МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности

Направление 02.03.01 Математика и компьютерные науки

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Алгоритм представление пространственных кривых B-сплайн»

Обучающийся: _____ Шклярова Ксения Алексеевна

Руководитель: _____ Курочкин Михаил Александрович

«_____»_____20___г.

Санкт-Петербург, 2025

Содержание

Введение **3**

1 Постановка задачи **4**

2 Алгоритм представления пространственных кривых B-сплайн **5**

2.1 Постановка задачи 5

2.2 Степень B-сплайна 5

2.3 Общие свойства B-сплайновых кривых 5

2.4 Матричная форма кубического B-сплайна 6

2.5 Вектор параметров 6

2.5.1 Пример вычисления 6

2.6 Рекурсивное вычисление базисных функций (алгоритм де Бура–Кокса) 7

2.6.1 Пример вычисления для $p = 3$ 7

2.6.2 Запись в матричной форме 7

2.7 Геометрический вектор контрольных точек 8

2.8 Условия непрерывности между сегментами 9

2.9 Сложность алгоритма 9

2.10 Пошаговый алгоритм построения B-сплайна 9

3 Описание реализации построения B-сплайна **12**

3.1 Генерация контрольных точек (`generate_control_points`) 12

3.2 Функция построения B-сплайна матричным методом (matrix_bspline) 13

3.3 Функция построения B-сплайна при помощи библиотечных функций 15

4 Результаты работы **17**

5 Сравнение результатов работы алгоритмов **18**

Заключение **20**

Введение

В компьютерной графике, геометрическом моделировании и системах автоматизированного проектирования (CAD) важную роль играет задача построения и представления пространственных кривых. Для этих целей широко используются B-сплайны. Они позволяют эффективно аппроксимировать сложные кривые, обеспечивая локальный контроль формы, гладкость и вычислительную устойчивость, что делает их незаменимыми в задачах траекторного планирования, анимации, обработке трехмерных данных и инженерном анализе.

Алгоритмы компьютерной графики представляют собой набор математических и программных методов, используемых для создания, обработки и визуализации изображений на компьютере.

B-сплайн — это сплайн, представленный в виде линейной комбинации базисных функций (B-функций). В отличие от кривых Безье, B-сплайны позволяют локально управлять формой кривой без глобального влияния контрольных точек. Понятие B-сплайновой кривой естественным образом обобщает идею параметрического представления кривых, сочетая гибкость управления и математическую строгость.

1 Постановка задачи

В данной лабораторной работе необходимо:

1. Изучить алгоритм представления пространственных кривых B-сплайн;
2. Программно реализовать данный алгоритм;
3. Сравнить собственную реализацию с библиотечной.

2 Алгоритм представления пространственных кривых B-сплайн

2.1 Постановка задачи

Дано:

- Набор контрольных точек в трёхмерном пространстве: $S = \{P_0, P_1, \ldots, P_n\}, \quad P_i = (x_i, y_i, z_i)$

- Степень B-сплайна: p

Требуется:

- Имея заданный набор точек, построить параметрическую кривую $C(t)$ на основе B-сплайна, обеспечивающую наилучшее приближение к этим точкам.

2.2 Степень B-сплайна

Степень B-сплайна p определяет уровень гладкости кривой:

– при $p = 1$ кривая представляет собой ломаную линию,

– при $p = 2$ состоит из параболических сегментов и обеспечивает C^1-гладкость,

– при $p = 3$ получается кубическая кривая с естественной C^2-гладкостью.

Увеличение степени B-сплайна повышает гладкость кривой, но также увеличивает вычислительную сложность.

В данной работе был реализован алгоритм для построения кубического B-сплайна. Кубические B-сплайны (степени 3) представляют собой оптимальный компромисс между гладкостью и вычислительной эффективностью. В отличие от линейных (C^0) и квадратичных (C^1) сплайнов, они обеспечивают достаточную гладкость и гибкость для аппроксимации большинства реальных форм, не подвержены осцилляциям и не требуют значительных вычислительных ресурсов, в отличие от сплайнов более высоких степеней.

2.3 Общие свойства B-сплайновых кривых

B-сплайны обладают следующими важными свойствами:

- **Локальность:** изменение одной точки влияет только на ограниченное число соседних сегментов.

- **Гладкость:** при равномерном узловом векторе достигается C^{p-1}-непрерывность.

- **Вычислительная устойчивость:** численно стабильная форма.

- **Разбиение единицы:** $\sum N_{i,p}(u) = 1$, что гарантирует выпуклую комбинацию контрольных точек, $N_{i,p}(u)$ — базисные функции B-сплайна степени p.

- **Выпуклая оболочка:** кривая лежит внутри выпуклой оболочки контрольных точек.

2.4 Матричная форма кубического B-сплайна

Для $p = 3$ общая формула может быть выражена в матричной форме:

$$C(t) = T \cdot M_s \cdot G$$

Где:

- $T = [t^3\ t^2\ t\ 1]$ — вектор параметров,

- $M_s = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$ — матрица базисных функций,

- $G = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$ — геометрический вектор контрольных точек текущего сегмента.

Необходимо отдельно вычислять $x(t), y(t), z(t)$, используя соответствующие координаты точек.

2.5 Вектор параметров

Вектор параметров $T = [t^3; t^2; t; 1]$ — это строка, содержащая степени параметра t (или u, если выполнена нормировка), которые используются для представления кубического полинома в матричной форме.

Он кодирует параметрическое представление кривой на текущем сегменте $t \in [t_i, t_{i+1})$.

Для равномерного B-сплайна (с равномерным узловым вектором):

Параметр t нормируется на текущий сегмент:

$$u = \frac{t - t_i}{t_{i+1} - t_i} \in [0, 1), \quad T = [u^3; u^2; u; 1].$$

Это упрощает вычисления, так как матрица M_s становится универсальной для всех сегментов.

2.5.1 Пример вычисления

Пусть $t \in [2, 3)$, $u = t - 2 \in [0, 1)$. Тогда:

$$T = [u^3; u^2; u; 1], \quad C(u) = T \cdot M_s \cdot G.$$

Для точки $u = 0.5$:

$$T = [0.125; 0.25; 0.5; 1].$$

Умножение на M_s и G даст координаты $C(0.5)$.

2.6 Рекурсивное вычисление базисных функций (алгоритм де Бура–Кокса)

Базисные функции строятся рекуррентно:

Для $p = 0$:

$$N_{i,0}(u) = \begin{cases} 1, & u_i \le u < u_{i+1} \\ 0, & \text{иначе} \end{cases}$$

Для $p > 0$:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

Если знаменатель равен нулю, соответствующее слагаемое принимается за 0.

2.6.1 Пример вычисления для $p = 3$

Для сегмента $t \in [i, i+1)$ кубический B-сплайн зависит от четырёх функций:

- $N_{i-3,3}(t)$,
- $N_{i-2,3}(t)$,
- $N_{i-1,3}(t)$,
- $N_{i,3}(t)$.

После раскрытия рекурсии (как в вашем примере для $N_{1,3}(t)$) получаем:

$$\begin{cases} N_{i-3,3}(t) = \dfrac{(t-i+3)^3}{6}, & i-3 \le t < i-2, \\ N_{i-2,3}(t) = \dfrac{-3(t-i+2)^3 + 3(t-i+2)^2 + 3(t-i+2) + 1}{6}, & i-2 \le t < i-1, \\ N_{i-1,3}(t) = \dfrac{3(t-i+1)^3 - 6(t-i+1)^2 + 4}{6}, & i-1 \le t < i, \\ N_{i,3}(t) = \dfrac{-(t-i)^3 + 3(t-i)^2 - 3(t-i) + 1}{6}, & i \le t < i+1. \end{cases}$$

2.6.2 Запись в матричной форме

Для сегмента $t \in [0, 1)$ (нормировка параметра $u = t - i$):

1. Выразим каждую базисную функцию через u:

$$N_{i-3,3}(u) = \frac{(1-u)^3}{6},$$
$$N_{i-2,3}(u) = \frac{3u^3 - 6u^2 + 4}{6},$$
$$N_{i-1,3}(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6},$$
$$N_{i,3}(u) = \frac{u^3}{6}.$$

2. Запишем сплайн как линейную комбинацию контрольных точек $P_{i-3}, P_{i-2}, P_{i-1}, P_i$:

$$C(u) = N_{i-3,3}(u)P_{i-3} + N_{i-2,3}(u)P_{i-2} + N_{i-1,3}(u)P_{i-1} + N_{i,3}(u)P_i.$$

3. Представим это в виде матричного умножения:

$$C(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}.$$

Здесь:

- Вектор $\mathbf{U} = [u^3, u^2, u, 1]$ отвечает за параметризацию.
- Матрица $\mathbf{M}_B$ содержит коэффициенты полиномов базисных функций.
- Вектор $\mathbf{G}$ — контрольные точки.

2.7 Геометрический вектор контрольных точек

Геометрический вектор $\mathbf{G}$ — это столбец, содержащий координаты контрольных точек, влияющих на текущий сегмент B-сплайна. Для кубического B-сплайна (степень 3) он включает 4 последовательные контрольные точки:

$$\mathbf{G} = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix},$$

где $P_{i-1}, P_i, P_{i+1}, P_{i+2}$ — точки в 2D или 3D пространстве.

Правила построения:

1. **Зависимость от степени сплайна (p).**

   Для степени p требуется $p + 1$ контрольная точка на сегмент. В кубическом случае ($p = 3$) — 4 точки.

2. **Скользящее окно.**

   Для сегмента $t \in [t_i, t_{i+1}]$ вектор $\mathbf{G}$ включает точки с индексами от $i - 1$ до $i + 2$:

$$\mathbf{G}_i = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

   При переходе к следующему сегменту $t \in [t_{i+1}, t_{i+2}]$ окно сдвигается на одну точку:

$$\mathbf{G}_{i+1} = \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}$$

2.8 Условия непрерывности между сегментами

Рассмотрим два соседних сегмента:

- Первый: $C_i(t)$, использует $G_i = [P_{i-1}, P_i, P_{i+1}, P_{i+2}]^T$

- Второй: $C_{i+1}(t)$, использует $G_{i+1} = [P_i, P_{i+1}, P_{i+2}, P_{i+3}]^T$

Условие	Выражение
Непрерывность положения (C)	$C_i(1) = C_{i+1}(0) = \frac{P_i + 4P_{i+1} + P_{i+2}}{6}$
Первая производная (C^1)	$C_i'(1) = C_{i+1}'(0) = \frac{-P_i + P_{i+2}}{2}$
Вторая производная (C^2)	$C_i''(1) = C_{i+1}''(0) = P_i - 2P_{i+1} + P_{i+2}$

Эти условия гарантируют плавность кривой в точках соединения сегментов.

2.9 Сложность алгоритма

- **Сложность вычисления одного значения $C(t)$:** $O(1)$, если матрицы уже загружены.

- **Общая сложность построения кривой для m точек и n сегментов:** $O(m \cdot n)$

Плюсы:

- Очень высокая скорость вычисления

- Простая реализация на GPU или SIMD-архитектурах

- Четко разделены координаты $x(t), y(t), z(t)$

Минусы:

- Подходит только для $p = 3$

- Требует выбора 4 точек для каждого сегмента

2.10 Пошаговый алгоритм построения B-сплайна

Входные данные:

- Массив контрольных точек $P = \{P_0, P_1, \ldots, P_n\}$

- Количество точек на каждом сегменте m

- Геометрическая матрица G — выбирается для каждого сегмента из 4 соседних точек

- Базисная матрица M_s для кубического B-сплайна:

$$M_s = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Выход: Массив точек кривой $C(t)$

Шаги:

1. Разделить кривую на $n - 3$ сегментов, каждый зависит от 4 точек:

$$G = [P_{i-1}, P_i, P_{i+1}, P_{i+2}]$$

2. Для каждого сегмента:

   - Рассчитать $t \in [0, 1]$ с шагом $\Delta t = 1/m$

   - Для каждого t:

   - Создать вектор параметров $T = [t^3, t^2, t, 1]$

   - Вычислить координаты точки:

$$C_x(t) = T \cdot M_s \cdot G_x C_y(t) = T \cdot M_s \cdot G_y C_z(t) = T \cdot M_s \cdot G_z$$

3. Сохранить все точки $C(t)$ для построения кривой.

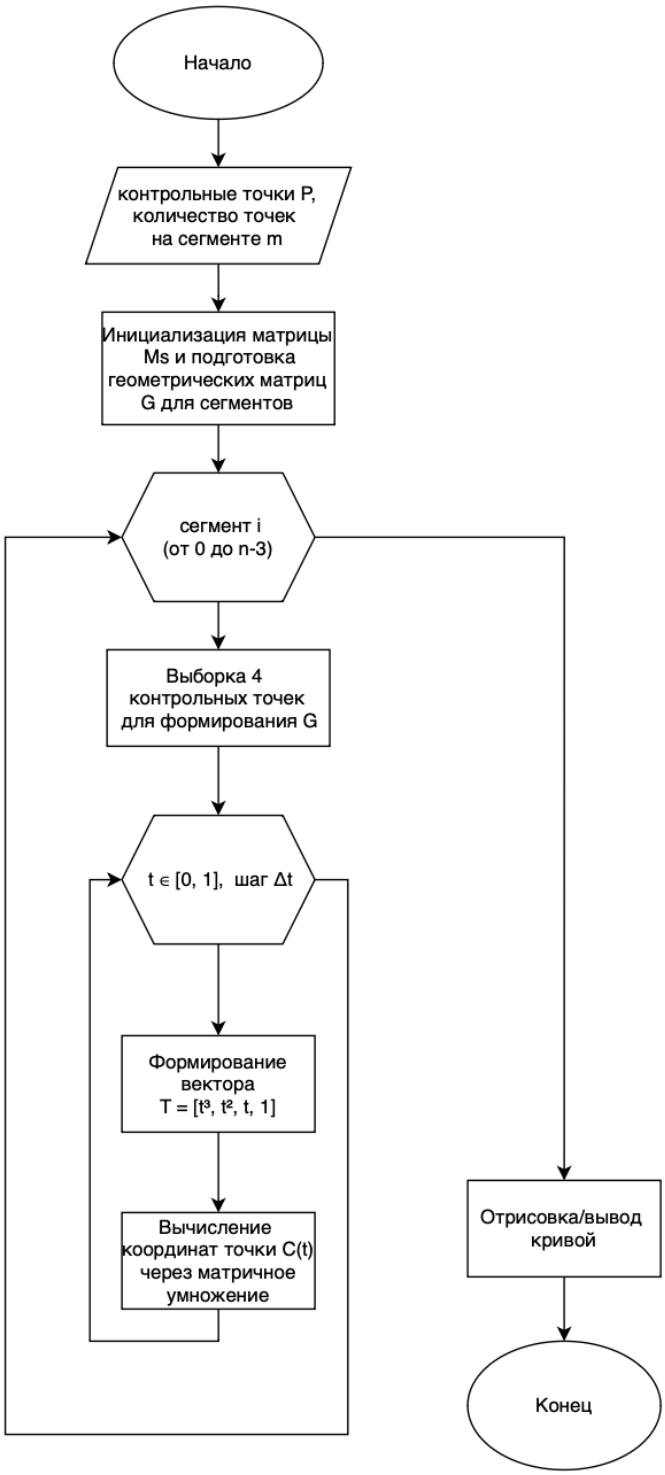


Рис. 1. Блок-схема алгоритма

3 Описание реализации построения B-сплайна

3.1 Генерация контрольных точек (`generate_control_points`)

Функция создаёт массив контрольных точек для B-сплайна в виде 3D-спирали.

Параметры:

- **num_waves** — количество «волн» (витков спирали).
- **amplitude** — амплитуда колебаний по осям Y и Z.
- **points_per_turn** - количество точек для одного витка (волны).

Логика работы:

1. **Количество точек:**

$$\text{steps} = \text{num\_waves} \times 3 + 1$$

Для каждой волны требуется минимум 3 точки (начало, максимум, спад), плюс одна точка для завершения кривой.

Например, при num_waves $= 4$ будет $4 \times 3 + 1 = 13$ точек.

2. **Формула точек:**

$$x = i \times 1.5 \quad \text{— равномерное движение по оси X.}$$
$$y = \text{amplitude} \times \sin\left(\frac{i \times \pi}{1.5}\right) \quad \text{— синусоидальное колебание по Y.}$$
$$z = \text{amplitude} \times \cos\left(\frac{i \times \pi}{1.5}\right) \quad \text{— косинусоидальное колебание по Z.}$$

Вместе (x, y, z) дают **3D-спираль** с увеличивающимся радиусом.

Реализацию данной функции см. в листинге 1.

Листинг 1. Реализация генерации контрольных точек

```
1 // === Генерация контрольных точек ===
2 std::vector<Eigen::Vector3d> generate_control_points(int num_waves = 4, double amplitude =
      2.0, int points_per_turn = 3) {
3     std::vector<Eigen::Vector3d> control_points;
4     int steps = num_waves * points_per_turn + 1;
5     for (int i = 0; i < steps; ++i) {
6         double x = i * 1.5;
7         double y = amplitude * sin(i * M_PI / 1.5);
8         double z = amplitude * cos(i * M_PI / 1.5);
9         control_points.emplace_back(x, y, z);
```

```
    }
    return control_points;
}
```

3.2 Функция построения B-сплайна матричным методом (matrix_bspline)

Назначение

Реализует построение кубического B-сплайна на основе матричного представления.

Параметры

- `control_points` — список контрольных точек вида $[[x_0, y_0, z_0], [x_1, y_1, z_1], \dots]$

- `num_points` — общее число точек, в которых будет рассчитана кривая

Логика работы

Определение базисной матрицы M_{bspline}: Это фиксированная матрица для кубического B-сплайна ($p = 3$):

$$M_{\text{bspline}} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Подготовка к построению кривой: - $n_{\text{segments}} = \text{len(control\_points)} - 3$ — количество сегментов, так как каждый сегмент зависит от 4 соседних точек.

- `curve` — список для хранения точек кривой.

Цикл по сегментам: Для каждого сегмента выбираются 4 последовательные точки:

$$G = \texttt{control\_points}[i : i + 4]$$

Цикл по параметру $t \in [0, 1]$: Для каждого значения t формируется вектор параметров:

$$T = [t^3,\ t^2,\ t,\ 1]$$

Матричное умножение: Точка на кривой вычисляется как:

$$C(t) = T \cdot M_{\text{bspline}} \cdot G$$

где:

- T — вектор параметров,

- M_{bspline} — матрица базисных функций,

- G — геометрическая матрица из 4 точек текущего сегмента.

Сборка результата: Все точки добавляются в список `curve`. Функция возвращает массив curve и время выполнения.

Релизацию данной функции см. в листинге 2.

Листинг 2. Реализация построения B-сплайна матричным методом

```
// === Матрица B-Spline ===
Matrix4d get_bspline_matrix() {
    return (Matrix4d() <<
        -1,  3, -3,  1,
         3, -6,  3,  0,
        -3,  0,  3,  0,
         1,  4,  1,  0
    ).finished() * (1.0 / 6.0);
}

// === Матричный метод построения B-сплайна ===
std::vector<Eigen::Vector3d> matrix_bspline(const std::vector<Eigen::Vector3d>& control_points
    , int num_points = 1000) {
    auto start = std::chrono::high_resolution_clock::now();

    std::vector<Eigen::Vector3d> curve;
    Matrix4d M = get_bspline_matrix();
    int n_segments = control_points.size() - 3;
    int samples_per_segment = num_points / n_segments;

    for (int i = 0; i < n_segments; ++i) {
        // G - матрица из 4 контрольных точек
        Matrix4d G;
        for (int j = 0; j < 4; ++j) {
            auto pt = control_points[i + j];
            G.col(j) << pt.x(), pt.y(), pt.z(), 1.0;
        }

        for (int k = 0; k < samples_per_segment; ++k) {
            double t = static_cast<double>(k) / (samples_per_segment - 1);
            Vector4d T(t*t*t, t*t, t, 1.0);
            Vector4d result = T * M * G;
            curve.emplace_back(result[0], result[1], result[2]);
        }
    }

    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> elapsed = end - start;
    std::cout << "Время выполнения (матричный): " << elapsed.count() << " сек\n";

    return curve;
}
```

3.3 Функция построения B-сплайна при помощи библиотечных функций

Используемые библиотеки:

– Geom_BSplineCurve

Предназначение: класс, представляющий B-сплайн кривую в 3D пространстве.

Реализует как B-сплайны, так и NURBS (если использовать веса).

Можно оценивать точки, производные, длину, касательные и т.д.

– gp_Pnt

Предназначение: структура данных, представляющая точку в 3D пространстве (x, y, z).

Логика работы

1. Создание объекта кривой

Используется класс Geom_BSplineCurve, который реализует обобщённую формулу B-сплайна для построения гладкой кривой в трёхмерном пространстве.

2. Установка параметров кривой

Степень сплайна (degree) — задаёт степень базисных полиномов, определяющих форму кривой (в данном случае — кубический сплайн, degree = 3).

Контрольные точки (poles) — массив точек, влияющих на форму кривой. Передаются как входной вектор и преобразуются в формат, используемый OpenCASCADE.

Узловой вектор (knots) — последовательность значений параметра, определяющая, как сегменты кривой связаны между собой. В данном случае используется равномерный узловой вектор.

Кратности узлов (mults) — количество повторений каждого узла. Устанавливается одинаковым для всех узлов, что обеспечивает начало и конец кривой в первом и последнем полюсе соответственно.

3. Построение кривой

На основе переданных параметров библиотека OpenCASCADE вычисляет внутреннее представление B-сплайна, позволяя в дальнейшем получать точки, производные и другие характеристики кривой.

4. Возврат объекта кривой

Функция возвращает указатель на созданный объект Geom_BSplineCurve, который можно использовать для последующих вычислений или визуализации.

Реализацию данной функции см. в листинге 4.

Листинг 3. Реализация построения B-сплайна при помощи библиотечных функций

```
1 Handle(Geom_BSplineCurve) create_bspline_curve(const std::vector<Eigen::Vector3d>&
      control_points) {
2     int degree = 3;
3     int n = control_points.size();
```

```
4       TColgp_Array1OfPnt poles(1, n);
5       for (int i = 0; i < n; ++i)
6           poles.SetValue(i + 1, gp_Pnt(control_points[i].x(), control_points[i].y(),
                control_points[i].z()));
7
8       // Создание равномерного вектора узлов
9       TColStd_Array1OfReal knots(1, n - degree + 1);
10      for (int i = 1; i <= n - degree + 1; ++i)
11          knots.SetValue(i, i);
12
13      TColStd_Array1OfInteger mults(1, n - degree + 1);
14      mults.Init(degree + 1);  // Uniform multiplicity
15
16      return new Geom_BSplineCurve(poles, knots, mults, degree);
17  }
```

4 Результаты работы

На рис. 3 представлен результат работы алгоритма матричным методом при общем числе точек 400.

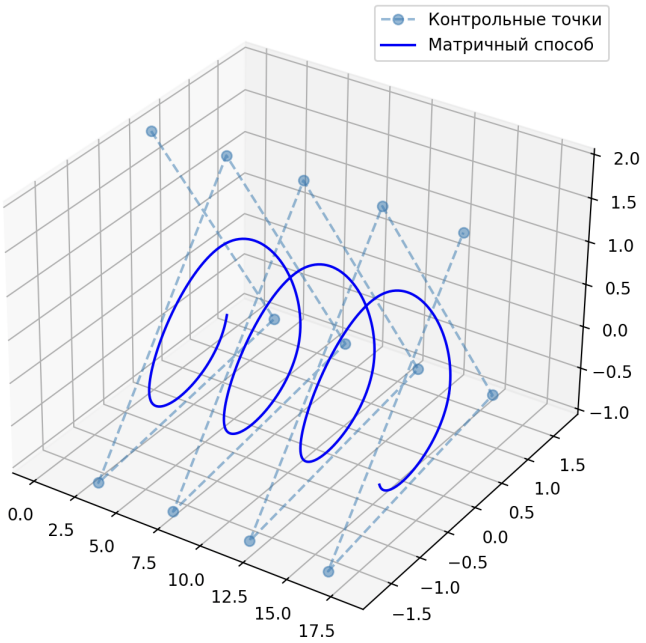


Рис. 2. Результат работы реализованного алгоритма №1

На рис. 5 представлен результат работы алгоритма при помощи библиотечных методов при общем числе точек 400.

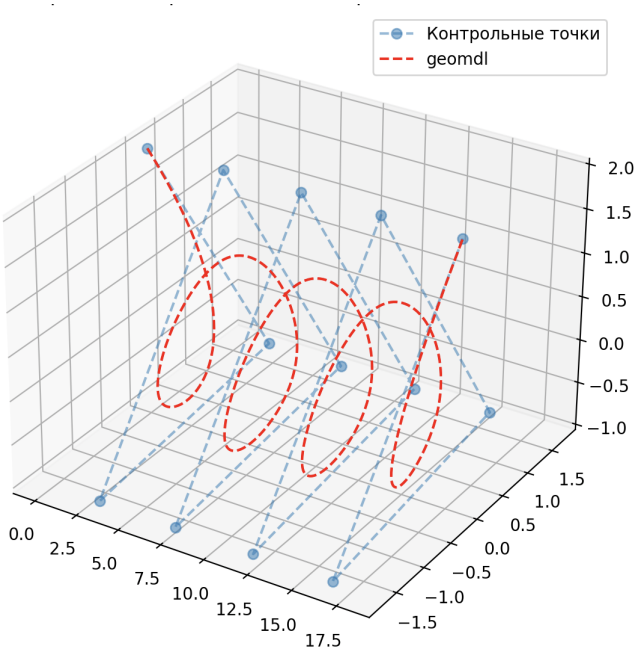


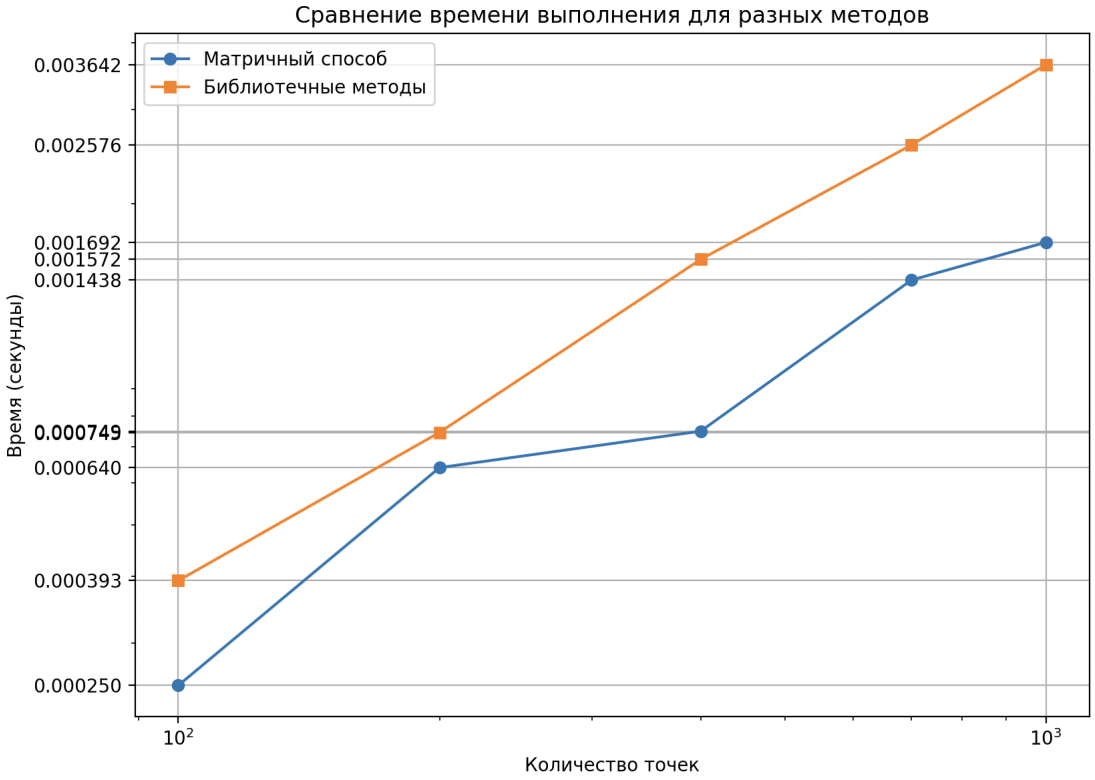
Рис. 3. Результат работы библиотечного алгоритма

5 Сравнение результатов работы алгоритмов

После реализации собственого алгоритма построения B-сплайновых кривых и использования библиотечного метода Geom_BSplineCurve был проведен анализ времени выполнения для различного количества контрольных точек. Результаты сравнения производительности представлены в таблице 1.

Кол-во точек	Матричный способ	Библиотечные методы
100	0.000250 сек	0.000393 сек
200	0.000640 сек	0.000745 сек
400	0.000749 сек	0.001572 сек
700	0.001438 сек	0.002576 сек
1000	0.001692 сек	0.003642 сек

Таблица 1. Сравнение среднего времени выполнения алгоритмов



1. Матричный способ

Почему быстрее:

– Использует предварительно предвычисленную базисную матрицу

– Вычисления сводятся к простым матричным операциям (умножение матриц и векторов)

– Линейная сложность O(n) по количеству сегментов

– Нет рекурсивных вызовов

– Минимальное количество операций на каждой итерации

2. Библиотечный метод (Geom_BSplineCurve)

Особенности:

– Оптимизированная реализация на C/C++

– Использует эффективные алгоритмы вычисления

– Имеет дополнительную логику для обработки разных случаев (проверки, безопасность)

– Накладные расходы на вызовы функций библиотеки

– Автоматическая обработка краевых условий

Заключение

В результате выполнения лабораторной работы были изучены теоретические основы B-сплайновых кривых, рассмотрены их свойства, такие как локальность, гладкость, выпуклая оболочка и разбиение единицы. Также был проанализирован алгоритм построения кривых:

1. Матричная форма B-сплайна, применяемая для кубических сплайнов (p = 3). Данный подход обеспечивает высокую скорость вычисления за счёт предварительно заданных матриц и отсутствия рекурсии. Его сложность составляет $O(m \cdot n)$, что делает его особенно эффективным для программной реализации.

Было выполнено:

- Реализован алгоритм для построения B-сплайновой кривой.

- Построены B-сплайновые кривые и визуализированы для разного количества точек.

- Выполнена сравнительная оценка производительности собственного алгоритма и библиотечной реализации через библиотеку Geom_BSplineCurve.

Анализ показал, что:

– Матричный метод является быстрее, так как использует оптимизированные матричные операции без рекурсии.

– Библиотечный метод (Geom_BSplineCurve) показал хорошую производительность, однако уступает матричному методу из-за дополнительных проверок и обработки общих случаев.

Таким образом, работа позволила не только изучить математическую модель B-сплайнов, но и провести сравнительный анализ различных способов их программной реализации.