Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА «АВТОМАТИЗИРОВАННЫЕ СТАНОЧНЫЕ СИСТЕМЫ»

Индивидуальное задание
по курсу «Вычислительная математика»

Тема: Интерполирование кубическими сплайнами

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)
Проверил: _____ Ямникова О.А.
(подпись)

Тула 2012

Содержание

Задание

Разработать программу, позволяющая по заданной таблице значений функции построить интерполяционный кубический сплайн и его график. Предусмотреть как и ручной ввод таблицы значений, так и из текстового файла, а также возможность генерации таблицы значений с помощью задаваемой аналитически функции. Предусмотреть вывод графика в файл распространенных графических форматов, а также вывод графика вместе с таблицей исходных данных в файл формата PDF.

Математическое описание

Одной из основных задач численного анализа является задача об интерполяции функций. Пусть на отрезке $a \le \xi \le$ задана сетка $\omega = \{x_i : x_0 = a < x_1 < \cdots < x_i < \cdots < x_n = b\}$ и в её узлах заданы значения функции $y(x)$, равные $y(x_0) = y_0, \cdots, y(x_i) = y_i, \cdots, y(x_n) = y_n$. Требуется построить *интерполянту* — функцию $f(x)$, совпадающую с функцией $y(x)$ в узлах сетки:

$$f(x_i) = y_i, i = 0,1,\cdots,n. \quad (1)$$

Основная цель интерполяции — получить быстрый (экономичный) алгоритм вычисления значений $f(x)$ для значений x, не содержащихся в таблице данных.

Интерполируюшие функции $f(x)$, как правило строятся в виде линейных комбинаций некоторых элементарных функций:

$$f(x) = \sum_{k=0}^{N} c_k \Phi_k(x),$$

где $\{\Phi_k(x)\}$ — фиксированный линейно независимые функции, $c_0, c_1, \cdots, c_n$ — не определенные пока коэффициенты.

Из условия (1) получаем систему из $n+1$ уравнений относительно коэффициентов $\{c_k\}$:

$$\sum_{k=0}^{N} c_k \Phi_k(x_i) = y_i, i = 0,1,\cdots,n.$$

Предположим, что система функций $\Phi_k(x)$ такова, что при любом выборе узлов $a < x_1 < \cdots < x_i < \cdots < x_n = b$ отличен от нуля определитель системы.

Тогда по заданным $y_i (i = 1, \cdots, n)$ однозначно определяются коэффициенты $c_k (k = 1, \cdots, n)$.

Интерполяция кубическими сплайнами является частным случаем кусочно-полиномиальной интерполяцией. В этом специальном случае между любыми двумя соседними узлами функция интерполируется кубическим полиномом. его коэффициенты на каждом интервале определяются из условий сопряжения в узлах:

$$f_i = y_i, f'(x_i - 0) = f'(x_i + 0), f''(x_i - 0) = f''(x_i + 0), i = 1, 2, \cdots, n-1.$$

Кроме того, на границе при $x = x_0$ и $x = x_n$ ставятся условия

$$f''(x_0) = 0, f''(x_n) = 0. \quad (2)$$

Будем искать кубический полином в виде

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

Из условия $f_i = y_i$ имеем

$$f(x_{i-1}) = a_i = y_{i-1}, f(x_i) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i, h_i = x_i - x_{i-1}, i = 1, 2, \cdots, n-1.$$

(4)

Вычислим производные:

$$f'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2, f''(x) = 2c_i + 6d_i(x - x_{i-1}), x_{i-1} \le \xi \le \xi_i,$$

и потребуем их непрерывности при $x = x_i$:

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2, c_{i+1} = c_i + 3d_i h_i, i = 1, 2, \cdots, n-1. \quad (5)$$

Общее число неизвестных коэффициентов, очевидно, равно $4n$, число уравнений (4) и (5) равно $4n-2$. Недостающие два уравнения получаем из условия (2) при $x = x_0$ и $x = x_n$:

$$c_1 = 0, c_n + 3d_n h_n = 0.$$

Выражение из (5) $d_i = \frac{c_{i+1} - c_i}{3h_i}$, подставляя это выражение в (4) и исключая $a_i = y_{i-1}$, получим

$$b_i = \left[\frac{y_i - y_{i-1}}{h}\right]_i - \frac{1}{3}h_i(c_{i+1} + 2c_i), i = 1, 2, \cdots, n-1, b_n = \left[\frac{y_n - y_{n-1}}{h}\right]_n - \frac{2}{3}h_n c_n.$$

Подставив теперь выражения для b_i, b_{i+1} и d_i в первую формулу (5), после несложных преобразований получаем для определения c_i разностное уравнение второго порядка

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = 3\left(\frac{y_{i+1}-y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right), i = 1,2,\cdots,n-1. \quad (6)$$

С краевыми условиями

$$c_1 = 0, c_{n+1} = 0. \quad (7)$$

Условие $c_{n+1} = 0$ эквивалентно условию $c_n + 3d_n h_n = 0$ и уравнению $c_{i+1} = c_i + d_i h_i$. Разностное уравнение (6) с условиями (7) можно решить методом прогонки, представив в виде системы линейных алгебраических уравнений вида $A^* x = F$, где вектор x соответствует вектору $\{c_i\}$, вектор F поэлементно равен правой части уравнения (6), а матрица A имеет следующий вид:

где $A_i = h_i, i = 2,\cdots,n, B_i = h_{i+1}, i = 1,\cdots,n-1$ и $C_i = 2(h_i + h_{i+1}), i = 1,\cdots,n$.

Итак, требуется решить систему линейных алгебраических уравнений, в которой матрица имеет вид:

$A_i x_{i-1} + B_i x_i + C_i x_{i+1} = F_i$

$A_1 = C_n = 0$

i=1,2..n

Такая матрица называется трехдиагональной, и для решения систем с такими матрицами используется специальная модификация метода Гаусса под названием «метод прогонки».

Прямой ход метода прогонки сводится к исключению неизвестного x_{i-1} в каждом уравнении системы. Получаемая в результате прямого хода система содержит в каждом уравнении только два неизвестных x_i и x_{i+1}, и матрица ее – верхняя треугольная с двумя диагоналями. Запишем *i-ю* строку преобразованной двухдиагональной матрицы в виде

$$x_i = P_{i+1} x_{i+1} + Q_{i+1}. \quad (2.7)$$

Если система (2.6) приведена к виду (2.7), то обратный ход метода Гаусса очевиден. Однако использование общих алгоритмов прямого и обратного хода

нецелесообразно. Построим эффективную вычислительную схему, которая и составляет суть метода прогонки. Для этого, уменьшив в (2.7) индекс на единицу, запишем

$$x_{i-1} = P_i x_i + Q_i.$$

Подставляя x_{i-1} в систему (2.6), получим соотношение

$$a_i(P_i x_i + Q_i) - b_i x_i + c_i x_{i+1} = d_i,$$

из которого нетрудно получить

$$x_i = [b_i - a_i P_i]^{-1} c_i x_{i+1} + [b_i - a_i P_i]^{-1}[a_i Q_i - d_i].$$

Сравнивая это соотношение с (2.7), можем записать рекуррентные соотношения

$$P_{i+1} = c_i [b_i - a_i P_i]^{-1}, \quad Q_{i+1} = [a_i Q_i - d_i][b_i - a_i P_i]^{-1} \qquad (2.8)$$

для вычисления так называемых ПРОГОНОЧНЫХ КОЭФФИЦИЕНТОВ.

Подчеркнем, что последующие значения прогоночных коэффициентов P_{i+1}, Q_{i+1} вычисляются только по известным коэффициентам системы (2.6) и известным предыдущим значениям прогоночных коэффициентов P_i, Q_i.

Для начала прямого хода метода прогонки необходимо задать начальные (стартовые) значения прогоночных коэффициентов, например, P_i, Q_i. Отметим, что, вообще говоря, начальные значения коэффициентов P_i, Q_i в рассмотренной схеме вычислений не требуются, так как значения коэффициентов P_2, Q_2 вычисляются только через коэффициенты первого уравнения системы (2.6): при $i = 1$ из (2.6) получаем соотношение $-b_1 x_1 + c_1 x_2 = d_1$. Сравнивая это выражение с (2.7) при $i = 1$, получаем $P_2 = c_1 / b_1$; $Q_2 = -d_1 / b_1$, а значение x_1 в обратном ходе вычисляем по

соотношению $x_1 = P_2 x_2 + Q_2$. Использование P_i, Q_i в качестве начальных значений целесообразно по двум причинам: сохраняется однородность вычислительного алгоритма для всех *i = 2, 3, …,n;* упрощается обсуждение и доказательство условия корректности и устойчивости метода прогонки. Из того обстоятельства, что коэффициенты P_2, Q_2 не зависят от P_1, Q_1 (в соотношениях(2.8) при *i =1* коэффициенты P_i, Q_i умножаются на $a_1 = 0$), Обычно удобно положить $P_1 = Q_1 = 0.$ Для начала обратного хода метода прогонки необходимо для вычисления $x_i = P_{i+1} x_{i+1} + Q_{i+1}$ задать значение x_{n+1}. Так как $c_n = 0$, то из первого соотношения (2.8) вытекает, что $P_{n+1} = 0$ и, следовательно, можно задать любое значение для x_{n+1}. Обычно полагают $x_{n+1} = 0$, и тогда $x_n = Q_{n+1}$.

Метод прогонки устойчив, если $|P_i| \le 1.$ Метод прогонки корректен, если $b_i - a_i P_i \neq 0.$

Описание ввода/вывода

Входными данными для программы является таблица значений интерполируемой функции. Для ввода данных используется файл. В файле в текстовом виде должны быть в каждой строке записаны через пробел пары X и Y – вещественные числа.

Выходными данными является график построенного интерполяционного многочлена. Программа может вывести график в файл формата BMP,JPG,PNG, или вместе с таблицей с исходными данными вывести в файл формата PDF.

Схема алгоритма

Алгоритм расчета кубического сплайна включает в себя решение трехдиагональной системы линейных алгебраических уравнений. Поэтому перед тем, как перейти к рассмотрению указанного алгоритма, рассмотрим алгоритм метода прогонки – метода решения трехдиагональной системы линейных алгебраических уравнений, схема которого представлена на рисунке 1.

Данный алгоритм получает на вход массив параметров трехдиагональной системы `params`, элементы которого имеют поля `A`,`B`,`C`,`F`; и размер данного массива `count`. Результатом является массив `X`, в котором находятся решения системы.

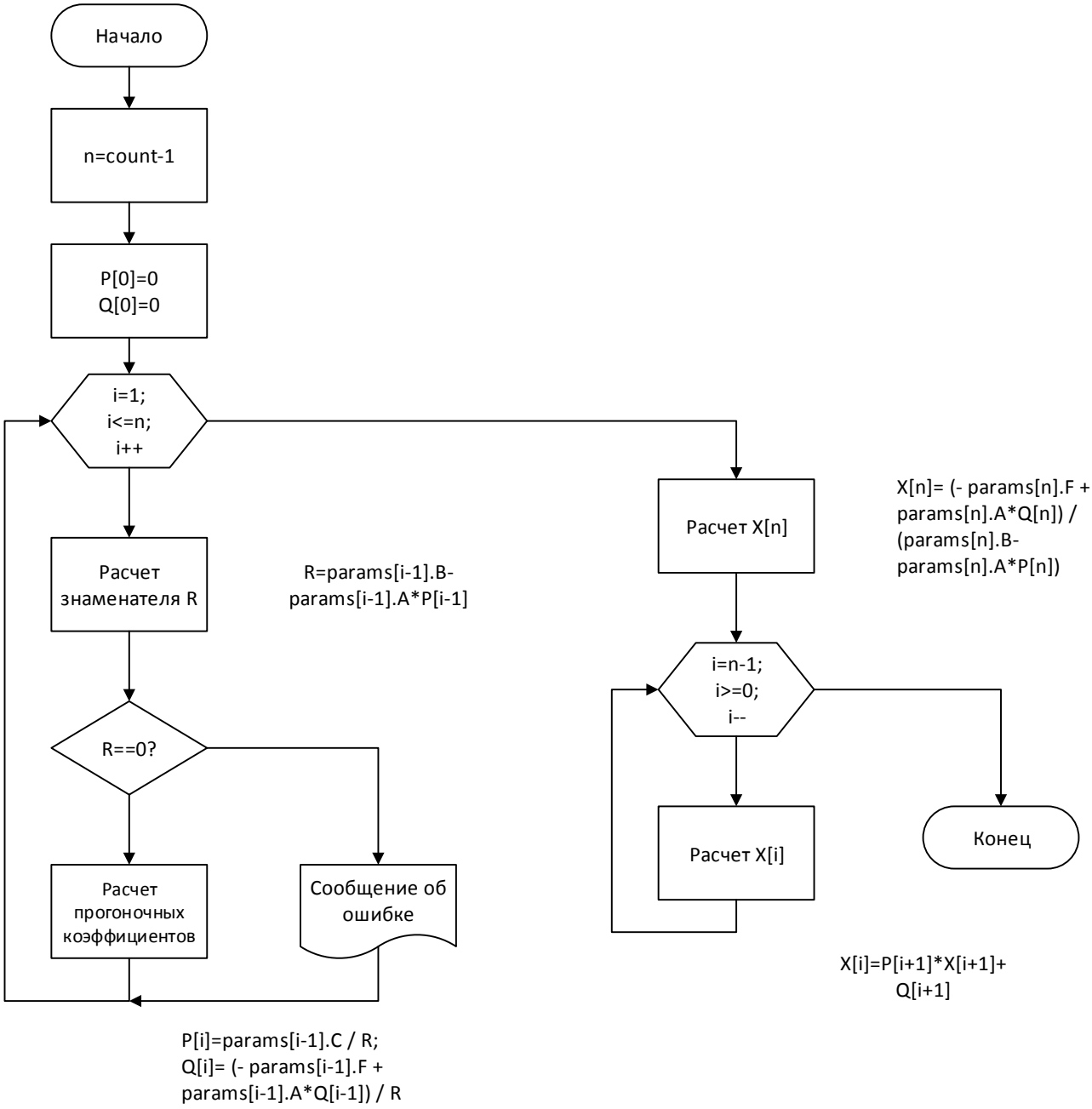

Рисунок 1 – Схема алгоритма метода прогонки

Перейдем к алгоритму расчета параметров кубического сплайна. Его схема представлена на рисунке 2. Алгоритм получает на вход массив структур – таблицу значений `F`, каждый элемент которой содержит поля `X` и `Y`. Результатом является массив записей – массив коэффициентов сплайнов `S`, который имеет поля `A`,`B`,`C`,`D`, где `A` – свободный член, `B`,`C`,`D` – коэффициенты при x первой, второй, третьей степени.

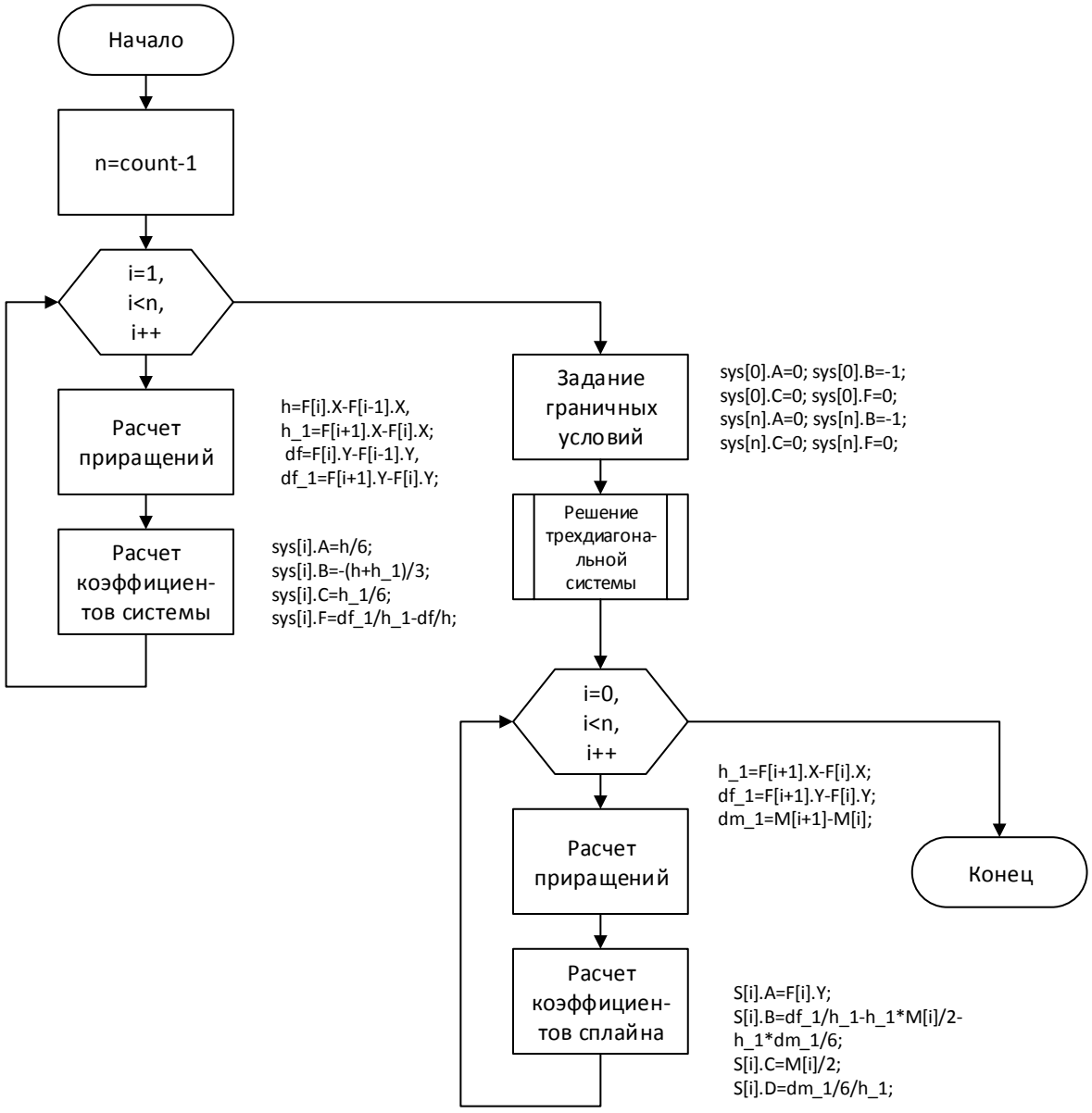

Рисунок 2 – Схема алгоритма расчета параметров кубического сплайна

Инструкция пользователю

Программа позволяет построить интерполяционный кубический сплайн. Чтобы узнать больше о кубических сплайнах, посетите раздел «Теоретические основы».

Программе задается список точек (X,Y) различными способами: из файла, ручным вводом, или аналитической функцией; после чего программа вычисляет параметры кубического сплайна и строит интерактивный график. График можно сохранить отдельно в файл формата JPG, PNG или вместе с исходными данными в формат PDF.

Окно программы содержит следующие зоны:

1. Строка меню – позволяет загрузить исходные данные, сохранить отчет и график, выйти из программы; обновить график; запустить справочную систему.

2. Панель инструментов – позволяет выполнить самые часто используемые операции: загрузить данные, сохранить отчет, обновить график.

3. Прикрепляемые модули – модули выполняют вспомогательные функции, позволяя вручную редактировать исходные данные и генерировать исходные данные с помощью аналитически заданной функции.

4. График – отображает сплайн-функцию, построенную по исходным данным.

Меню позволяет выполнять различные действия с программой.

1.Меню «Файл»

A.«Открыть файл…»

Позволяет ввести данные из файла. Для этого в открывшемся диалоге выберите имя файла, который хотели бы открыть. В файле в текстовом виде должны быть в каждой строке записаны через пробел пары X и Y – вещественные числа. После загрузки файла программа автоматически рассчитывает сплайн и строит график.

B.«Сохранить график…»

Сохраняет график в графических файлах JPG или PNG.

C.«Сохранить отчет…»

Сохраняет график вместе с исходными данными в файле PDF.

D.«Выход»

Выходит из программы.

2. Меню «Вид». Пункт «Обновить» позволяет построить заново график без повторного ввода исходных данных.

3.Меню «Справка». Пункт «Содержание» позволяет запустить данную справочную систему.

Панель инструментов позволяет выполнить самые часто используемые операции.

1.«Открыть файл»

Позволяет ввести данные из файла. Для этого в открывшемся диалоге выберите имя файла, который хотели бы открыть. В файле в текстовом виде

должны быть в каждой строке записаны через пробел пары X и Y – вещественные числа. После загрузки файла программа автоматически рассчитывает сплайн и строит график.

2.«Сохранить отчет»

Сохраняет график вместе с исходными данными в файле PDF.

3.«Обновить»

Позволяет построить заново график без повторного ввода исходных данных

Прикрепляемые модули выполняют вспомогательные функции.

1.Редактор данных. Позволяет редактировать исходные данные – просто выделите нужную ячейку двойным щелчком и введите нужное значение. Учтите, что при вводе нового значения X таблица может быть пересортирована по возрастанию X. Кнопка позволяет очистить данные. Кнопка добавляет новую строку в конец таблицы. Кнопка позволяет удалить выделенные строки (строки выделяются синим цветом; для выделения нескольких строк используется <Ctrl>). Кнопка выполняет обновление графика.

2.Демо-модуль. Данный модуль позволяет наглядно оценить точность интерполирования кубическим сплайном. Позволяет построить сплайн для известной аналитической функции. Задайте функцию, интервал, на котором строится сплайн и количество опорных точек для построения. После этого нажмите кнопку, и заданная функция и сплайн будут построены.

График является основным результатом работы программы.

Данный график является интерактивным – вы можете передвигать область просмотра, зажав и перемещая левую кнопку мыши; также можно изменять масштаб с помощью колесика мыши или зажав и перемещая правую кнопку мыши. Построенная интерполяционная функция строится синим цветом, демонстрационная аналитически заданная функция (если предоставлена) – зелёным, опорные точки, по которым строился сплайн, отмечаются серыми кружочками. Данная информация (легенда) также отображается внизу графика.

Программа позволяет сохранить график в графический файл или документ PDF.

Текст программы

Ниже представлен текст программы для заполнения матрицы, написанной на языке C++, в среде Qt Creator 2.6.0rc + GCC 4.7.2 с использованием библиотеки Qt.

splinesinterpol.h:

```
#ifndef SPLINESINTERPOL_H
#define SPLINESINTERPOL_H
#include <QList>
#include <QAbstractTableModel>

typedef struct {
       double A,B,C,F;
} ThreeDiagMatrix;

typedef struct {
    double A,B,C,D;
} CubicSpline;
typedef struct Point{
    double X,Y;

} Point;
bool operator <(const Point &a,const Point &other);
class FuncTable:public QList<Point>,public QAbstractTableModel{

    int rowCount ( const QModelIndex & parent) const ;
    int columnCount ( const QModelIndex & parent ) const;
    QVariant data ( const QModelIndex & index, int role ) const;
    QVariant headerData ( int section, Qt::Orientation orientation, int role )
const;
    Qt::ItemFlags flags ( const QModelIndex & index ) const;
    bool setData ( const QModelIndex & index, const QVariant & value, int role );
    bool insertRows ( int row, int count, const QModelIndex & parent = QModelIn-
dex() );
    bool removeRows ( int row, int count, const QModelIndex & parent = QModelIn-
dex() );
public:
    void clearModel();

};

#endif // SPLINESINTERPOL_H
```

splinesinterpol.cpp:

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <cmath>
#include <SplinesInterpol.h>
#include <QDebug>

double *TDMA(ThreeDiagMatrix *params,int count)
{
    int i;
    int n=count-1;
    double *P=new double[count],*Q=new double[count];

    P[0]=0; Q[0]=0;
    for (i=1;i<=n;i++){
        double R=params[i-1].B-params[i-1].A*P[i-1];
        if(fabs(R)<0.000000001)
            qDebug()<<"Error!";
```

```
        P[i]=params[i-1].C / R;
        Q[i]= (- params[i-1].F + params[i-1].A*Q[i-1]) / R;
    }

    double *X=new double[count];

    X[n]= (- params[n].F + params[n].A*Q[n]) / (params[n].B-params[n].A*P[n]);
    for (i=n-1;i>=0;i--){
        X[i]=P[i+1]*X[i+1]+Q[i+1];
    }

    delete P; delete Q;
    return X;
}
CubicSpline* getCubicSpline(const FuncTable &F, int count){
    ThreeDiagMatrix *sys=new ThreeDiagMatrix[count];
    double *M;
    int n=count-1,i;
    for(i=1;i<n;i++){
        double h=F[i].X-F[i-1].X, h_1=F[i+1].X-F[i].X;
        double df=F[i].Y-F[i-1].Y, df_1=F[i+1].Y-F[i].Y;

        sys[i].A=h/6;
        sys[i].B=-(h+h_1)/3;
        sys[i].C=h_1/6;
        sys[i].F=df_1/h_1-df/h;


    }

    sys[0].A=0; sys[0].B=-1; sys[0].C=0; sys[0].F=0;
    sys[n].A=0; sys[n].B=-1; sys[n].C=0; sys[n].F=0;
    M=TDMA(sys,count);

    CubicSpline *S=new CubicSpline[n];
    for (i=0;i<n;i++){
        double h_1=F[i+1].X-F[i].X;
        double df_1=F[i+1].Y-F[i].Y;
        double dm_1=M[i+1]-M[i];

        S[i].A=F[i].Y;
        S[i].B=df_1/h_1-h_1*M[i]/2-h_1*dm_1/6;
        S[i].C=M[i]/2;
        S[i].D=dm_1/6/h_1;
    }
    delete M; delete sys;
    return S;
}

int FuncTable::rowCount ( const QModelIndex & parent = QModelIndex() ) const {
    return this->size();
}

int FuncTable::columnCount ( const QModelIndex & parent = QModelIndex() ) const{
    return 2;
}

QVariant FuncTable::data ( const QModelIndex & index, int role = Qt::DisplayRole )
const{
    switch(role){
        case Qt::DisplayRole:
        case Qt::EditRole:
            if(index.row()<this->size()){
                if(index.column()==0)
                    return QVariant(this->at(index.row()).X);
                else if (index.column()==1);
```

```
                    return QVariant(this->at(index.row()).Y);
            } else
                return QVariant(QVariant::Invalid);
        case Qt::TextAlignmentRole:
            return QVariant(Qt::AlignRight|Qt::AlignVCenter);
        default:
            return QVariant(QVariant::Invalid);
    }


}

QVariant FuncTable::headerData ( int section, Qt::Orientation orientation, int
role = Qt::DisplayRole ) const{
    switch(role){
        case Qt::DisplayRole:
        case Qt::EditRole:
            if(orientation==Qt::Vertical){
                return QVariant(section);
            } else if (orientation==Qt::Horizontal){
                if (section==0)
                    return QVariant("X");
                else if(section==1)
                    return QVariant("Y");
                else
                    return QVariant(QVariant::Invalid);
            } else
                return QVariant(QVariant::Invalid);
        default:
            return QVariant(QVariant::Invalid);
    }


}

Qt::ItemFlags FuncTable::flags ( const QModelIndex & index ) const{
    return Qt::ItemIsSelectable|Qt::ItemIsEnabled|Qt::ItemIsEditable;
}

bool FuncTable::setData ( const QModelIndex & index, const QVariant & value, int
role = Qt::EditRole ){
    switch(role){
        case Qt::EditRole:
            if(index.row()<this->size()){
                if(index.column()==0){
                    (*this)[index.row()].X=value.toDouble();
                    emit this->dataChanged(index,index);
                    qSort(*this);
                    return true;
                }
                else if (index.column()==1){
                    (*this)[index.row()].Y=value.toDouble();
                    emit this->dataChanged(index,index);
                    //qSort(*this);
                    return true;
                }
                return false;
            } else
                return false;
        default:
            return false;
    }
}

bool FuncTable::insertRows ( int row, int count, const QModelIndex & parent ){
    if (count>0&&row<=this->size()){
```

```
        this->beginInsertRows(parent,row,row+count-1);
        while(count){
            Point t={0,0};
            *this<<t;
            --count;
        }
        endInsertRows();
        return true;
    } else return false;
}

bool FuncTable::removeRows ( int row, int count, const QModelIndex & parent ){
    if (count>0&&row<=this->size()){
        this->beginRemoveRows(parent,row,row+count-1);
        while(count){
            this->removeAt(row);
            --count;
        }
        endRemoveRows();
        return true;
    } else return false;
}

bool operator <(const Point &a, const Point &other){
    return (a.X)<(other.X);
}


void FuncTable::clearModel(){
    this->clear();
    this->reset();
}
```

mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <qwt.h>
#include <qwt_plot_curve.h>
#include <qwt_symbol.h>
#include <qwt_plot_magnifier.h>
#include <qwt_plot_panner.h>
#include <qwt_plot_zoomer.h>
#include <qwt_plot_grid.h>
#include <qwt_legend.h>
#include <QPrinter>
#include "SplinesInterpol.h"
#include <muParser.h>
#include <string>
const QString filetypes("Portable Network Graphics  (*.png *.PNG);;"\
                        "Windows Bitmap (*.bmp *.BMP);;"\
                        "Joint Photographic Experts Group   (*.jpg *.JPG *.jpeg
*.JPEG)");
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
```

```
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:

    void on_action_triggered();
    void on_action_2_triggered();
    void on_action_3_triggered();
    void on_action_4_triggered();

    void on_toolButton_3_clicked();

    void on_action_5_triggered();

    void on_toolButton_2_clicked();

    void on_pushButton_clicked();

    void on_toolButton_4_clicked();

    void on_action_6_triggered();

    void on_toolButton_clicked();

private:
    Ui::MainWindow *ui;

    QPrinter *printer;

    QwtPlotGrid grid;
    QwtLegend *legend;
    QwtPlotPanner *pan;
    QwtPlotMagnifier *magn;
    QwtPlotZoomer *zoom;
    QwtPlotCurve curv;
    QwtPlotCurve demo;
    QwtPlotCurve source;
    QwtSymbol *symbol1;

    FuncTable F;
    CubicSpline *S;
    double *X,*Y,*X1,*Y1,*demoX,*demoY;
    void ClearArrays();
    void ClearModel();
    void GenerateModel();
    void DrawPlot();
    void DrawDemo();
    mu::Parser parser;
    double parsX;
    void ReadData(QString fname);
};

#endif // MAINWINDOW_H
```

mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "SplinesInterpol.h"
#include <cmath>
#include <QMessageBox>
#include <QFileDialog>
#include <QRegExp>
#include <QPrintDialog>
#include <QTextDocument>
#include <QTextCursor>
```

```
#include <QTextTable>
#include <QProcess>
#include <qwt_plot_renderer.h>
#include <QImage>
#include <QPicture>
#include <QAbstractTextDocumentLayout>
#include <QUrl>
QString reports_head("");

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    printer=new QPrinter();
    printer->setOutputFormat(QPrinter::PdfFormat);
    //printer->setResolution(300);

    ui->qwtPlot->setAutoFillBackground( true );
    QPalette p = ui->qwtPlot->palette();
    p.setColor(QPalette::Window, QColor(Qt::white));
    ui->qwtPlot->setPalette(p);

    ui->qwtPlot->canvas()->setPaintAttribute(QwtPlotCanvas::Opaque);

    QwtLegend *legend = new QwtLegend;
    legend->setFrameStyle(QFrame::Box|QFrame::Sunken);
    ui->qwtPlot->insertLegend(legend, QwtPlot::BottomLegend);

    QRect r=ui->dockWidget_2->geometry();
    r.setWidth(155);
    ui->dockWidget_2->setGeometry(r);

    magn=new QwtPlotMagnifier(ui->qwtPlot->canvas());
    pan=new QwtPlotPanner(ui->qwtPlot->canvas());
    //zoom=new QwtPlotZoomer(ui->qwtPlot->canvas());

    curv.setTitle("Сплайн-функция");
    //curv.setLegendAttribute(QwtPlotCurve::LegendShowLine);
    curv.setPen(QPen(QColor("Blue")));
    curv.attach(ui->qwtPlot);

    demo.setPen(QPen(QColor("Green")));
    //demo.setLegendAttribute(QwtPlotCurve::LegendShowLine);
    //demo.attach(ui->qwtPlot);

    symbol1=new QwtSymbol();
    symbol1->setStyle(QwtSymbol::Ellipse);
    symbol1->setPen(QColor(Qt::black));
    symbol1->setSize(4);

    source.setTitle("Опорные точки");
    source.setLegendAttribute(QwtPlotCurve::LegendShowSymbol);
    source.setPen(QPen(QColor("Red")));
    source.attach(ui->qwtPlot);
    source.setStyle(QwtPlotCurve::NoCurve);
    source.setSymbol(symbol1);

    ui->qwtPlot->canvas()->setLineWidth(0);

    grid.attach(ui->qwtPlot);

    X=NULL;Y=NULL;X1=NULL;Y1=NULL;
    demoX=NULL;demoY=NULL;
    S=NULL;
```

```
    //F=new FuncTable();
    ui->tableView->setModel(&F);

    parser.DefineVar("x",&parsX);
}

void MainWindow::ReadData(QString fname){
    QFile file(fname);
    if (!file.exists()||!file.open(QIODevice::ReadOnly | QIODevice::Text))
        return;
    ClearModel();
    QTextStream in(&file);
    while (!in.atEnd()) {
        Point t;
        in>>t.X>>t.Y;
        F<<t;
    }
    qSort(F);
    file.close();
}

MainWindow::~MainWindow()
{
    delete magn;
    delete pan;
    //delete zoom;
    delete ui;
    delete symbol1;
    ClearArrays();
    ClearModel();
}

void MainWindow::ClearModel(){
    F.clearModel();
}

void MainWindow::ClearArrays(){
    curv.detach();
    demo.detach();
    source.detach();
    curv.setSamples(NULL,NULL,0);
    if (X!=NULL){
        delete X;
        X=NULL;
    }

    if (Y!=NULL){
        delete Y;
        Y=NULL;
    }

    demo.setSamples(NULL,NULL,0);
    if (demoX!=NULL){
        delete demoX;
        demoX=NULL;
    }

    if (demoY!=NULL){
        delete demoY;
        demoY=NULL;
    }

    source.setSamples(NULL,NULL,0);
    if (X1!=NULL){
        delete X1;
        X1=NULL;
```

```
    }

    if (Y1!=NULL){
        delete Y1;
        Y1=NULL;
    }

    if (S!=NULL){
        delete S;
        S=NULL;
    }
    ui->qwtPlot->replot();
}

void MainWindow::on_action_triggered(){
    //QPixmap pix=QPixmap::grabWidget(ui->qwtPlot);

    QString filename=QFileDialog::getSaveFileName(this,"Сохранить гра-
фик...",QString(),filetypes);
    if(filename!=""){
        QImage *img=new QImage(3.5*ui->qwtPlot->canvas()->width(),
                               3.5*ui->qwtPlot->canvas()->height(),
                               QImage::Format_RGB32);
        img->setDotsPerMeterX(3.5*ui->qwtPlot->canvas()->logicalDpiX()*100/2.54);
        img->setDotsPerMeterY(3.5*ui->qwtPlot->canvas()->logicalDpiY()*100/2.54);
        img->fill(Qt::white);
        QwtPlotRenderer render;
        render.renderTo(ui->qwtPlot,*img);
        img->save(filename);
        delete img;
        //pix.save(filename);
    }
}

void MainWindow::DrawPlot(){
    int n=F.size();
    double dx=(F.last().X-F.first().X)/ui->qwtPlot->canvas()->width(); int i=0;
    qDebug()<<dx;

    S=getCubicSpline(F,n);
    X=new double[int((F.last().X-F.first().X)/dx)+1];
    Y=new double[int((F.last().X-F.first().X)/dx)+1];
    X1=new double[n];
    Y1=new double[n];

    i=0;
    X[0]=F.at(0).X;Y[0]=F.at(0).Y;
    for(int len=1;len<=int((F.last().X-F.first().X)/dx);++len){
        X[len]=X[len-1]+dx;
        Y[len]=S[i].A+(S[i].B+(S[i].C+S[i].D*(X[len]-F.at(i).X))*(X[len]-
F.at(i).X))*(X[len]-F.at(i).X);
        //qDebug()<<Y[len];
        if(i<n-1&&X[len]>=F.at(i+1).X){
            i++;
        }
    }

    curv.setRawSamples(X,Y,(F.last().X-F.first().X)/dx+1);
    for (int i=0;i<n;i++){
        X1[i]=F.at(i).X;
        Y1[i]=F.at(i).Y;
    }
    source.setRawSamples(X1,Y1,n);
    curv.attach(ui->qwtPlot);
    source.attach(ui->qwtPlot);
    ui->qwtPlot->setAxisAutoScale(QwtPlot::xBottom);
```

```
    ui->qwtPlot->setAxisAutoScale(QwtPlot::yLeft);
    ui->qwtPlot->updateAxes();
    ui->qwtPlot->replot();
}

void MainWindow::GenerateModel(){
    QString filename=QFileDialog::getOpenFileName(this,"Открытие файла...");
    if(filename=="")
        return;
    ReadData(filename);
}

void MainWindow::DrawDemo(){
    double dx=(F.last().X-F.first().X)/ui->qwtPlot->canvas()->width();
    int size=int((F.last().X-F.first().X)/dx)+1;
    demoX=new double[size];
    demoY=new double[size];

    parsX=F.at(0).X;
    demoX[0]=F.at(0).X;demoY[0]=F.at(0).Y;
    for(int i=1;i<size;++i){
        parsX+=dx;
        try{
            demoX[i]=parsX;
            demoY[i]=parser.Eval();
        }catch (mu::ParserError &e){
            --i;
        }
    }

    demo.setTitle(ui->lineEdit->text());
    demo.setRawSamples(demoX,demoY,size);
    demo.attach(ui->qwtPlot);

}

void MainWindow::on_action_2_triggered(){

    QString filename=QFileDialog::getOpenFileName(this,"Открытие файла...");
    if(filename=="")
        return;
    ReadData(filename);

    ClearArrays();
    DrawPlot();

};

void MainWindow::on_action_3_triggered(){
    bool draw_demo=false;
    if(demoX!=NULL&&demoY!=NULL)
        draw_demo=true;
    ClearArrays();
    //qSort(F);
    if(draw_demo)
        DrawDemo();
    DrawPlot();
}

void MainWindow::on_action_4_triggered(){
    close();
}

void MainWindow::on_toolButton_3_clicked()
{
    ui->action_3->activate(QAction::Trigger);
```

```
}

void MainWindow::on_action_5_triggered()
{
    QString name=QFileDialog::getSaveFileName(this,"Сохранение отчета","","Файлы
PDF (*.pdf)");
    if (name!=""){
        printer->setOutputFileName(name);
        //printer->setPageMargins(0,10,0,10,QPrinter::Millimeter);

        QTextDocument doc;
        doc.documentLayout()->setPaintDevice(printer);
        doc.setDocumentMargin(0);
        doc.setPageSize(printer->pageRect().size());

        QTextCursor main_curs(&doc),curs;
        QTextTable *table;
        QTextTableFormat tform;
        QTextCharFormat cform;
        QTextBlockFormat bform;
        QTextImageFormat iform;

       //qDebug()<<printer.resolution();

        bform.setAlignment(Qt::AlignHCenter);
        cform.setFontCapitalization(QFont::AllUppercase);
        cform.setFontWeight(QFont::Bold);
        cform.setFontFamily("Arial");
        cform.setFontPointSize(20);
        main_curs.insertBlock(bform,cform);
        main_curs.insertText("входные данные\n\n");

        main_curs.movePosition(QTextCursor::End);

        tform.setAlignment(Qt::AlignHCenter);
        tform.setCellSpacing(0);
        tform.setBorderStyle(QTextFrameFormat::BorderStyle_Solid);
        table=main_curs.insertTable(F.size()+1,3);
        table->setFormat(tform);

        curs=table->cellAt(0,1).firstCursorPosition();
        bform.setAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        curs.setBlockFormat(bform);
        curs.insertText(QString("X"));

        curs=table->cellAt(0,2).firstCursorPosition();
        bform.setAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        curs.setBlockFormat(bform);
        curs.insertText(QString("Y"));

        for(int i=0;i<F.size();++i){
            QString msg;


            curs=table->cellAt(i+1,0).firstCursorPosition();
            bform.setAlignment(Qt::AlignRight|Qt::AlignVCenter);
            curs.setBlockFormat(bform);
            curs.insertText(msg.sprintf("%d",i));

            curs=table->cellAt(i+1,1).firstCursorPosition();
            bform.setAlignment(Qt::AlignRight|Qt::AlignVCenter);
            curs.setBlockFormat(bform);
            curs.insertText(msg.sprintf("%f",F[i].X));

            curs=table->cellAt(i+1,2).firstCursorPosition();
            bform.setAlignment(Qt::AlignRight|Qt::AlignVCenter);
```

```
            curs.setBlockFormat(bform);
            curs.insertText(msg.sprintf("%f",F[i].Y));

        }
        main_curs.movePosition(QTextCursor::End);

        bform.setAlignment(Qt::AlignHCenter|Qt::AlignTop);
        bform.setPageBreakPolicy(QTextFormat::PageBreak_AlwaysBefore);
        cform.setFontCapitalization(QFont::AllUppercase);
        cform.setFontWeight(QFont::Bold);
        cform.setFontFamily("Arial");
        cform.setFontPointSize(20);
        main_curs.insertBlock(bform,cform);
        main_curs.insertText("График результата");

        main_curs.movePosition(QTextCursor::End);

        bform.setPageBreakPolicy(QTextFormat::PageBreak_Auto);
        bform.setAlignment(Qt::AlignLeft);
        QImage *img=new QImage(3.5*doc.pageSize().width(),
                               0.8*3.5*doc.pageSize().height(),
                               QImage::Format_RGB32);
        img->setDotsPerMeterX(printer->logicalDpiX()*350/2.54);
        img->setDotsPerMeterY(printer->logicalDpiY()*350/2.54);
        qDebug()<<img->logicalDpiX()<<img->physicalDpiX();
        img->fill(Qt::white);
        QwtPlotRenderer rend;
        rend.renderTo(ui->qwtPlot,*img);
        doc.addResource(QTextDocument::ImageResource,QUrl("abracadabra"),QVari-
ant(*img));

        iform.setWidth(doc.pageSize().width());
        iform.setHeight(0.8*doc.pageSize().height());
        iform.setName("abracadabra");
        main_curs.insertBlock(bform);
        main_curs.insertImage(iform);

        doc.print(printer);
        delete img;

    }
}

void MainWindow::on_toolButton_2_clicked()
{
    F.insertRow(F.size());
    qDebug()<<F.size();
}


void MainWindow::on_pushButton_clicked()
{
    parser.SetExpr(ui->lineEdit->text().toAscii().constData());
    int n=ui->spinBox->value();
    double  a=ui->doubleSpinBox->value(),
            b=ui->doubleSpinBox_2->value(),
            h=(b-a)/(n-1);
    ClearArrays();
    ClearModel();

    Point t;
    try{
        parsX=a;t.X=a;t.Y=parser.Eval();
    } catch (mu::Parser::exception_type &e){
        QMessageBox msg;
        msg.setText(e.GetMsg().c_str());
```

```
            msg.exec();
            return;
        }

        F<<t;
        for(int i=1;i<n;i++){
            parsX+=h;
            try{
                t.X=parsX;t.Y=parser.Eval();
                F<<t;
            } catch (...){
                --i;
            }
        };

        DrawDemo();
        DrawPlot();

}

void MainWindow::on_toolButton_4_clicked()
{
    ClearArrays();
    ClearModel();
}

void MainWindow::on_action_6_triggered()
{
    QProcess *process = new QProcess;
    QStringList args;
    args << QLatin1String("-collectionFile")
        << QLatin1String("C:/Users/Wolf/Documents/NumMat/NumMat-1.0.qhc")
        << QLatin1String("-enableRemoteControl");
    process->start(QLatin1String("assistant"), args);
}

void MainWindow::on_toolButton_clicked()
{
    QModelIndexList list=ui->tableView->selectionModel()->selectedRows();
    for(int i=list.size()-1;i>=0;--i){
        F.removeRow(list[i].row());
    }
}
```

Тестовый пример

На рисунке 3 представлен пример работы программы построения интерполяционного кубического сплайна для аналитической функции $x^3 - x^2$ на отрезке [-10;10] с четырьмя опорными точками.

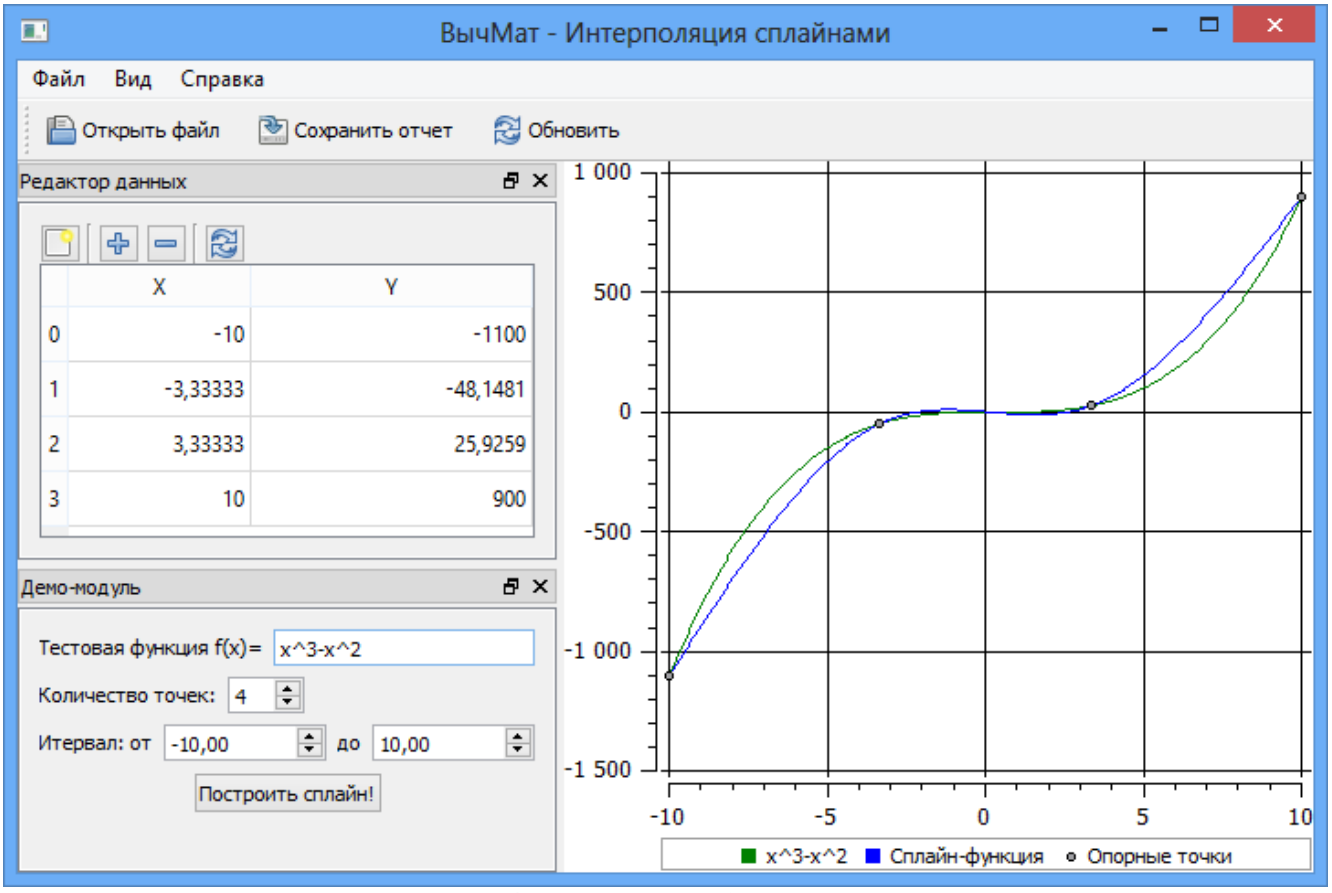

Рисунок 4 — Пример работы программы

Проверка результата

Построенный программой график является гладкой кривой, проходящей через все заданные точки. Кроме того, как видно на рисунках 4-6, с увеличением количества опорных точек интерполирующая кривая стремится к графику исходной функции. Так что можно сделать вывод, что программа работает корректно.

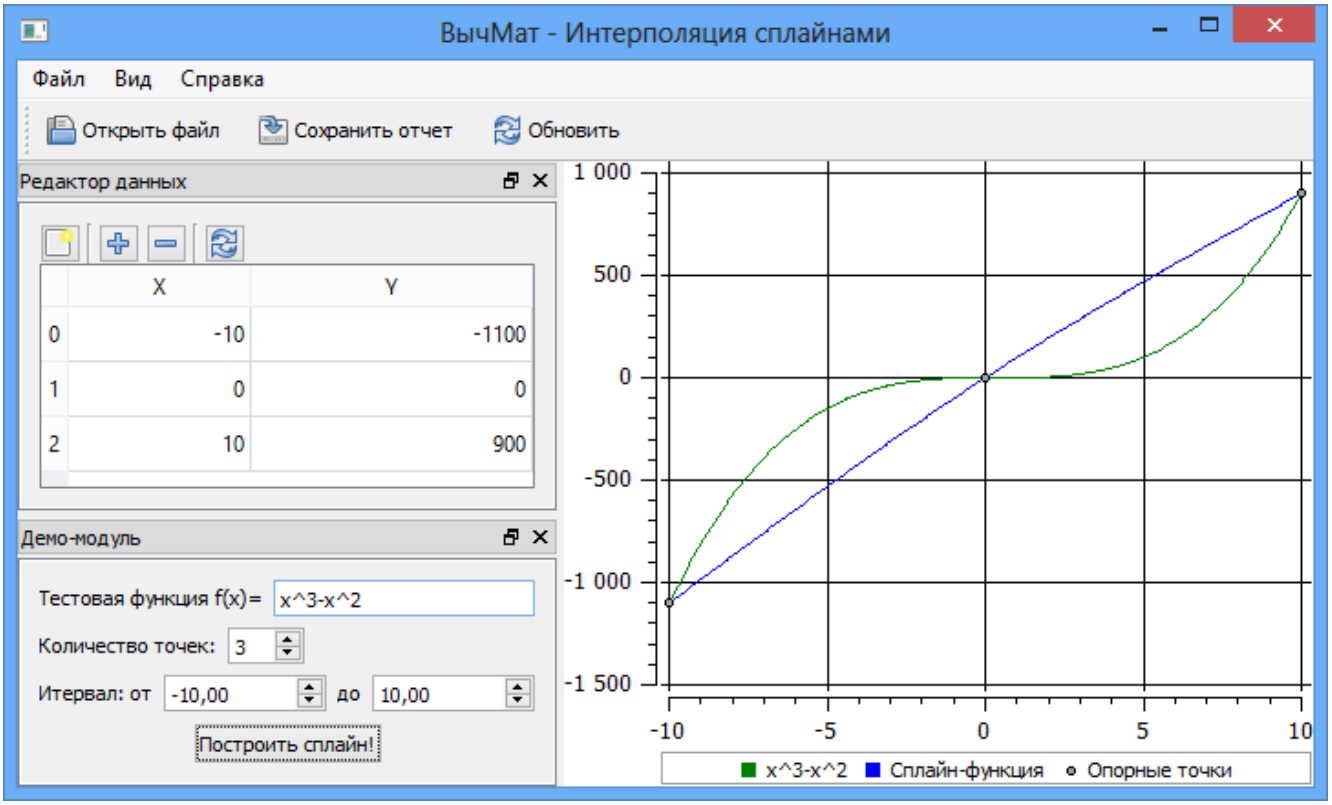

Рисунок 4 – Результат интерполяции по 3 точкам

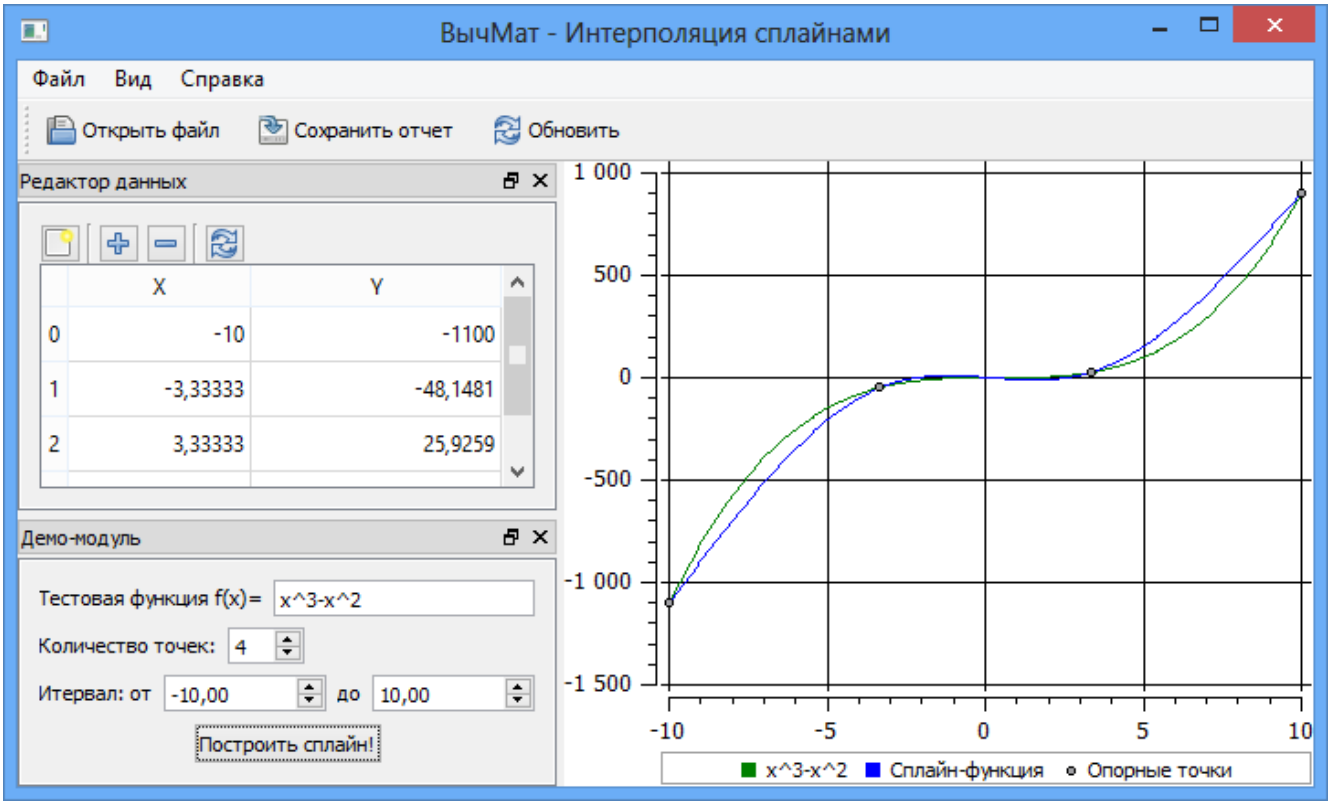

Рисунок 5 - Результат интерполяции по 4 точкам

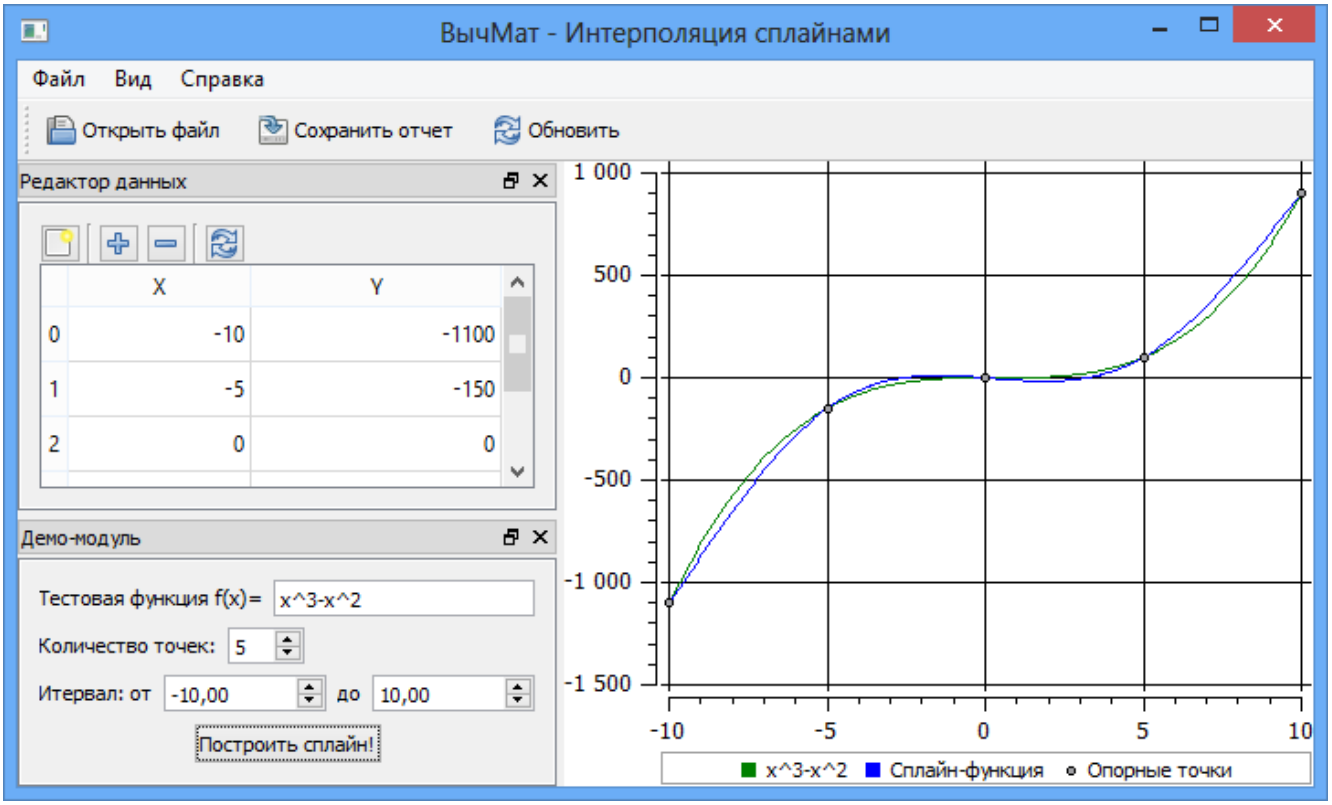

Рисунок 6 - Результат интерполяции по 5 точкам

Список использованной литературы

1. Киреев В.И., Пантелеев А.В. Численные методы в примерах и задачах - 3-е изд., стер. — М.: Высш. шк. , 2008.
2. Бахвалов Н.С. Численные методы : учеб.пособие для вузов - 5-е изд. — М. : БИНОМ.Лаборатория Знаний, 2007
3. Ю. Рыжиков. «Вычислительные методы» - изд. BHV, 2007 г.
4. Костомаров, Д.П. Вводные лекции по численным методам : учеб.пособие для вузов - М. : Логос, 2006 .