полученной формы.

Результаты этой работы будут полезны для повышения точности и эффективности гидроакустических систем, основанных на гибких пространственных кривых.

ABSTRACT

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ВВЕДЕНИЕ

В современном мире гидроакустические системы играют важную роль в многих областях, таких как геофизика, морское исследование и военное дело. Одним из ключевых элементов таких систем является способность восстанавливать форму гибких пространственных кривых, таких как гидроакустические антенны, по данным, изменяющимся в реальном времени. Эта задача требует восстановления формы кривой с высокой степенью точности, особенно при работе в сложных условиях окружающей среды. В настоящее время существует множество методов восстановления формы гибких пространственных кривых по данным в реальном времени, включая методы, основанные на кривых Безье, кривых Б-сплайнов, кривых Катмулла-Рома, а также методы, использующие многочлены и рациональные функции. Каждый метод имеет свои преимущества и недостатки, выбор оптимального метода зависит от конкретных условий и требований. Однако, существующие методы восстановления формы кривой, не всегда обеспечивают необходимую точность, особенно при работе в сложных условиях окружающей среды. Кроме того, многие методы требуют вычислительно сложных алгоритмов, что может затруднять их использование в реальном времени.

ОПИСАНИЕ ОСНОВНЫХ МЕТОДОВ ИНТЕРПОЛЯЦИИ.

2.1 Понятие интерполяции.

Интерполяция является одним из основных методов аппроксимации функций и широко используется в различных областях, где требуется вычисление значений функции между заданными точками. В общем случае интерполяция — это процесс нахождения функции, которая проходит через некоторый набор точек. На рисунке 2.1 показан пример интерполяции, где красные точки представляют собой известные значения функции, через которые проводится синяя линия, представляющая интерполяционную функцию. Таким образом, можно вычислить промежуточные значения функции в других точках.

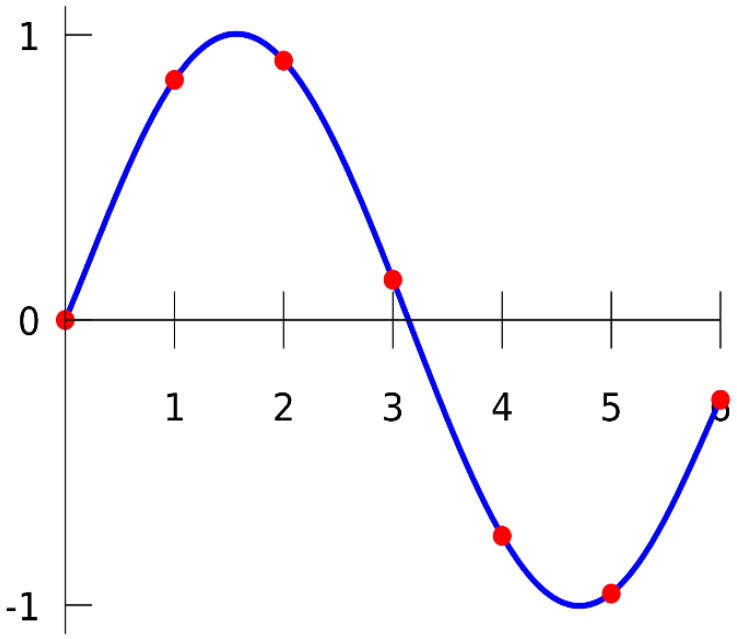

Рисунок 2.1 – Пример работы интерполяции

Существует несколько разновидностей интерполяции, включая полиномиальную, дробно-рациональную, сплайновую и кусочно-полиномиальную. В данном разделе мы рассмотрим основные методы интерполяции и принципы их работы.

2.2 Полиномиальная интерполяция.

Полиномиальная интерполяция — это метод, который позволяет приближенно вычислить значения функции $f(x)$ на заданном отрезке $[a, b]$ по ее значениям в узлах интерполяции $x_0, x_1, \dots, x_n$. При этом предполагается, что функция $f(x)$ достаточно гладкая и определена на всем отрезке $[a, b]$.

Полиномиальная интерполяция основана на использовании многочлена Лагранжа или многочлена Ньютона. Многочлен Лагранжа определяется следующей формулой:

$$L_n(x) = \sum_{j=0}^{n} y_i \, l_i(x),$$

где $y_i = f(x_i)$ — значения функции в узлах интерполяции, а $l_i(x)$ — полиномы Лагранжа, которые определяются следующим образом:

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Многочлен Ньютона выглядит следующим образом:

$$N_i(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0) \dots (x - x_{n-1}),$$

где $a_0, a_1, \dots, a_n$ — разделенные разности, которые могут быть найдены с помощью таблицы разделенных разностей.

Оба этих метода позволяют построить многочлен, который проходит через все заданные узлы интерполяции и приближенно описывает функцию $f(x)$ на всем отрезке $[a, b]$.

2.2.1 Интерполяция Эрмита.

Интерполяция Эрмита — это метод интерполяции, который использует производные функции в заданных точках для создания интерполяционного полинома. Этот метод может быть использован для интерполяции функций с разрывами или особенностями, и он обеспечивает более точные результаты, чем полиномиальная интерполяция.

Интерполяционный полином Эрмита $H_n(x)$ порядка n может быть записан в виде:

$$H_n(x) = \sum_{i=0}^{n} f\,(x_i) h_i(x)$$

где $f(x_i)$ — значение функции в точке x_i, $h_i(x)$ - интерполяционный полином, который задается формулой:

$$h_i(x) = [1 - 2(x - x_i)L_i'(x_i) + (x - x_i)L_i^{"}(x_i)](L_i(x))^2$$

здесь $L_i(x)$ — полином Лагранжа степени n:

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_i}{x_i - x_j}$$

А $L_i'(x_i)$ и $L_i^{"}(x_i)$ — первая и вторая производные полинома Лагранжа в точке x_i:

$$L_i'(x_i) = \sum_{j=0, j\neq i}^{n} \frac{1}{x_i - x_j},$$

$$L_i^{"}(x_i) = \sum_{j=0, j\neq i}^{n} \frac{2}{(x_i - x_j)^2}$$

2.2.2 Интерполяция Уиттакера-Шеннона.

Интерполяция Уиттакера-Шеннона, также известная как семплирование Шеннона или теорема Котельникова-Шеннона, используется для восстановления непрерывной функции $f(x)$ по ее дискретному набору значений $f(n)$, взятым с некоторым шагом дискретизации T. Формула для интерполяции Уиттакера-Шеннона выглядит следующим образом:

$$f(x) = \sum_{n=-\infty}^{\infty} f(n) \cdot sinc(\frac{x - nT}{T}),$$

где $sinc(x) = \frac{\sin(\pi x)}{\pi x}$ — функция синк.

Данная формула утверждает, что функция $f(x)$ может быть полностью восстановлена из ее значений $f(n)$ при условии, что шаг дискретизации T не превосходит половины наименьшей периодической составляющей функции $f(x)$.

2.3 Дробно-рациональная интерполяция.

Некоторые функции не могут быть приближены полиномами с достаточной точностью, либо полиномиальное приближение сходится очень медленно. В таких случаях дробно-рациональное приближение (иногда называемое рациональным) может быть более эффективным методом, которое соответствует отношению двух многочленов.

$$R(x) = \frac{a_0 + a_1 x + \cdots + a_p x^p}{b_0 + b_1 x + \cdots + b_p x^p}, \quad p + q + 1 = n$$

Коэффициенты a_i, b_i можно найти из совокупности соотношений $R(x_i) = y_i, \ i = 1, \dots, n$, которые можно записать в виде:

$$\sum_{i=0}^{p} a_i x_i^i - f(x_i) \sum_{i=0}^{q} b_j x_i^j = 0, i = 1, \dots, n$$

Из этого процесса получаем систему из n линейных алгебраических уравнений относительно $n+1$ неизвестных. Если n нечетное и $p = q$ или n четное и $p - q = 1$, то функция $R(x)$ может быть записана явно. Для этого необходимо вычислить обратные разделенные разности в соответствии с определенными условиями

$$f^{-}(x_k; x_l) = \frac{x_k - x_l}{f(x_k) - f(x_l)}$$

и рекуррентным соотношением

$$f^{-}(x_k; \dots; x_l) = \frac{x_l - x_k}{f^{-}(x_{k+1}; \dots; x_l) - f^{-}(x_k; \dots; x_{l-1})}$$

после чего интерполирующая рациональная функция записывается в виде цепной дроби

$$f^{-}(x_k; \dots; x_n) = \frac{x - x_1}{f^{-}(x_1; x_2) + \frac{x - x_2}{f^{-}(x_1; x_2; x_3) + \dots + \frac{z - z_{N-1}}{I^{-}(z_k; \dots; z_n)}}}$$

Для функции, которая имеет нерегулярное поведение, рациональное интерполирование может быть эффективным методом при условии, что узлы выбраны правильно.

2.4 Сплайновая интерполяция.

Сплайновая интерполяция — это метод интерполяции, при котором используется набор полиномов малой степени, называемых сплайнами, для аппроксимации кривой или поверхности, проходящей через заданные точки.

Для построения сплайновой интерполяции необходимо разбить исходный интервал на меньшие интервалы, называемые узлами. Затем для каждого узла строится полином, проходящий через узел и соседние узлы. Эти полиномы должны удовлетворять некоторым условиям, называемым граничными условиями, которые определяются в зависимости от конкретной задачи.

Обычно для сплайновой интерполяции используются кубические сплайны, т.е. полиномы третьей степени. При этом каждый кубический сплайн определяется следующим образом:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x_i \le x \le x_{i-1}$$

где x_i — узел интерполяции, a_i, b_i, c_i, d_i — коэффициенты полинома, которые необходимо найти.

Граничные условия могут быть разными. Например, для натурального сплайна используется условие равенства нулю второй производной на концах интервала:

$$S^{"}(x_0) = S^{"}(x_n) = 0$$

Для кубического сплайна на каждом интервале, кроме первого и последнего, также должны быть выполнены условия непрерывности первой и второй производных:

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}),$$

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}),$$

$$S_i^{"}(x_{i+1}) = S_{i+1}^{"}(x_{i+1})$$

Таким образом, сплайновая интерполяция позволяет более гладко аппроксимировать кривую или поверхность, чем полиномиальная интерполяция, и имеет более гибкие граничные условия.

2.4.1 Модифицированная интерполяция Акима.

Модифицированная интерполяция Акима — это метод интерполяции данных, который использует локальные полиномы для аппроксимации данных в окрестности каждой точки исходного набора данных.

Для построения интерполяционной кривой на основе данных $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, где x_i — координаты точек на оси x, а y_i — соответствующие значения функции на оси y, сначала вычисляются конечные разности второго порядка:

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$$

где $\Delta y_i = y_i - y_{i-1}$.

Затем для каждой точки используются локальные полиномы, которые строятся по 5 точкам с помощью формулы:

$$p(x) = c_1 + c_2(x - x_i) + c_3(x - x_i)^2 + c_4(x - x_i)^3$$

где c_1, c_2, c_3 и c_4- коэффициенты полинома, которые находятся из условий, что:

1. полином проходит через центральную точку исходного набора данных: $p(x_i) = y_i$
2. первые производные полинома совпадают с конечными разностями: $p'(x_i) = \Delta y_i$, p'(x_{i+1}) = Δy_{i+1}
3. полином является монотонным, то есть, $p'(x_i)$ и p'(x_{i+1}) имеют одинаковый знак.

Коэффициенты c_1, c_2, c_3 и c_4могут быть найдены аналитически с помощью следующих формул:

$$c_1 = y_i$$

$$c_2 = \Delta y_i$$

$$c_3 = \frac{3\Delta^2 y_i - 2\Delta^2 y_{i-1} - \Delta^2 y_{i+1}}{x_{i+1} - x_i}$$

$$c_4 = \frac{2\Delta^2 y_i + \Delta^2 y_{i-1} - 3\Delta^2 y_{i+1}}{(x_{i+1} - x_i)^2}$$

Интерполяционная кривая может быть получена путем соединения всех полиномов.

2.4.2 Кусочно-кубический полином Эрмита.

Интерполяция кусочно-кубическим полиномом Эрмита — это метод интерполяции, который использует кубические полиномы для интерполяции функции и ее производной. Для каждого интервала $[x_i,\ x_{i+1}]$ метод определяет значения функции и ее производной в четырех точках: $f(x_i), f(x_{i+1}), f'(x_i), f'(x_{i+1})$. Затем на этом интервале строится кубический полином, который проходит через эти точки и имеет те же значения производной на концах интервала, что и исходная функция.

Формулы для построения кусочно-кубического полинома Эрмита на интервале $[x_i, x_{i+1}]$ выглядят следующим образом:

$$H(x) = h_{00}(x)f(x_i) + h_{10}(x)f(x_{i+1}) + h_{01}(x)f'(x_i) + h_{11}(x)f'(x_{i+1}),$$

Где $f(x_i), f(x_{i+1}), f'(x_i), f'(x_{i+1})$ — известные значения функции и ее производной в точках x_i и x_{i+1}, а $h_{00}(x), h_{10}(x), h_{01}(x)$ и $h_{11}(x)$ - кубические базисные функции Эрмита:

$$h_{00}(x) = 2\left(\frac{x - x_i}{h}\right)^3 - 3\left(\frac{x - x_i}{h}\right)^2 + 1$$

$$h_{10}(x) = -2\left(\frac{x - x_{i+1}}{h}\right)^3 + 3\left(\frac{x - x_{i+1}}{h}\right)^2$$

$$h_{01}(x) = \left(\frac{x - x_i}{h}\right)^3 - 2\left(\frac{x - x_i}{h}\right)^2 + \frac{x - x_{i+1}}{h}$$

$$h_{11}(x) = -2\left(\frac{x - x_{i+1}}{h}\right)^3 + \left(\frac{x - x_{i+1}}{h}\right)^2$$

где $h = x_{i+1} - x_i$ — длина интервала.

2.4.3 Квадратичные B-сплайны.

Интерполяция квадратичными B-сплайнами представляет собой метод интерполяции, основанный на построении кусочно-квадратичного сплайна с использованием функций Б-сплайнов второго порядка.

Формула квадратичного B-сплайна имеет вид:

$$S(x) = \begin{cases} \dfrac{(x - t_i)^2}{2(t_{i+1} - t_i)(t_{i+2} - t_i)}, & t_i \le x \le t_{i+1} \\ \dfrac{(x - t_{i+1} + 1)(x - t_{i-1})}{(t_{i+1} - t_i)(t_{i+1} - t_{i-1})}, & t_{i+1} \le x \le t_{i+2} \\ \dfrac{(t_{i+2} - x)^2}{2(t_{i+2} - t_{i+1})(t_{i+2} - t_i)}, & t_{i+2} \le x \le t_{i+3} \end{cases}$$

где t_i — узлы интерполяции, $i = 0,1,2, \dots, n + 1$.

Для интерполяции значения функции в узлах интерполяции используются условия:

$$S(t_i) = f(t_i), \qquad S'(t_i) = f'(t_i)$$

где $f(t_i)$ и $f'(t_i)$ — значения функции и ее производной в узлах интерполяции.

Интерполяционная кривая представляет собой кусочно-квадратичный сплайн, который проходит через все узлы интерполяции и имеет непрерывные первую производную.

2.4.4 Кубические B-сплайны.

Интерполяция кубическими B-сплайнами — это метод интерполяции, который использует кубические функции-сплайны для аппроксимации кусочно-гладкой функции на заданном интервале. Кубические B-сплайны имеют преимущества перед другими методами интерполяции, такими как полиномиальная интерполяция, потому что они более устойчивы к выбросам и шумам.

Кубический B-сплайн на каждом отрезке $[x_i, x_{i+1}]$ представляется кубическим полиномом $S_i(x)$, который является решением системы линейных уравнений, удовлетворяющих следующим условиям:

1. $S_i(x_i) = f(x_i)$ и $S_i(x_{i+1}) = f(x_{i+1})$ — сплайн проходит через соседние узлы.
2. $S_{i-1}(x_i) = S_i(x_i)$ и $S'_{i-1}(x_i) = S'_i(x_i)$ — первая и вторая производные сопрягаются на точке пересечения.
3. $S''_{i-1}(x_n) = S''_n(x_n) = 0$ — краевые условия.

Кубический сплайн $S_i(x)$ на отрезке $[x_i, x_{i+1}]$ может быть записан в следующей форме:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

где a_i, b_i, c_i и d_i - коэффициенты, которые могут быть найдены из уравнений выше.

2.5 Выбор критериев для сравнения.

После того, как были рассмотрены основные методы интерполяции, необходимо проанализировать их достоинства и недостатки. Для этого нужно опредлится с критериями, которые будут оказывать наибольшее влияние на программный продукт. Среди множества критериев выделяются следующие:

1. Сложность алгоритма – характеристика, которая указывает на то, сколько времени, либо какой объем памяти потребуется для выполнения.
2. Наличие готовой реализации позволит ускорить разработку продукта, а также снизить количество ошибок и некачественного кода.
3. Гибкость настройки отражает способность метода настраиваться под конкретную задачу, что благоприятно сказывается на точности вычисления, но увеличивает количество аспектов на которые стоит обратить внимание при разработке.

2.6 Сравнение методов по критериям.

После выбора критериев было проведено сравнение, результаты представлены в таблице *:

Метод/ Критерий	Сложность алгоритма	Реализация в библиотеках	Гибкая настройка параметров
Интерполяция Виттекера-Шеннона	$O(N)$	+	-
Кубическая интерполяция Эрмита	$O(log(N))$	+	+
Кардинально-квадратичная B-сплайновая интерполяция	$O(N)$	+	+
Кардинально кубическая интерполяция B-сплайна	$O(N)$	+	+
Барицентрическая рациональная интерполяция	$O(N)$	+	+
Модифицированная интерполяция Акима	$O(log(N))$	+	+

Таблица * - Сравнение методов интерполяции по критериям

2.7 Вывод по разделу.

Рассмотрены основные методы интерполяции, которые используются в данном приложении. Каждый метод имеет свои преимущества и ограничения, и выбор конкретного метода зависит от требований к точности и эффективности вычислений. Некоторые методы, такие как интерполяция Эрмита, могут обеспечить точность интерполяции при условии наличия дополнительной информации о функции. Другие методы, такие как сплайны, позволяют достичь гладкости и непрерывности интерполирующей функции. Все они доступны в приложении и могут быть использованы для интерполяции пространственных кривых.

РАЗРАБОТКА ИНСТРУМЕНТА ВИЗУАЛИЗАЦИИ

4.1 Требования к разрабатываемому продукту.

Опираясь на проведённый анализ существующих алгоритмов восстановления данных методами интерполяции, можно сформировать требования, которых должна придерживаться разрабатываемая программная система:

1. Продукт должен включать один или несколько алгоритмов восстановления.
2. В программной системе должны использоваться методы, реализованные в готовых математических библиотеках.
3. Программная система должна иметь возможность менять условия или параметры используемых алгоритмов интерполяции в зависимости от ситуации.

Таким образом разрабатываемая система, производящая моделирование и визуализацию пространственной кривой по восстановленным данным, удовлетворит всем требованиям. Это позволит пользователям в реальном времени отслеживать ориентацию гибкой буксируемой антенны в пространстве во время буксировки, даже если некоторые координаты с датчиков не будут получены. Использование быстрых и реализованных алгоритмов снизит вероятность ошибок при работе программы.

4.2 Используемые технологии.

Для решения поставленной задачи по разработке архитектуры приложения, удовлетворяющего описанным выше требованиям, были выбраны следующие технологии и цели их использования:

- C++ - лучший вариант по производительности кода в высоконагруженных приложениях. На этом языке будет написана вся логика программы.

- boost - математическая библиотека на языке C++. Из неё будут задействованы готовый алгоритмы интерполяции.
- Qt - фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++. Он позволит быстро создать графический интерфейс продукта и смоделировать пространственную кривую.

Программная реализация подразумевает монолитную архитектуру. На вход приложению в реальном времени поступают потоковые данные с датчиков гибкой антенны. Координаты с неисправных датчиков, либо незахваченные потоком, проходят процедуру восстановления выбранным методом интерполяции. Алгоритм позволит сформировать модель кривой, при этом гарантируя её прохождение через имеющиеся точки. Затем на основе восстановленных данных моделируется пространственная ориентация антенны и результат выводится на экран. Обработка всех графических элементов производится в среде Qt.

4.3 Описание доступных методов интерполяции.

// Нужен ли вообще этот пункт?

4.4 Общая архитектура.

При проектировании и разработке приложений часто используют различные архитектурные паттерны, такие как MVC (Model-View-Controller). Этот паттерн помогает разделить приложение на три компонента: модель, представление и контроллер. Каждый компонент выполняет свою задачу, что облегчает сопровождение и развитие приложения. На рис. * представлена обобщенная архитектура решения:

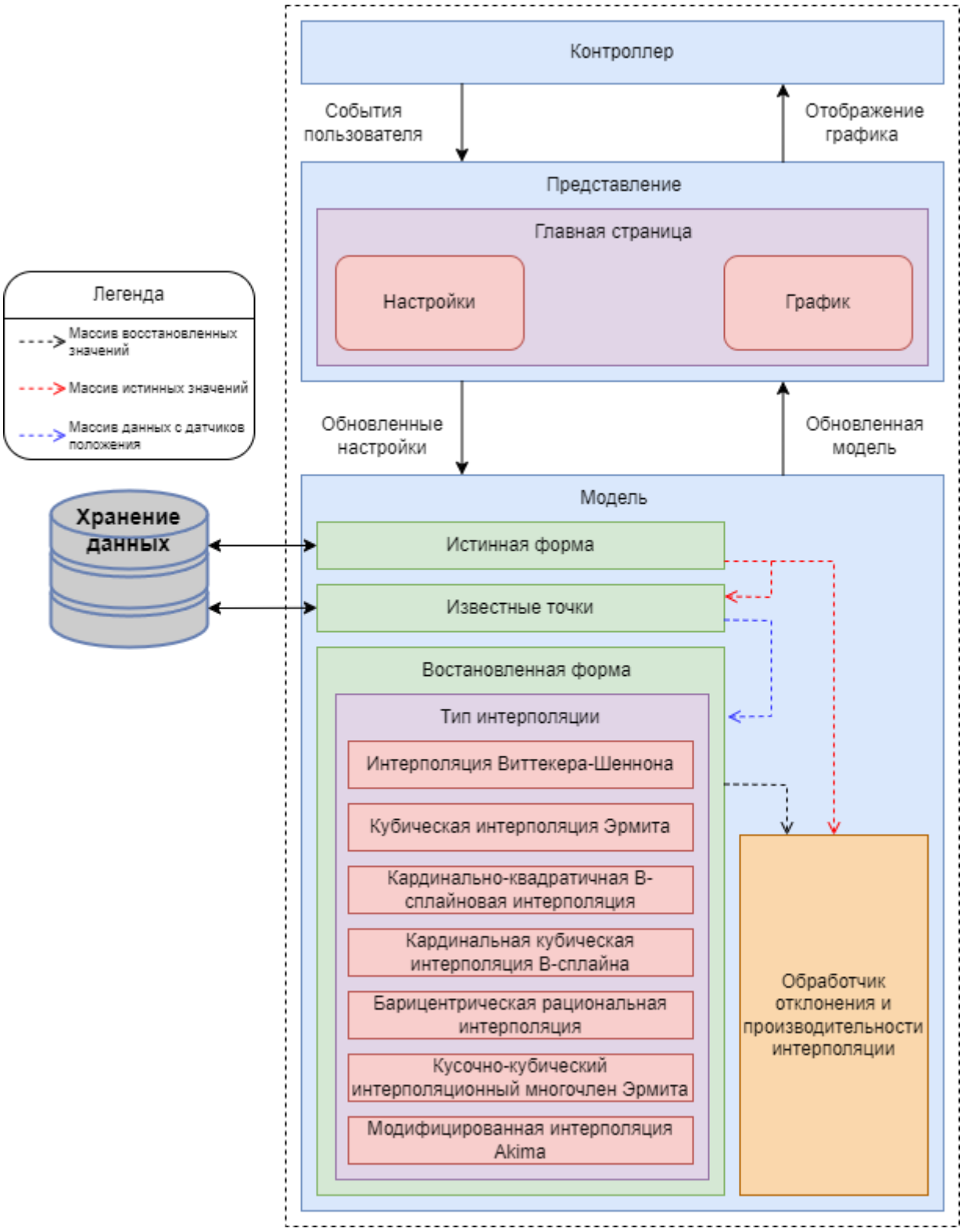

Рисунок * - Обобщенная архитектура решения

В этом разделе мы рассмотрим общую архитектуру приложения, использующую паттерн MVC, и каждый из компонентов более подробно.

4.4.1 Контроллер.

Уровень контроллера в приложении отвечает за обработку пользовательского ввода и управление взаимодействием между компонентами приложения. Он является посредником между пользователем и моделью, а также управляет представлением, которое отображает данные, полученные от модели.

В данном случае, контроллер обрабатывает различные пользовательские действия, такие как добавление, удаление или изменение точек кривой, выбор определенного вида кривой для визуализации, изменение масштаба и поворота камеры в 3D пространстве и т.д. Контроллер также должен обеспечить правильное взаимодействие с моделью, так что любые изменения, сделанные пользователем, отображены в представлении.

4.4.2 Представление.

Уровень представления отвечает за отображение данных пользователю и обработку пользовательского ввода. В данном случае, главная страница приложения включает в себя два основных элемента:

1. Панель настройки, которая задает базовую модель, известные точки и метод восстановления, а также управляет визуальной составляющей отображающегося графика.
2. Поле визуализации, которое отображает график, построенный на основе введенных пользователем параметров. В этом поле пользователь может видеть исходную модель, известные точки, а также восстановленную кривую и взаимодействовать с ней (вращать, приближать/отдалять).

На уровне представления также реализована валидация пользовательского ввода. Более подробный разбор пользовательского интерфейса будет в соответствующем разделе.

4.4.3 Модель.

Уровень модели отвечает за хранение данных и бизнес-логику приложения. Он является независимым от пользовательского интерфейса и контроллера.

Модель представляет собой совокупность информации об истинной форме модели, наборе известных точек, а также восстановленной форме. Кроме того, модель включает блок обработки отклонений и производительности, который производит замеры времени и отвечает за сравнение восстановленной формы с исходной моделью.

Восстановленная форма включает в себя доступ к математическим библиотекам для выполнения процесса интерполяции и построения аппроксимирующих функций на основе имеющихся данных.

Для удобства пользователей приложения, предусмотрена возможность сохранения и загрузки информации. Все установленные данные будут сохранены на локальной машине, и загружены с новым запуском.

4.5 Интерфейс пользователя.

Интерфейс пользователя — это один из ключевых компонентов любого приложения. Он позволяет пользователям взаимодействовать с приложением, управлять его функциями и просматривать результаты работы. Хорошо спроектированный интерфейс должен быть интуитивно понятным, удобным и привлекательным для пользователя. На рисунке * представлено главное окно приложения:

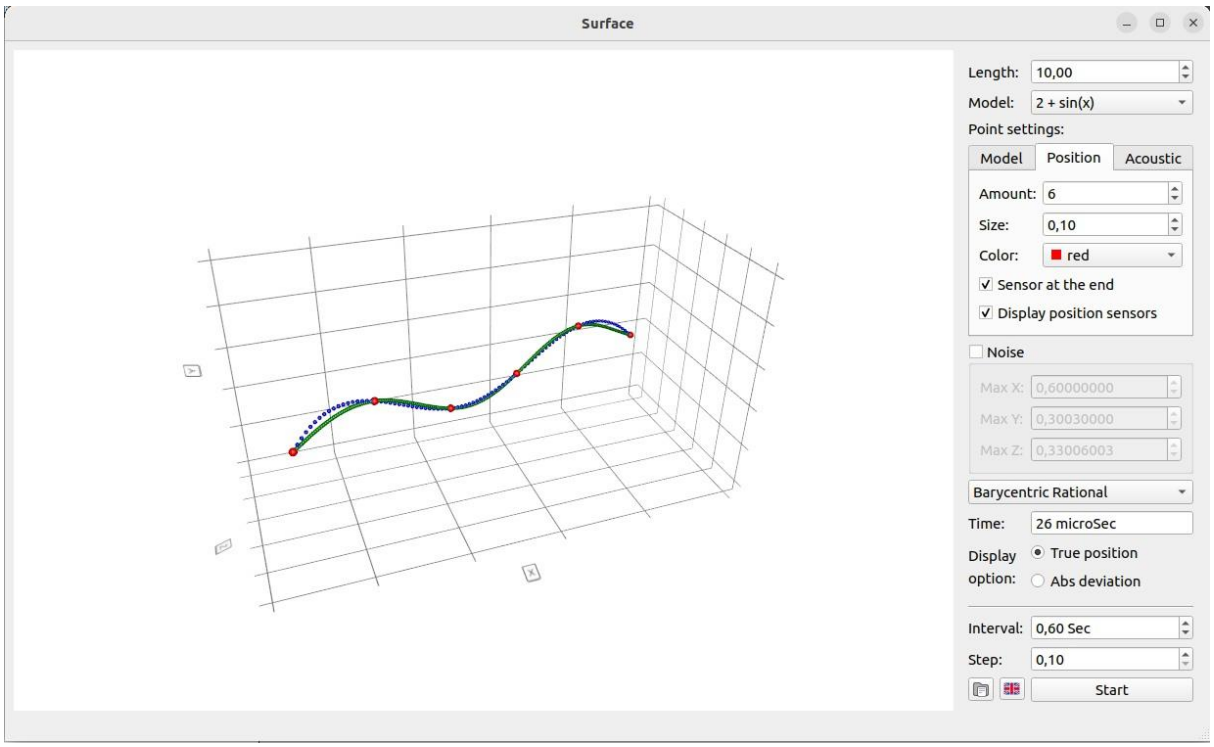

Рисунок * - Главное окно приложения

Основную часть всего главного окна занимает поле визуализации смоделированного графика. В данном окне пользователь имеет возможность выбрать наиболее подходящий для себя угол обзора, соответствующее приближение и в общих чертах оценить отклонение выбранного метода восстановления от базовой модели.

Меню настроек предоставляет богатый функционал по настройке исходных данных. Пользователь может влиять на следующие аспекты:

- Длину истинной модели
- Функцию истинной модели
- Количество, цвет и размер всех отображаемых точек
- Наличие отклонений и их максимальные значения по всем осям
- Тип метода восстановления
- Отображаемую информацию при выборе конкретной точки
- Интервал с которым будет обновляться эмуляция
- Шаг на который по графику будет продвигаться эмуляция
- Осуществлять запуск или остановку эмуляции

Настройка визуализаций всех категории точек производится в соответствующих меню, представленных на рисунках *, *, * для истинной модели, известных точек и восстановленной формы соответственно:

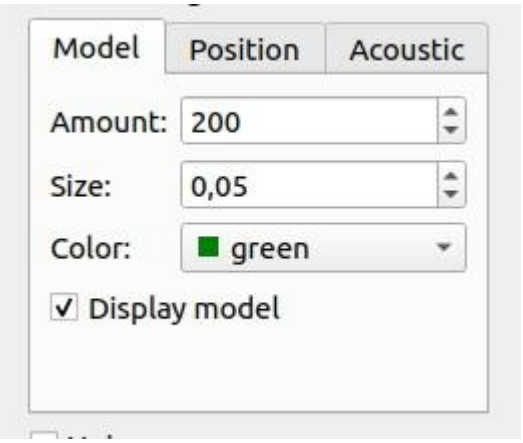

Рисунок * - Настройка визуализации для истинной формы

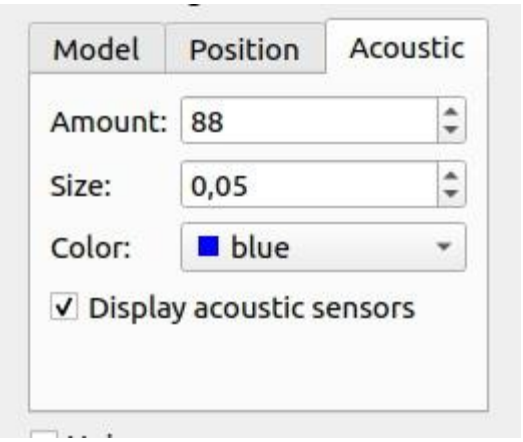

Рисунок * - Настройка визуализации для восстановленных данных

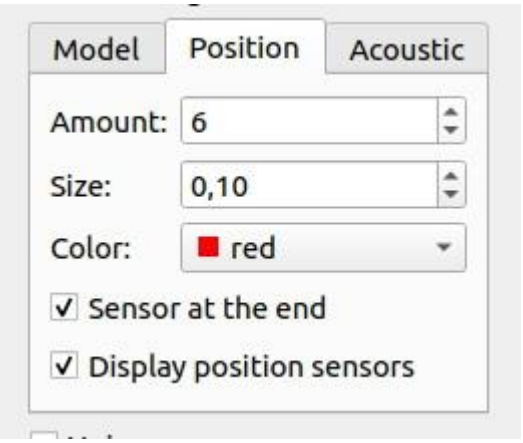

Рисунок * - Настройка для известных точек

Между настройкой точек для истинной формы и восстановленных данных различий нет, они идентичны, на всю длину выбирается количество отображаемых точек, размер, цвет и флаг видимости. Отличие же настроек известных точек состоит в наличии специального флага, который изменяет шаг таким образом, чтобы последняя известная точка находилась строго на конце модели.

Для симуляции отклонений используется меню, представленное на рисунке *:

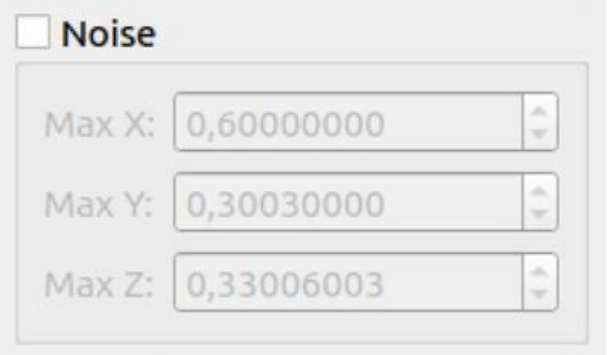

Рисунок * - Меню настройки отклонений

Наличие отклонений, а также их максимальное значение, определяет пользователь. Присутствующая валидация не позволяет устанавливать отклонения превышающее определенного значения, которое нарушит явный порядок точек.

Эмуляция движения модели осуществляется по нажатию на кнопку, изображенную на рисунке *:

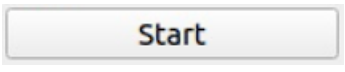

Рисунок * - Кнопка начала эмуляции

Таймер с заданным интервалом начинает двигаться по модели, шаг и интервал задается с помощью меню, изображенном на рисунке *:

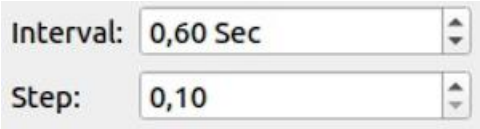

Рисунок * - Меню интервала и шага эмуляции

Если включено отклонение, то на каждый шаг известные точки будут принимать новое отклонение, не превышающее заданного.

Для оценки метода восстановления пользователю предоставлен следующий функционал:

- Окно с временными затратами выбранного метода
- Панель с отклонениями для конкретной точки

На рисунке * представлено окно, которое оценивает временные затраты, время указывается в микросекундах:

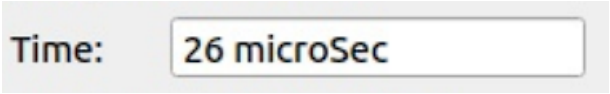

Рисунок * - Окно временных затрат выбранного метода восстановления

При выборе конкретной точки на визуализированном графике отображаются её координаты, пример представлен на рисунке *:

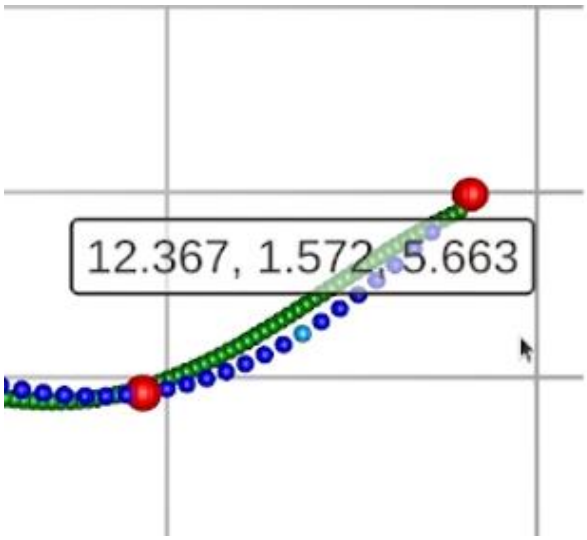

Рисунок * - Визуализация координат на графике

Если изменить тип отображения в соответствующем меню, представленном на рисунке *, то отображаемая информация будет указывать на модуль отклонения, пример отображаемого отклонения представлен на рисунке *:

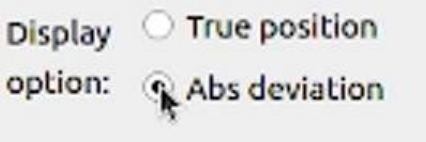

Рисунок * - Меню выбора типа отображающейся информации

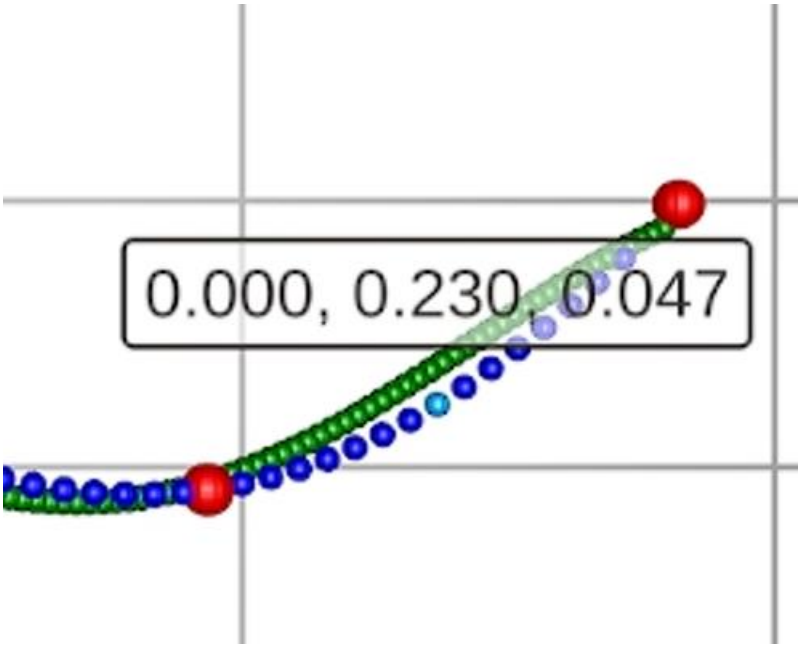

Рисунок * - Отклонение восстановленных данных от истинной формы

Для облегчения использования программы предоставлена возможность загрузки исходной модели и известных точек из файла формата .txt, кнопка выбора файла изображена на рисунке *:

Рисунок * - Кнопка выбора файла

Программа поддерживает Русский и Английский языки, смена происходит без необходимости перезагрузки программы, кнопка смены языка представлена на рисунке *, а переведенное главное окно на рисунке *:

Рисунок * - Кнопка смены языка

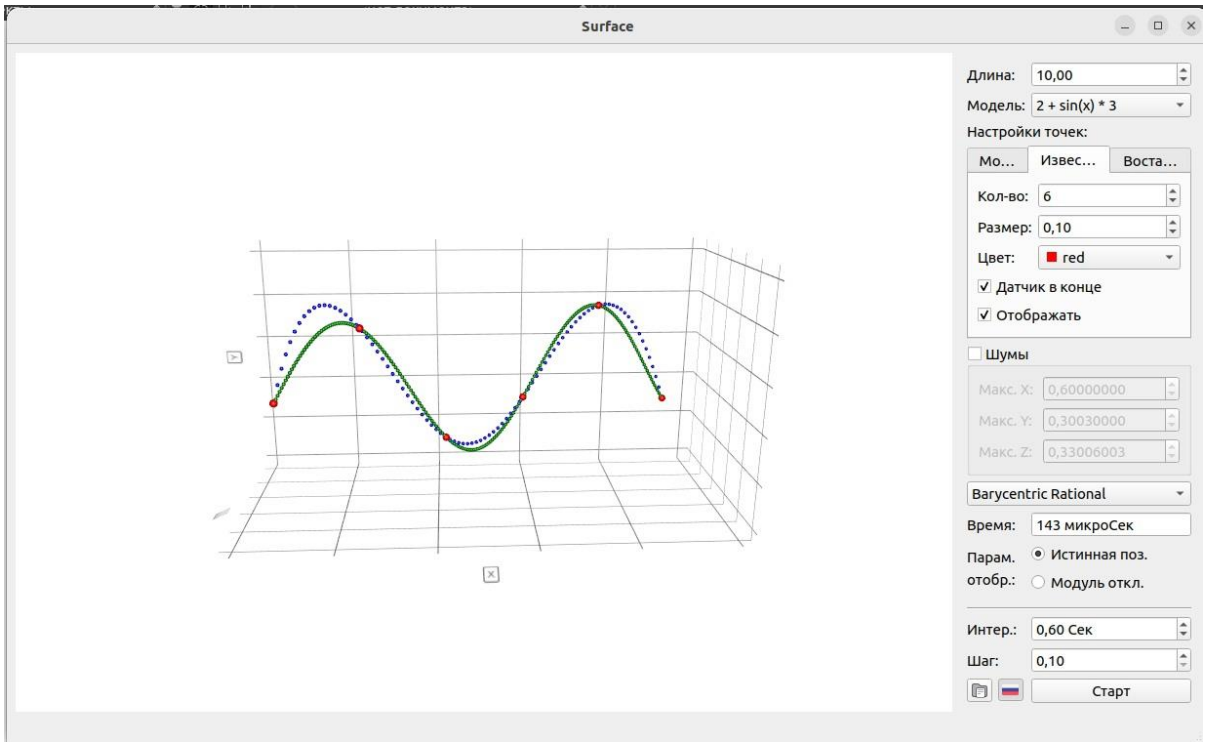

Рисунок * - Переведенное главное окно

4.5.1 Сценарии использования.

Использование данного решения заключается в создании графического отображения и анализе полученных результатов, в таблице * представлены сценарии использования:

Краткое описание	Выбор модели, метода обработки, количества датчиков позиции и настройка отображения.
Действующее лицо	Пользователь

Предусловие	Программа запущена, фокус на главном окне
Основной сценарии	
Пользователь	**Система**
Выбирает модель, длину и параметры отображения	Обновляет поле с визуализированным графиком
Задает количество известных точек и параметры отображения	Обновляет поле с визуализированным графиком
Выбирает тип интерполяции и параметры отображения	Обновляет поле с визуализированным графиком
Задает время обновления и шаг	
Выставляет галочку в чекбоксе “Помехи”	Поля X, Y, Z в разделе “Помехи” становятся активны
Задет максимальное отклонение датчиков позиции по осям X, Y, Z	
Нажимает кнопку “Start”	Поле с графиком начинает обновляться согласно заданному интервалу и шагу. Датчики позиции на каждый шаг принимают отклонение, не превышающее заданного
Альтернативный сценарии №1 – Загрузка из файла	
Пользователь	**Система**
Нажимает на кнопку “Читать из файла”	Предоставляет системное окно выбора файла
Выбирает нужный файл	Считывает файл, устанавливает модель, метод обработки, количество датчиков позиции, помехи, шаг, интервал и настройки отображения
Нажимает кнопку “Start”	Поле с графиком начинает обновляться согласно заданному интервалу и шагу. Датчики позиции на каждый шаг принимают отклонение, не превышающее заданного

Таблица * - Сценарии использования

4.5.2 Графическая схема.

// Макет интерфейса с графом переходов

4.6 Вывод по разделу.

Для разработки программного модуля был выбран фреймворк Qt, который был использован для создания интерфейсного приложения. Исходный код модуля доступен в открытом репозитории на платформе GitHub.

ПРОВЕДЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

5.1

5.2 Основные характеристики ПК.

- Тактовая частота центрального процессора (CPU): 12th Gen Intel(R) Core(TM) i5-12400F 2.50 GHz;
- Количество ядер центрального процессора: 6 ядер;
- Объем оперативной памяти компьютера (ОЗУ): 32 ГБ
- Объем памяти видеокарты: 2 ГБ;
- Размер свободного пространства на жестком диске или SSD: 256 ГБ;
- Версия Windows: Windows 10 Pro 64-bit

На рисунке * представлены основные характеристики компьютера:

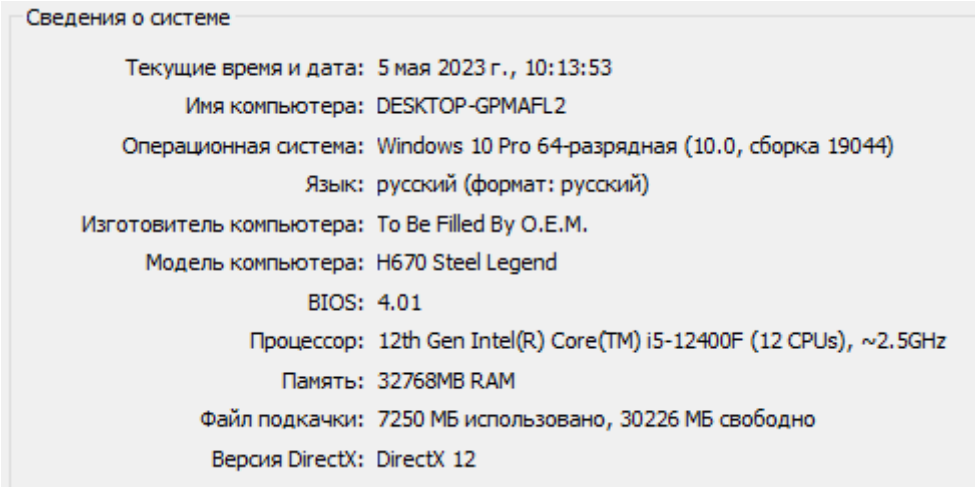

Рисунок * - Основные характеристики компьютера

5.3 Тестирование программы.

НОРМАТИВНО-ПРАВОВОЕ РЕГУЛИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

6.1 Общие положения

Под интеллектуальной собственностью понимают особый вид гражданских прав (исключительное право) в отношении результатов интеллектуальной деятельности, таких как изобретения, промышленные образцы(дизайн), компьютерные программы, другие произведения науки, произведения литературы, искусства, которые принято называть объектами интеллектуальной собственности, а также различных средств индивидуализации производителя товаров и услуг, таких как товарные знаки, знаки обслуживания, фирменные наименования и др. [30, ст. 1225]. Основным содержанием таких прав является монополия их владельца на использование этих объектов, включая право запретить или разрешить их использование другим, а также право переуступить другому лицу эти правомочия или отказаться от них вовсе.

Согласно определению интеллектуальной собственности, принятому в российском законодательстве, а также на основании определения Стокгольмской конференции от 14 июля 1967 г., программы для ЭВМ (компьютерные программы) и базы данных относятся к объектам интеллектуальной собственности. Программам для ЭВМ и базам данных предоставляется охрана нормами авторского права как литературным произведениям в соответствии с Бернской конвенцией, причем программы для ЭВМ охраняются как литературные произведения, а базы данных — как сборники.

В Российской Федерации вопросы предоставления правовой охраны программам для ЭВМ и базам данных регулируются Гражданским кодексом РФ, Часть 4 (ГК РФ Ч. 4).

Под программой для ЭВМ понимается «. . . *представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определенного*

результата». Кроме того, в понятие программы для ЭВМ входят «. . . *подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения*» [30, ст. 1261].

С точки зрения программистов и пользователей программа для ЭВМ представляет собой детализацию алгоритма решения какой-либо задачи и выражена в форме определенной последовательности предписаний, обеспечивающих выполнение компьютером преобразования исходных данных в искомый результат. Можно выделить следующие объективные формы представления программы для ЭВМ:

- исходная программа (или исходный текст) — последовательность предписаний на алгоритмическом (понятном человеку) языке высокого уровня, предназначенных для автоматизированного перевода этих предписаний в последовательность команд в объектном коде;
- рабочая программа (или объектный код) — последовательность машинных команд, т. е. команд, представленных на языке, понятном ЭВМ;
- программа, временно введенная в память ЭВМ, — совокупность физических состояний элементов памяти запоминающего устройства ЭВМ(ОЗУ), сохраняющихся до прекращения подачи электропитания к ЭВМ;
- программа, постоянно хранимая в памяти ЭВМ, — представленная на языке машины команда (или серия команд), выполненная в виде физических особенностей участка интегральной схемы, сохраняющихся независимо от подачи электропитания.

Исходная и рабочая программы, как правило, представляются в виде записи на том или ином языке, выполненной на бумаге или машиночитаемом носителе данных: магнитном или оптическом диске, магнитной ленте и т. п.

Предоставляемая законодательством правовая охрана распространяется «. . . *на все виды программ для ЭВМ (в том числе на операционные системы и*

программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код . . .» [30, ст. 1261]. Преобразование исходного текста программы для ЭВМ в объектный (машинный) код не меняет сущности данной программы как произведения. Значит, если охраняется исходный текст программы, то охране подлежит и соответствующий ей объектный код. Обратное тоже справедливо. Правовая охрана программ для ЭВМ распространяется только в отношении формы их выражения и «*. . . не распространяется на идеи, концепции, принципы, методы, процессы, системы, способы, решения технических, организационных или иных задач, открытия, факты, языки программирования*» [30, ст. 1259, п. 5].

В ходе выполнения данного дипломного проекта создана программа для ЭВМ «*Инструмент моделирования и визуализации формы пространственной кривой, восстановленной по данным, изменяющимся в реальном времени*», которая является объектом интеллектуальной собственности и которой может быть предоставлена правовая охрана в рамках авторского права. Автором является Попов Дмитрий Сергеевич. Правообладателем является СПбГЭТУ “ЛЭТИ”, т. к. программа создана при выполнении дипломной работы. Далее приведен комплект документов, для проведения государственной регистрации объекта.

6.2 Комплект основных документов для официальной регистрации программы

В комплект документов для официальной регистрации программы для ЭВМ входит:

1. заявление на официальную регистрацию программы для ЭВМ;
2. титульный лист к депонируемым материалам;
3. состав регистрируемого объекта;
4. реферат к программе для ЭВМ;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Boost C++ libraries [Электронный ресурс].
   URL: https://www.boost.org/doc/libs/1_81_0/libs/math/doc/html/index.html
   (дата обращения 05.05.2023)
2. Qt [Электронный ресурс].
   URL: https://www.qt.io/
   (дата обращения 05.05.2023)
3. М.Н. Магомедов, М.В. Чигирь. Нормативно-правовое регулирование результатов интеллектуальной деятельности: учебно-методическое пособие по выполнению дополнительного раздела выпускных квалификационных работ бакалавров СПб.: Изд-во СПбГЭТУ “ЛЭТИ”, 2019. 20 с
4. Гражданский кодекс Российской Федерации. (Часть четвертая) от 18.12.2006, № 230-ФЗ
5. Исходный код разработанного приложения.
   URL: https://github.com/DimonPopov/PopovDS_9304_GraduateWork