Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский Нижегородский
государственный университет им. Н.И. Лобачевского»
Институт информационных технологий, математики и механики

Отчет по лабораторной работе
ИНТЕРПОЛЯЦИЯ ФУНКЦИЙ КУБИЧЕСКИМИ СПЛАЙНАМИ

Выполнил: Власов Максим Сергеевич, студент группы 381806-1
Проверил: к. ф.-м. н., доцент кафедры ДУМЧА Эгамов А. И.

Нижний Новгород
2020

Содержание

Введение

В повседневной жизни человеку постоянно приходится сталкиваться с различными процессами или изменяющимися объектами. Состояния таких объектов иногда необходимо математически описать, и сделать это можно с помощью функций. Но за реальными процессами невозможно следить непрерывно, поэтому состояния фиксируются с определенными интервалами. В то же время может потребоваться смоделировать поведение объекта непрерывно.

Способы решения этой задачи предлагает вычислительная математика, предметом изучения которой являются численные методы решения задач математического анализа. Главным разделом вычислительной математики является реализация численных методов на ЭВМ, то есть составление программы для требуемого алгоритма и решения с ее помощью конкретной задачи.

Данная работа направлена на изучение методов вычисления и построения кубических сплайнов по заданным точкам. Основной учебной целью работы является практическое освоение вычислительных методов. В ходе выполнения разрабатывается общая форма представления кубического сплайна, программная реализация метода прогонки, разрабатываются программы работы с подобными структурами, которые могут быть использованы и в других областях приложений.

1. Постановка задачи

Задача: разработать и реализовать приложение, выполняющее построение приближения (интерполирование) функции кубическими многочленами.

Входные данные: точки интерполируемой функции.

Выходные данные: изображение графика интерполированной функции.

2. Руководство пользователя

В данном руководстве содержатся пошаговые инструкции по работе с программой, для того чтобы вы могли как можно быстрее приступить к использованию приложения.

1. Запустите файл **01_spline.exe** из папки с программой. Перед вами отобразится основное окно программы. Ознакомьтесь с пунктами меню “Help” и “About” в верхней части окна, чтобы узнать о возможностях и разработчиках программы.

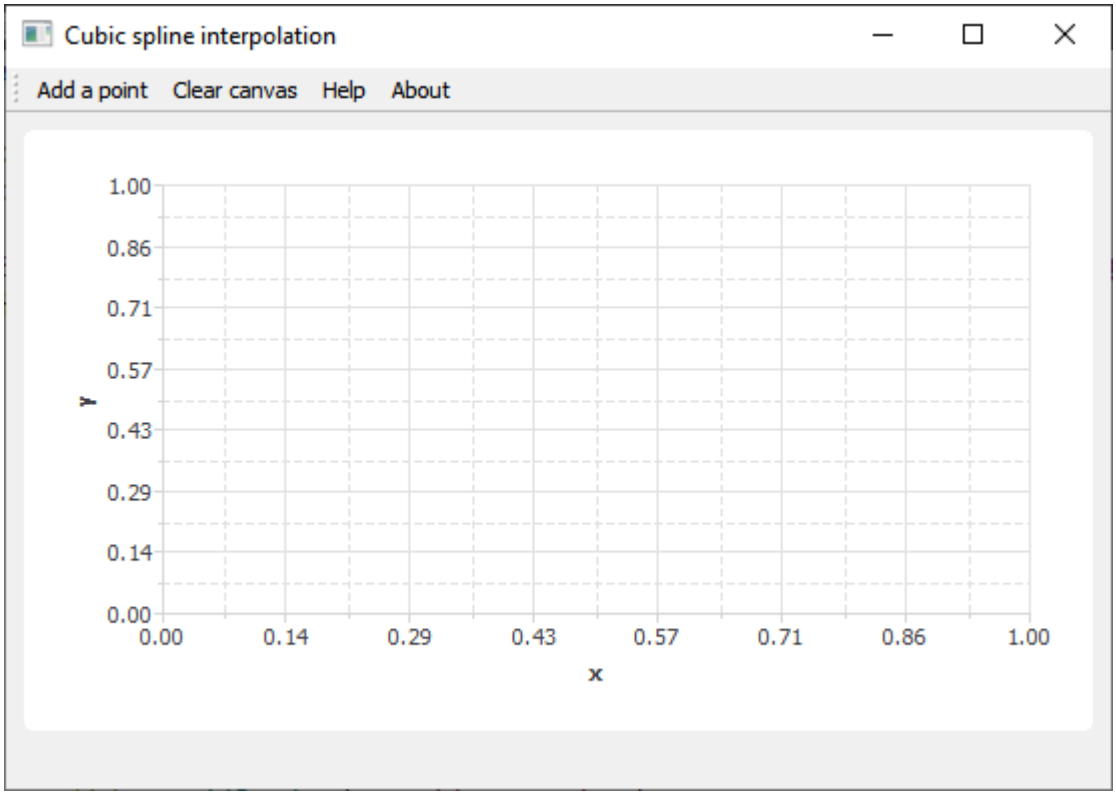

Рис 1. Программа после запуска.

2. Нажмите “Add a point”, чтобы вызвать окно добавления точки. Введите координаты и нажмите “OK”. В случае неверного ввода вы увидите сообщение об ошибке.

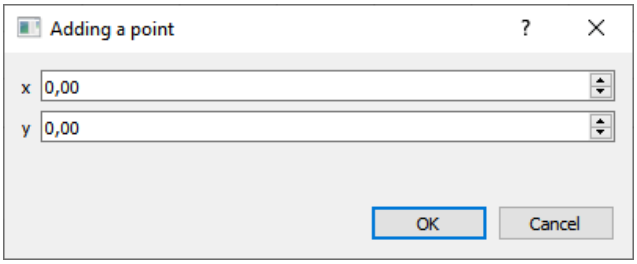

Рис 2. Окно добавления точки.

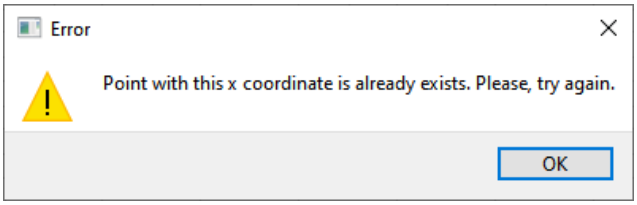

Рис 3. Сообщение об ошибке.

3. Повторите процедуру ввода координат точек необходимое количество раз. По мере добавления точки будут отображаться на полотне. Как только их число превысит 1, программа автоматически будет перестраивать график интерполированной функции при каждом добавлении точки.

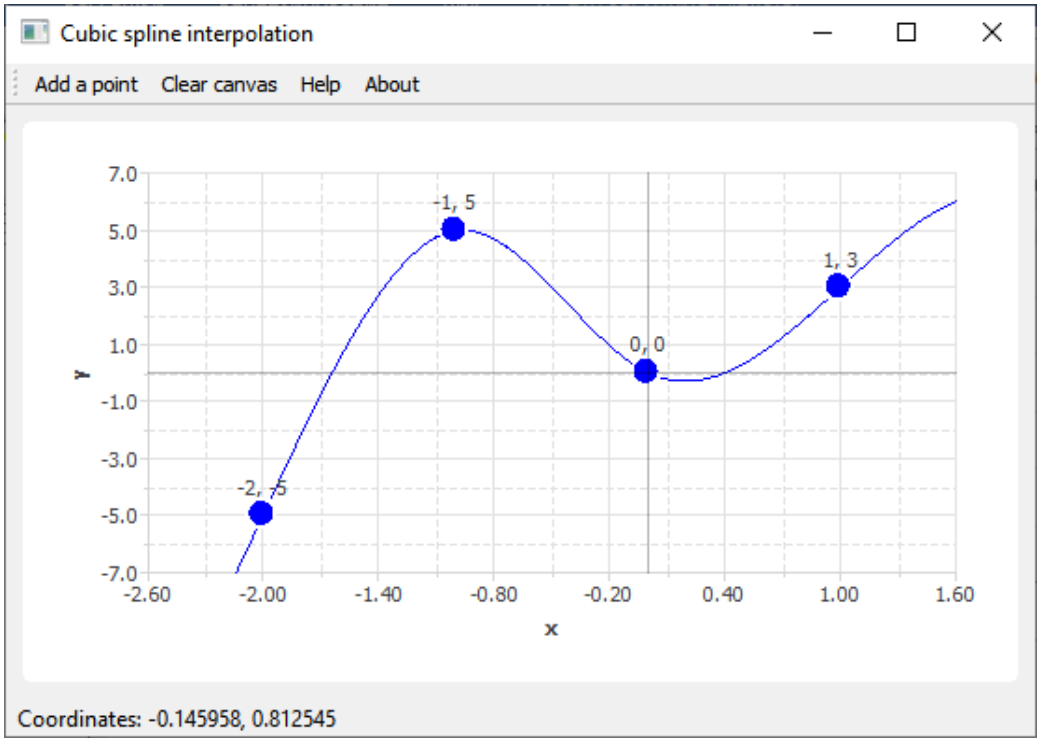

Рис 4. Пример построенного сплайна.

4. Чтобы отобразить координаты точек, наведите курсор мыши на любую из точек. Чтобы проследить координаты на линии графика, наведите курсор мыши на график, и его координаты отобразятся в нижней части окна.
5. Чтобы очистить полотно и начать заново, нажмите “Clear canvas”.
6. По завершении работы с программой нажмите крестик в правом верхнем углу окна.

3. Руководство программиста

3.1. Структура программы

Исходный код программы содержится в следующих модулях:

1. **Main** (включает в себя файл main.cpp) – основной модуль (точка входа, использующая модули ниже).
2. **MainWindow** (включает в себя файлы mainwindow.h, mainwindow.cpp, mainwindow.ui) – модуль Основное окно (содержит объявление и реализацию класса, отвечающего за основное окно программы).
3. **Spline** (включает в себя файлы spline.h и spline.cpp) – модуль Сплайн (содержит объявление и реализацию класса, хранящего координаты точек и вычисляющего мета-данные, необходимые для построения графика).
4. **SplineChart** (включает в себя файлы splinechart.h и splinechart.cpp) – модуль График (содержит объявление и реализацию класса, выполняющего построение графика и размещение координатных осей, точек и сетки).
5. **PointDialog** (включает в себя файлы pointdialog.h, pointdialog.cpp, pointdialog.ui) – модуль Окно добавления точки (содержит объявление и реализацию класса, отвечающего за получение координат точки от пользователя).
6. **LoadingDialog** (включает в себя файлы loadingdialog.h, loadingdialog.cpp, loadingdialog.ui) – модуль Окно загрузки (содержит объявление и реализацию класса, отвечающего за отображение индикатора занятости приложения).
7. **HelpDialog** (включает в себя файлы helpdialog.h, helpdialog.cpp, helpdialog.ui) – модуль Окно справки (содержит объявление и реализацию класса, отвечающего за окно с информацией о работе с программой).
8. **AboutDialog** (включает в себя файлы aboutdialog.h, aboutdialog.cpp, aboutdialog.ui) – модуль Окно информации о программе (содержит объявление и реализацию класса, отвечающего за окно с информацией о создателях программы).

3.2. Описание алгоритмов

3.2.1. Сплайн

Пусть отрезок $[a, b]$ разбит на N равных частичных отрезков $[x_i,\ x_{i+1}]$, где $x_i = a + i * h, i = 0, 1, \dots, N - 1, x_n = b, h = \frac{(b - a)}{N}$. **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.

Максимальная по всем частичным отрезкам степень многочленов называется **степенью сплайна**, а разность между степенью сплайна и порядком наивысшей непрерывной на [a, b] производной - **дефектом сплайна**.

Например, непрерывная кусочно-линейная функция (ломаная) является сплайном первой степени с дефектом, равным единице, так как непрерывна только сама функция (нулевая производная), а первая производная уже имеет разрыв.

На практике наиболее широкое распространение получили сплайны третьей степени, имеющие на $[a, b]$ непрерывную, по крайней мере, первую производную. Эти сплайны называются кубическими и обозначаются через $S(x)$.

3.2.2. Кубический сплайн

Пусть на отрезке $[a, b]$ задана функция $y = f(x)$ и сетка узлов

$$a = x_0 < x_1 < x_2 < \dots < x_n = b$$

Тогда кубическим сплайном функции $y = f(x)$ называется функция $S(x)$, которая:

- на каждом отрезке $[x_i,\ x_{i+1}]$ является многочленом степени не выше третьей;
- имеет непрерывные первую и вторую производные на всём отрезке $[a, b]$;
- в точках x_i выполняется равенство $S(x_i) = f(x_i)$, т. е. сплайн S(x) интерполирует функцию f в точках x_i.

Для однозначного задания сплайна перечисленных условий недостаточно, для построения сплайна необходимо наложить дополнительные требования — граничные условия:

- “естественный сплайн” – граничные условия вида $S``(a) = S``(b) = 0$;
- понижение степени сплайна на краях до второй $S```(a) = S```(b) = 0$;
- периодический сплайн $S`(x_0) = S`(x_N), S``(x_0) = S``(x_N)$.

Далее мы будем использовать естественный способ задания сплайна.

Поскольку сплайн кубический, т. е. имеет степень равную 3, все $s_i(x)$ функции составляющий его будут являться многочленами степени 3.

Запишем общий вид $s_i(x)$:

$$s_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3.$$

Фактически, задача была сведена к отысканию коэффициентов упомянутого полинома 3 степени на каждом из отрезков.

Эта форма записи соответствует ряду Тейлора для многочлена $s_i(x)$ в окрестности точки x_i. Так как $s_i(x)$ – кубический многочлен, его ряд Тейлора заканчивается после кубического слагаемого. Поэтому можем сделать вывод, что коэффициенты есть

$$a_i = s_i(x_i), b_i = s_i'(x_i), c_i = s_i''(x_i), d_i = s_i'''(x_i).$$

Сформулируем условия непрерывности и гладкости сплайна в терминах коэффициентов a_i, b_i, c_i, d_i. Так же введем обозначение $h_i = x_i - x_{i-1}$ - это есть длина i-ого отрезка. Запишем условия непрерывности функции в точке x_{i-1}:

$$a_{i-1} = a_i - b_i h_i + \frac{c_i}{2}h_i^2 - \frac{d_i}{6}h_i^3, i = 2, \dots, N.(1)$$

Аналогично с условиями непрерывности первой и второй производной функции в точке x_{i-1}:

$$b_{i-1} = b_i - c_i h_i + \frac{d_i}{2}h_i^2, i = 2, \dots, N,(2)$$

$$c_{i-1} = c_i - d_i h_i, i = 2, \dots, N.(3)$$

Учитывая условия интерполирования функции получаем:

$$a_i = s_i(x_i) = S(x_i) = f(x_i), i = 1, \dots, N.$$

Кроме этого, ещё и условие в точке x_0,

$$a_1 + b_1(x_0 - x_1) + \frac{c_1}{2}(x_0 - x_1)^2 + \frac{b_1}{6}(x_0 - x_1)^3 = s_1(x_0) = S(x_0) = f(x_0).$$

Нет необходимости требовать выполнения условия $s_{i+1}(x_i) = S(x_i)$, так как оно автоматически выполняется при выполнении условий непрерывности.

$$a_i = f(x_i), i = 1, \dots, N, (4)$$

$$a_1 - b_1 h_1 + \frac{c_1}{2}h_1^2 - \frac{d_1}{6}h_1^3 = f(x_0)(5).$$

Объединяя полученные уравнения (1), (2), (3), (4), (5) получаем систему уравнений.

3.2.3. Система уравнений

$$\begin{cases} a_{i-1} = a_i - b_i h_i + \frac{c_i}{2} h_i^2 - \frac{d_i}{6} h_i^3, i = 2, \dots, N, \\ b_{i-1} = b_i - c_i h_i + \frac{d_i}{2} h_i^2, i = 2, \dots, N, \\ c_{i-1} = c_i - d_i h_i, i = 2, \dots, N, \\ a_i = f(x_i), i = 1, \dots, N, \\ a_1 - b_1 h_1 + \frac{c_1}{2} h_1^2 - \frac{d_1}{6} h_1^3 = f(x_0). \end{cases},$$

Всего в системе $4N - 2$ уравнения, при этом неизвестных $a_i, b_i, c_i, d_i - 4N$. Ещё два условия выбираются обычно на концах отрезка в точках x_0 и x_N. В этом случае их называют краевыми условиями. Как говорилось раннее мы будем использовать «Естественный сплайн», поэтому, краевые условия:

$$c_N = s_N''(x_N) = S''(x_N) = 0$$
$$c_1 - d_1 h_1 = s_1''(x_0) = S''(x_0) = 0$$

В итоге получаем следующую систему:

$$\begin{cases} a_{i-1} = a_i - b_i h_i + \frac{c_i}{2} h_i^2 - \frac{d_i}{6} h_i^3, i = 2, \dots, N, \\ b_{i-1} = b_i - c_i h_i + \frac{d_i}{2} h_i^2, i = 2, \dots, N, \\ c_{i-1} = c_i - d_i h_i, i = 2, \dots, N, \\ a_i = f(x_i), i = 1, \dots, N, \\ a_1 - b_1 h_1 + \frac{c_1}{2} h_1^2 - \frac{d_1}{6} h_1^3 = f(x_0), \\ c_N = 0, \\ c_1 - d_1 h_1 = 0. \end{cases}$$

Данную систему возможно значительно упростить, сведя ее к системе линейных уравнений.

1. Исключим из этой системы неизвестные a_i. Заодно воспользуемся обозначением разделенных разностей Ньютона:

$$f(x_{i-1}, x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{a_i - a_{i-1}}{h_i}$$

Получаем систему:

$$\begin{cases} b_i - \frac{c_i}{2}h_i + \frac{d_i}{6}h_i^2 = f(x_{i-1}, x_i), i = 2, \dots, N, \\ b_{i-1} = b_i - c_i h_i + \frac{d_i}{2}h_i^2, i = 2, \dots, N, \\ c_{i-1} = c_i - d_i h_i, i = 2, \dots, N, \\ b_1 - \frac{c_1}{2}h_1 + \frac{d_1}{6}h_1^2 = f(x_0, x_1), \\ c_N = 0, \\ c_1 - d_1 h_1 = 0. \end{cases}$$

2. Выразим из уравнений (3) и (7) $d_i h_i$, исключим d_i из системы, приведя подобные при c_i. Получаем

$$\begin{cases} b_i - \frac{c_i}{2}h_i + \frac{c_{i-1}}{6}h_i = f(x_{i-1}, x_i), i = 2, \dots, N, \\ b_{i-1} - b_i - \frac{c_i}{2}h_i + \frac{c_{i-1}}{2}h_i = 0, i = 2, \dots, N, \\ b_1 - \frac{c_1}{2}h_1 + \frac{d_1}{6}h_1^2 = f(x_0, x_1), \\ c_N = 0. \end{cases}$$

3. Выразим b_i и подставим в уравнение (2). Так же доопределим коэффициент $c_0 = 0$. Получаем

$$\begin{cases} \frac{h_i}{6}c_{i-2} + \frac{h_i + h_{i-1}}{3}c_{i-1} + \frac{h_i}{6}c_i = f(x_{i-2}, x_{i-1}, x_i), i = 2, \dots, N, \\ c_N = c_0 = 0. \end{cases}$$

4. Введем понятие разделенной разности Ньютона для трех аргументов

$$f(x_{i-2}, x_{i-1}, x_i) = \frac{f(x_{i-1}, x_i) - f(x_{i-2}, x_{i-1})}{x_i - x_{i-2}}.$$

В результате получаем систему относительно $c_1, \dots, c_{N-1}$. Тем самым приведя ее к трехдиагональному виду.

$$\begin{array}{ccccccc} 2c_1 & + & \frac{h_2}{h_1 + h_2}c_2 & & & = & 6f(x_0, x_1, x_2), \\ & \ddots & & & & & \\ \frac{h_i}{h_i + h_{i+1}}c_{i-1} & + & 2c_i & + & \frac{h_{i+1}}{h_i + h_{i+1}}c_{i+1} & = & 6f(x_{i-1}, x_i, x_{i+1}), \\ & & & \ddots & & & \\ & & \frac{h_{N-1}}{h_{N-1} + h_N}c_2 & + & 2c_{N-1} & = & 6f(x_{N-2}, x_{N-1}, x_N). \end{array}$$

3.2.4. Метод прогонки

Данный метод применяется для численного решения систем с трехдиагональной матрицей, который представляет собой один из вариантов метода Гаусса. Рассмотрим метод на примере следующей системы:

$$\begin{cases} b_1x_1 + c_1x_2 = d_1, \\ a_ix_{i-1} + b_ix_i + c_ix_{i+1} = d_i, i = 2,3,\dots,n-1, \\ a_nx_{n-1} + b_nx_n = d_n. \end{cases}$$

Смысл метода в том, чтобы рекуррентно выразить x_i через x_{i+1}. Для этого первое уравнение системы разрешим относительно x_1 и получают

$x_1 = p_1x_2 + q_1$, где $p_1 = -\frac{c_1}{b_1}, q_1 = \frac{d_1}{b_1}$.

Найденное выражение для x_1 подставляют во второе уравнение системы. Тогда это уравнение преобразуется в

$$(a_2p_1 + b_2)x_2 + c_2x_3 = d_2 - a_2q_1.$$

Это уравнение не содержит x_1. Разрешив его относительно x_2, получаем

$$x_2 = p_2x_3 + q_2, \text{ где } p_2 = -\frac{c_2}{a_2p_1+b_2}, q_2 = \frac{d_2-a_2q_1}{a_2p_1+b_2}.$$

Аналогично поступая с выражением для x_2. В результате получаем соотношения

$$x_i = p_ix_{i+1} + q_i, \text{ где}$$

$$p_i = -\frac{c_i}{a_ip_{i-1} + b_i}, i = 2,3,\dots,n-1,$$

$$q_i = \frac{d_i - a_iq_{i-1}}{a_ip_{i-1} + b_i}, i = 2,3,\dots,n-1,$$

$$x_n = q_n.$$

Теперь для вычисления неизвестных $x_1, x_2, \dots, x_n$ вычисляются все коэффициенты p_i и q_i . Операцию вычисления называют прямой прогонкой, а коэффициенты p_i и q_i – прогоночными коэффициентами. После вычисления всех коэффициентов, находим x_n, и по рекуррентным формулам находятся $x_{n-1}, x_{n-2}, \dots, x_2, x_1$. Эта операция называется обратной прогонкой.

3.2.5. Блок-схемы алгоритмов

3.2.5.1. Метод прогонки

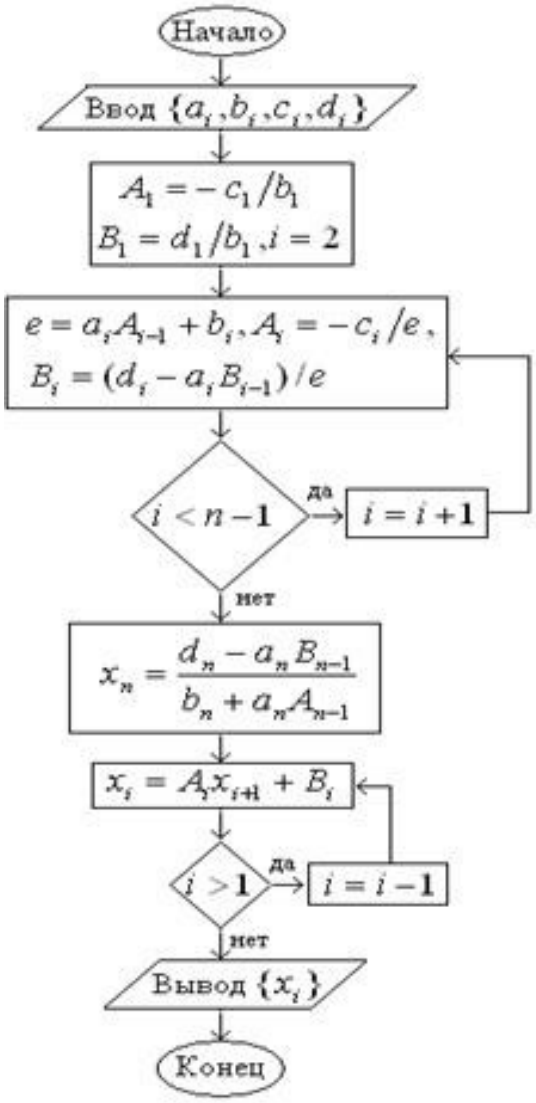

Рис 5. Блок-схема метода прогонки.

3.2.5.2. Интерполяция кубическими сплайнами

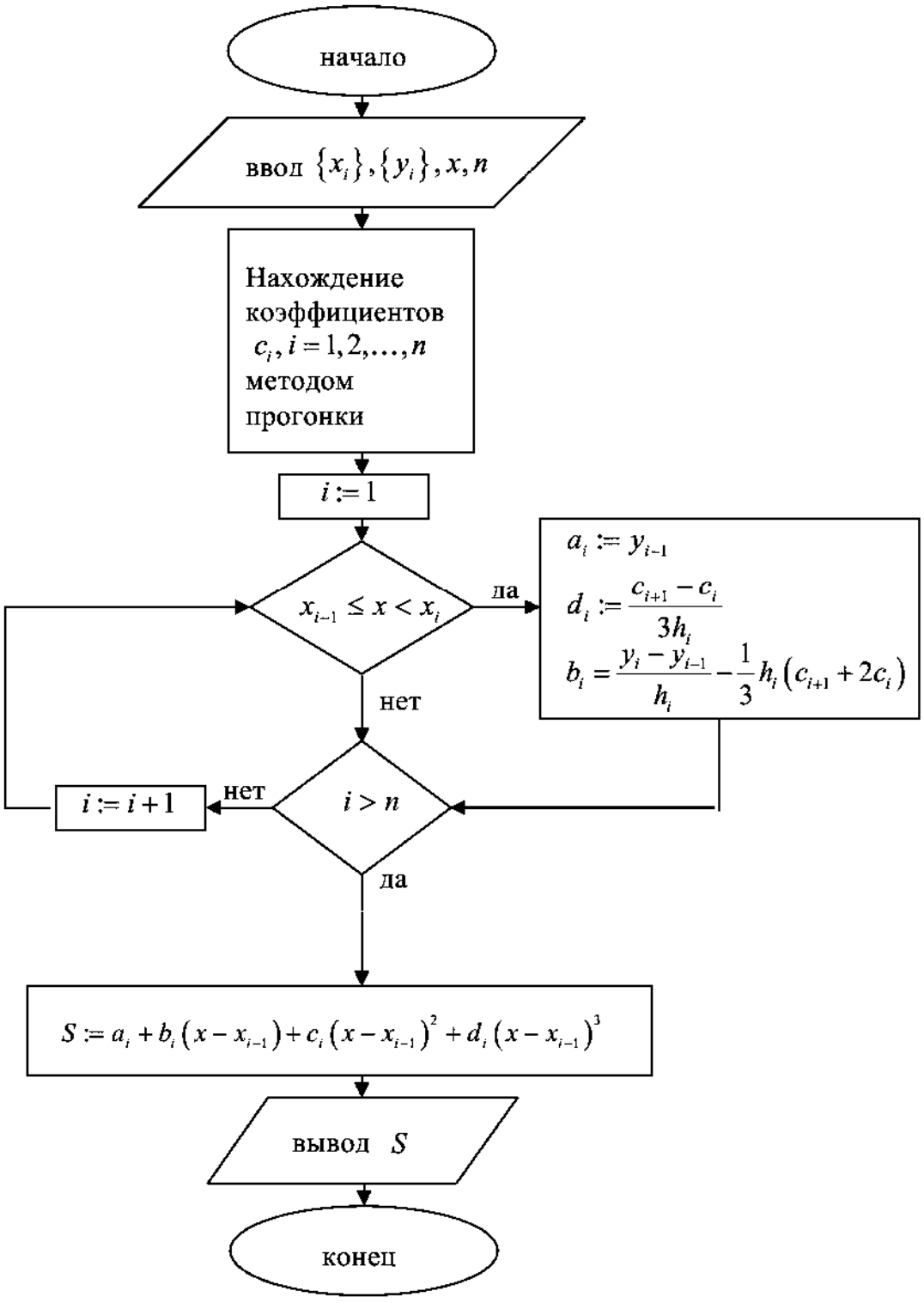

Рис 6. Блок-схема алгоритма интерполяции кубическими сплайнами.

3.3. Описание структур данных и функций

Приложение построено на фреймворке Qt для C++. Все описанные ниже классы (за исключением Spline) являются мета-сущностями Qt и содержат, помимо обычных полей и методов: реализацию механизма обмена данными, сокрытую в Q_OBJECT; специальные поля-структуры ui, инкапсулирующие указатели на элементы интерфейса; и не имеющие реализации особые методы-сигналы.

3.3.1. Класс MainWindow

Объявление класса:

```
class MainWindow : public QMainWindow
{
    Q_OBJECT
    SplineChart *p_chart;
    Spline m_spline;
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private:
    Ui::MainWindow *ui;
private slots:
    void showAddPointDialog();
    void showAboutDialog();
    void showHelpDialog();
    void clearCanvas();
    void statusCoordinates(const QPointF& point);
};
```

3.3.1.1. Поля класса

`SplineChart *p_chart;`

Назначение: хранит указатель на структуру данных, представляющую график.

`Spline m_spline;`

Назначение: хранит экземпляр структуры данных интерполированной функции.

3.3.1.2. Методы класса

```
explicit MainWindow(QWidget *parent = nullptr);
```

Назначение: конструктор инициализации.

Входные параметры: parent – указатель на родительский виджет.

Выходные данные: отсутствуют.

```
~MainWindow();
```

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void showAddPointDialog();
```

Назначение: вызывает диалог для ввода координат очередной точки.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void showAboutDialog();
```

Назначение: вызывает диалог с информацией о разработчиках программы.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void showHelpDialog();
```

Назначение: вызывает диалог с информацией о работе с программой.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void clearCanvas();
```

Назначение: вызывает очистку полотна.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void statusCoordinates(const QPointF& point);
```

Назначение: отображает координаты точки на графике, на которую указывает курсор мыши пользователя в данный момент.

Входные параметры: point – координаты указываемой точки.

Выходные данные: отсутствуют.

3.3.2. Класс Spline

Объявление класса:

```
class Spline
{
    std::vector<QPointF> m_points;
    std::vector<double> m_a;
    std::vector<double> m_b;
    std::vector<double> m_c;
    std::vector<double> m_d;
public:
    Spline() = default;
    ~Spline() = default;
    void insert(const QPointF& point);
    void removeAll();
    const std::vector<QPointF>& points() const;
    double interpolatedValue(size_t i, double x) const;
    bool available() const;
protected:
    void update();
    void tridiagonalMatrixAlgorithm();
};
```

3.3.2.1. Поля класса

`std::vector<QPointF> m_points;`

Назначение: хранит набор точек, указанных пользователем.

`std::vector<double> m_a;`

Назначение: хранит свободные члены полинома.

`std::vector<double> m_b;`

Назначение: хранит коэффициенты при первой степени полинома.

`std::vector<double> m_c;`

Назначение: хранит коэффициенты при второй степени полинома.

`std::vector<double> m_d;`

Назначение: хранит коэффициенты при старшей степени полинома.

3.3.2.2. Методы класса

```
Spline() = default;
```

Назначение: конструкто по умолчанию.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
~Spline() = default;
```

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void insert(const QPointF& point);
```

Назначение: добавляет точку к сплайну.

Входные параметры: point - точка.

Выходные данные: отсутствуют.

```
void removeAll();
```

Назначение: удаляет все точки сплайна.

Входные параметры: отсутствует.

Выходные данные: отсутствуют.

```
double interpolatedValue(size_t i, double x) const;
```

Назначение: вычисляет интерполированное значение функции.

Входные параметры: i – номер шага (индекс точки), x – абсцисса, для которой нужно вычислить ординату.

Выходные данные: ордината точки.

```
bool available() const;
```

Назначение: определяет, доступно ли построение графика (готовы ли данные).

Входные параметры: отсутствуют.

Выходные данные: истина или ложь.

```
void update();
```

Назначение: обновляет мета-данные графика и данные, необходимые для построения.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void tridiagonalMatrixAlgorithm();
```

Назначение: выполняет прогонку.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
const std::vector<QPointF>& points() const;
```

Назначение: возвращает ссылку на набор добавленных точек.

Входные параметры: отсутствуют.

Выходные данные: набор точек.

3.3.3. Класс SplineChart

Объявление класса:

```
class SplineChart : public QtCharts::QChart
{
    Q_OBJECT
protected:
    QtCharts::QLineSeries    *p_spline_series;
    QtCharts::QLineSeries    *p_axis_h_series;
    QtCharts::QLineSeries    *p_axis_v_series;
    QtCharts::QScatterSeries *p_points_series;
    QtCharts::QValueAxis *p_axis_x;
    QtCharts::QValueAxis *p_axis_y;
    qreal m_min_x;
    qreal m_max_x;
    qreal m_min_y;
    qreal m_max_y;
    void resetRanges();
public:
    explicit SplineChart(QGraphicsItem *parent = nullptr);
    ~SplineChart();
    void load(const Spline& spline);
public slots:
    void showPointLabels(const QPointF&, bool hovered);
    void emitSplineHovered(const QPointF& point);
    void clear();
signals:
    void proceed(int current, int total, const QString& message);
    void splineHovered(const QPointF& point);
};
```

3.3.3.1. Поля класса

`QtCharts::QLineSeries *p_spline_series;`

Назначение: хранит указатель на последовательность вычисленных точек интерполированной функции.

`QtCharts::QLineSeries *p_axis_h_series;`

Назначение: хранит указатель на последовательность точек, необходимых для отображения горизонтальной координатной оси.

`QtCharts::QLineSeries *p_axis_v_series;`

Назначение: хранит указатель на последовательность точек, необходимых для отображения вертикальной координатной оси.

```
QtCharts::QScatterSeries *p_points_series;
```

Назначение: хранит указатель на последовательность точек, указанных пользователем.

```
QtCharts::QValueAxis *p_axis_x;
```

Назначение: хранит указатель на горизонтальную ось отображения.

```
QtCharts::QValueAxis *p_axis_y;
```

Назначение: хранит указатель на вертикальную ось отображения.

```
qreal m_min_x;
```

Назначение: хранит минимальную абсциссу среди абсцисс указанных точек.

```
qreal m_max_x;
```

Назначение: хранит максимальную абсциссу среди абсцисс указанных точек.

```
qreal m_min_y;
```

Назначение: хранит минимальную ординату среди ординат указанных точек.

```
qreal m_max_y;
```

Назначение: хранит максимальную ординату среди ординат указанных точек.

3.3.3.2. Методы класса

```
explicit SplineChart(QGraphicsItem *parent = nullptr);
```

Назначение: конструктор инициализации.

Входные параметры: parent – указатель на родительский графический элемент.

Выходные данные: отсутствуют.

```
~SplineChart();
```

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void load(const Spline& spline);
```

Назначение: загружает данные интерполированной функции, необходимые для построения графика.

Входные параметры: spline – объект с мета-данными функции.

Выходные данные: отсутствуют.

```
void showPointLabels(const QPointF&, bool hovered);
```

Назначение: показывает или скрывает подписи с координатами точек.

Входные параметры: hovered – значение, показывающее, нужно ли скрыть или показать данные.

Выходные данные: отсутствуют.

```
void emitSplineHovered(const QPointF& point);
```

Назначение: вызывает срабатывание обработчика отображения координат графика.

Входные параметры: point – точка на графике.

Выходные данные: отсутствуют.

```
void clear();
```

Назначение: очищает полотно.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

```
void resetRanges();
```

Назначение: пересчитывает границы отображения графика и координатные оси.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

3.3.4. Класс PointDialog

Объявление класса:

```
class PointDialog : public QDialog
{
    Q_OBJECT
protected:
    QPointF m_point;
public:
    explicit PointDialog(QWidget *parent = nullptr);
    ~PointDialog();
    QPointF resultPointF();
private:
    Ui::PointDialog *ui;
};
```

3.3.4.1. Поля класса

`QPointF m_point;`

Назначение: хранит координаты введенной пользователем точки.

3.3.4.2. Методы класса

`explicit PointDialog(QWidget *parent = nullptr);`

Назначение: конструктор инициализации.

Входные параметры: parent – указатель на родительский виджет.

Выходные данные: отсутствуют.

`~ PointDialog();`

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

`QPointF resultPointF();`

Назначение: возвращает координаты точки, введенные пользователем.

Входные параметры: отсутствуют.

Выходные данные: объект точки.

3.3.5. Класс LoadingDialog

Объявление класса:

```
class LoadingDialog : public QDialog
{
    Q_OBJECT
public:
    explicit LoadingDialog(const SplineChart* chart, QWidget *parent = nullptr);
    ~LoadingDialog();
public slots:
    void updateState(int current, int total, const QString& message);
private:
    Ui::LoadingDialog *ui;
};
```

3.3.5.1. Методы класса

`explicit LoadingDialog(const SplineChart* chart, QWidget *parent = nullptr);`

Назначение: конструктор инициализации.

Входные параметры: chart – указатель на отслеживаемый график, parent – указатель на родительский виджет.

Выходные данные: отсутствуют.

`~ LoadingDialog();`

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

`void updateState(int current, int total, const QString& message);`

Назначение: обновляет состояние индикатора загрузки.

Входные параметры: current – номер текщщего шага, total – общее число шагов, message – информационное сообщение для пользователя.

Выходные данные: отсутствуют.

3.3.6. Класс HelpDialog

Объявление класса:

```
class HelpDialog : public QDialog
{
    Q_OBJECT
public:
    explicit HelpDialog(QWidget *parent = nullptr);
    ~HelpDialog();
private:
    Ui::HelpDialog *ui;
};
```

3.3.6.1. Методы класса

`explicit HelpDialog(QWidget *parent = nullptr);`

Назначение: конструктор инициализации.

Входные параметры: parent – указатель на родительский виджет.

Выходные данные: отсутствуют.

`~HelpDialog();`

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

3.3.7. Класс AboutDialog

Объявление класса:

```
class AboutDialog : public QDialog
{
    Q_OBJECT
public:
    explicit AboutDialog(QWidget *parent = nullptr);
    ~AboutDialog();
private:
    Ui::AboutDialog *ui;
};
```

3.3.7.1. Методы класса

`explicit AboutDialog(QWidget *parent = nullptr);`

Назначение: конструктор инициализации.

Входные параметры: parent – указатель на родительский виджет.

Выходные данные: отсутствуют.

`~AboutDialog();`

Назначение: деструктор.

Входные параметры: отсутствуют.

Выходные данные: отсутствуют.

3.3.8. Основная программа

`int main();`

Назначение: основная функция (точка входа).

Заключение

В результате работы над лабораторной работой были изучены методы решения систем линейных уравнений специального вида (метод прогонки), получен опыт реализации алгоритмов решения систем линейных уравнений, написано приложение для построения графиков интерполированных функций. Программа полностью соответствует описанным методам и решает поставленную задачу построения кубического сплайна для заданных точек.

Список используемых источников

1. Волков Е.А. Численные методы. [Электронный ресурс]: учеб. — Электрон. дан. — СПБ.: Лань, 2008 — 256 с. — Режим доступа: http://e.lanbook.com/book/54 — Загл. с экрана.
2. Д.П. Кострамов, А.П. Фаворский Вводные лекции по численным методам: Учеб. Пособие. – М.: Логос, 2004. – 184 с.: ил.
3. Кубический сплайн [Электронный ресурс]: Материал из Википедии — свободной энциклопедии URL: https://ru.wikipedia.org/wiki/Кубический_сплайн
4. Шевцов, Г.С. Численные методы линейной алгебры. [Электронный ресурс]: учеб. пособие / Г.С. Шевцов, О.Г. Крюкова, Б.И. Мызникова. — Электрон. дан. — СПб.: Лань, 2011. — 496 с.
5. Qt 5.15 Reference Pages [Электронный ресурс]: Документация к фреймворку Qt. URL: https://doc.qt.io/qt-5/reference-overview.html