МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Системний аналіз та управління»

Оцінка
_____
голова комісії
доц. каф.САіУ
_____ /Марченко Н.А./
« » _____ 2013 р.

КУРСОВА РОБОТА

Дисципліна: «Чисельні методи»

Тема: «Чисельна інтерполяція та апроксимація»

Варіант Б.9

Виконавець: ст. гр. ИФ-50^{б} Стрігін О.І.

“____”_____20___ р.

Керівник роботи: ст. викладач каф. САіУ Сідоренко А.Ю.

“____”_____20___ р.

Харків 2013

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Системний аналіз та управління»

Студент Стрігін О.І. Група ІФ-50б

ЗАВДАННЯ
на науково-дослідну курсову роботу

Дисципліна: "Чисельні методи"

Тема: : «Чисельна інтерполяція та апроксимація»

Короткий зміст роботи:
а) реферативна частина
Огляд літературних джерел з чисельних методів.

б) теоретична частина
Розробка методики інтерполювання та апроксимація функції.

в) експериментальна частина
Розробка програмного забезпечення, яке реалізує методи Лагранжа, Ньютона для інтерполювання функції та апроксимації МНК.

г) чисельні експерименти
Спрощення функції. Пошук емпіричної функції для апроксимації МНК. Аналіз та залежність від параметрів функцій та величин.

Дата видачі завдання: 10.10.2012 Термін захисту: 21.01.2013

Керівник курсової роботи:_____/ ст. викладач каф. САіУ
Сідоренко А.Ю. /

Содержание

Вступление

Многим приходилось сталкиваться с научными и инженерными расчетами, оперировать наборами значений, полученных опытным путем или же случайным образом. На основании полученных данных обычно стоит построить функцию, на которую с высокой точностью могли бы попадать другие значения однородные тем, по которым составлялась функция. Такая задача называется аппроксимация. Интерполяция функции, в свою очередь, является некой разновидностью аппроксимацией, отличается тем, что проводится непосредственно по всем имеющимся точкам. Если некоторая функция слишком сложна для производительных вычислений, можно попытаться вычислить её значение в нескольких точках, а по ним построить, то есть интерполировать, более простую функцию. Разумеется, использование упрощенной функции не позволяет получить такие же точные результаты , какие давала бы первоначальная функция.

В данной выполненной мною работе, приводится описание методов интерполирования и аппроксимации данной функции, рассматриваются вопросы алгоритмизации задачи, составление программы на языке C# и реализации вычислений значений полиномов и функций.

1. **Постановка задачи**

Для функции $f\ x\ = 16^{\sin x^{\cos x}} + \frac{6}{4^{sin^2\ x-\frac{\pi}{4}}} - 4 \cdot \beta$ где параметр β=0,5;1;1,5;2, на интервале (0,2] построить интерполяционные полиномы Лагранжа и Ньютона с числом узлов $10 \leq n \leq 20$. Провести аппроксимацию данной функцией, подобранной для метода наименьших квадратов. Полученные значения сравнить со значением функции f(x). Результаты оформить графически.

2. Обзор существующих методов решения задачи

2.1. Интерполяция

Интерполяция – численная процедура, при которой определенная функция проводиться непосредственно через совокупность заданных узлов. При этом в каждом из узлов с номером n значения аргумента и функции точно равны заданным x_n и f_n. Интерполяция применяется для отыскания значения функции в точках, отличных от узлов. Построение такой функции – интерполирование. Наиболее распространена алгебраическая интерполяция, когда в качестве интерполирующей функции используется многочлен соответствующей степени.

2.2. Метод Лагранжа

Пусть имеется таблица из (N+1) узлов { x_n, f_n }, n=0,1,2,...,N. Обычно их удобно нумеровать, начиная с n=0, при этом $x_0 = a$ b $x_N = b$, где a, b – границы аргументного интервала.

Алгебраическое интерполирования функции f(x), заданной своими значениями $f\ x_0$, $f\ x_1$,... $f\ x_N$ в точках $x_0, x_1, \dots x_N$ на отрезке [a, b], состоит в приближенной замене этой функции на данном отрезке многочленом $P_N\ x$ степени N. Этот многочлен в узлах интерполяции принимает те же значения, что и функция f(x)

$P_N\ x_n = f\ x_n$, *n=0,1,2…N.*

Существует единственный интерполяционный многочлен N-й степени, удовлетворяющий данным условиям.

Решение задачи алгебраической интерполяции обеспечивает интерполяционный многочлен Лагранжа:

$$P_N\ x\ = f\ x_0\ \cdot \frac{x - x_1\ \ x - x_2\ \ \dots\ \ x - x_N}{(x_0 - x_1)(x_0 - x_2)\dots(x_0 - x_N)} + f\ x_1$$
$$\cdot \frac{x - x_0\ \ x - x_2\ \ \dots\ \ x - x_N}{(x_1 - x_0)(x_1 - x_2)\dots(x_1 - x_N)} + \dots\dots\dots\dots\dots + f\ x_N$$
$$\cdot \frac{x - x_0\ \ x - x_1\ \ \dots(x - x_{N-1})}{(x_N - x_0)(x_N - x_1)\dots(x_N - x_{N-1})}$$

В точках отрезка [a, b],которые отличны от узлов интерполяции, разность $R\ x\ = f\ x\ - P_N(x)$ в общем случае не равна нулю. Эту разность представляет погрешность метода, которую называют остаточным членом интерполяции.

Оценка погрешности интерполяции в текущей точке x определяется выражением:

$$R\ x\ = \frac{M}{n+1\ !} \cdot\ x - x_1\ \ x - x_2\ \ \dots(x - x_N),\ M = \max_x |f^{n+1}\ x\ |.$$

2.3. Метод Ньютона

На практике таблица может дополняться новыми данными (в начале или в конце). В таких случаях при обработке данных удобнее пользоваться интерполяционными алгоритмами, отличными от полинома Лагранжа. Вычисление значений функции для аргументов, лежащих в начале таблицы, удобно проводить, пользуясь интерполяционной формулой Ньютона. Для случая равноотстоящих узлов $x_n = x_0 + nh$, n=0,1,2,…N с шагом h интерполяционный полином Ньютона имеет вид:

$P_N\ x\ = a_o + a_1\ x - x_o\ + a_2\ x - x_o\ \ x - x_1\ + \dots +\ a_N\ x - x_o\ \ x - x_1\ \dots(x - x_{N-1})$.

Коэффициенты $a_0, a_1 \dots, a_N$ определяются из условий

$P_N\ x_n\ = f_n, n = 0,1,2,\dots,N,$

что дает:

$a_0 = f_0,$

$a_1 = \frac{1}{h}\ f_1 - f_0\ = \frac{\Delta f_0}{h},$

$a_2 = \frac{1}{2 \cdot h^2}\ f_2 - 2f_1 + f_0\ = \frac{\Delta^2 f_0}{2 \cdot h^2},$

……

$$a_k = \frac{\Delta^k f_0}{k!\, h^k}$$

Здесь $\Delta^k f_0$- конечная разность k-го порядка, определяемая рекуррентно:

$\Delta f_0 = f_1 - f_0$,

$\Delta^2 f_0 = \Delta f_1 - \Delta f_0$,

……

$\Delta^k f_0 = \Delta^{k-1} f_1 - \Delta^{k-1} f_0$

$$P_N\ x\ = f_0 + \frac{\Delta f_0}{h}\ x - x_0\ + \frac{\Delta^2 f_0}{2h^2}\ x - x_0\ \ x - x_1\ + \cdots\cdots\cdots\cdots\cdots + \frac{\Delta^N f_0}{N!\, h^N}\ x - x_0\ \ x - x_1\ \cdots (x - x_{N-1})$$

Эта формула называется первой интерполяционной формулой Ньютона. При определнии значений функции f в точке с абсциссой x по первой интерполяционной формуле Ньютона удобно принимать в качестве x_0 ближайший к этой абсциссе узел интерполирования.

Для интерполирования в конце таблицы интерполяционный многочлен удобно представить в следующем виде:

$$P_N\ x\ = f_N + \frac{\Delta f_{N-1}}{h}\ x - x_N\ + \frac{\Delta^2 f_{N-2}}{2h^2}\ x - x_N\ \ x - x_{N-1}\ + \cdots\cdots\cdots\cdots\cdots + \frac{\Delta^N f_0}{N!\, h^N}\ x - x_N\ \ x - x_{N-1}\ \cdots (x - x_{1)}$$

Эта формула называется второй интерполяционной формулой Ньютона.

2.4. Аппроксимация. МНК.

Пусть в результате измерений в процессе опыта получено табличное задание некоторой функции $f(x)$. Конечно, можно найти формулу, выражающую эту зависимость аналитически, применив метод интерполяции. Однако совпадение значений полученного аналитического задания функции в узлах интерполяции с имеющимися эмпирическими данными часто может вовсе не означать совпадение характеров поведения исходной и интерполирующей функции на всем интервале наблюдения. При изучении количественных зависимостей различных показателей имеется некоторая их вариабельность.

Аппроксимация – приближенное выражение каких-либо математических объектов (например, чисел или функций) через другие более

простые и более удобные в использовании. Задача аппроксимации функции одной переменной обязательно учитывает характер поведения исходной функции на всем интервале наблюдений.

Наиболее распространенным методом аппроксимации экспериментальных данных является метод наименьших квадратов.

Метод наименьших квадратов состоит в том, чтобы подобрать функцию $\varphi(x)$, отклонение которой от опытных данных было бы минимальным среди всех функций данного вида. При этом возможно *точечное* аппроксимирование функции на отрезке, если исходные данные записаны в таблицу, и *интегральное* аппроксимирование функции на отрезке, если требуется приблизить заданную аналитическим выражением функцию некоторым многочленом.

Мерой отклонения аппроксимирующей функции $\varphi(x)$ от опытных данных $f(x_i)$ является величина $(\varphi(x_i) - f(x_i))^2$, а мерой общей ошибки S является сумма мер отклонений для всех опытов, то есть

$$S = \sum_{i=0}^{n} (\varphi(x_i) - f(x_i))^2 .$$

Квадраты отклонений рассматривают для того, чтобы избежать взаимного уничтожения отдельных слагаемых большой величины и разных знаков.

Метод определения констант, входящих в формулу $\varphi(x)$ путём минимизации функции S называется методом наименьших квадратов. Метод наименьших квадратов нацелен на уменьшение самых больших отклонений. Формула $\varphi(x)$, служащая для аналитического представления опытных данных, называется эмпирической.

Вид функции $\varphi(x)$ установлен или из теоретических соображений, или на основании характера расположения на координатной плоскости точек, соответствующих табличному значению функции. Если вид эмпирической формулы выбран, то возникает задача определения параметров, входящих в эту формулу, то есть

$\varphi = \varphi(x, a_0, a_1, ..., a_m)$, где $a_0, a_1, ..., a_m$ - неизвестные постоянные. Количество этих констант выбирается обычно меньше числа точек в таблице.

Наилучшими значениями параметров будут те, для которых сумма мер отклонений принимает наименьшее значение.

$$S = S(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} (\varphi(x_i, a_0, a_1, \ldots, a_m) - y_i)^2$$

Если рассматриваются S как функция от $a_0, a_1, ..., a_m$, то очевидно, что $S \geq 0$ и наименьшее значение этой функции существует. Это возможно

только в точке минимума функции S. Используя необходимые условия экстремума, получаем систему уравнений для определения параметров $a_0, a_1, ..., a_m$.

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 0 & 2\sum\limits_{i=0}^{n}(\varphi(x_i, a_0, a_1, \ldots, a_m) - y_i)\dfrac{\partial \varphi}{\partial a_0} = 0 \\ \dfrac{\partial S}{da_1} = 0 & 2\sum\limits_{i=0}^{n}(\varphi(x_i, a_0, a_1, \ldots, a_m) - y_i)\dfrac{\partial \varphi}{\partial a_1} = 0 \\ \ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \dfrac{\partial S}{da_m} = 0 & 2\sum\limits_{i=0}^{n}(\varphi(x_i, a_0, a_1, \ldots, a_m) - y_i)\dfrac{\partial \varphi}{\partial a_m} = 0 \end{cases}$$

Если эмпирическая формула выбрана в виде показательной функции $\varphi(x) = a_0 e^{a_1 x}$, в виде степенной $\varphi(x) = a_0 x^{a_1}$, или в виде логарифмической $\varphi(x) = a_0 \ln(a_1 x)$, то искомыми являются только два параметра.

Формула, полученная методом наименьших квадратов, называется уравнением регрессии, а соответствующая кривая – линией регрессии. Если в результате вычислений получено уравнение прямой линии, то регрессия называется линейной, в противном случае – нелинейной.

Исходные данные могут носить самый разнообразный характер и относиться к различным отраслям науки или техники, например:

1) зависимость продолжительности службы электрических ламп (y) от поданного на них напряжения (x);

2) зависимость пробивного напряжения конденсаторов (y) от температуры окружающей среды (x);

3) зависимость предела прочности стали (y) от содержания углерода (x);

4) зависимость показателей безработицы (y) и инфляции (x);

5) зависимость цен товара (y) от спроса (x) на этот товар;

6) зависимость частного потребления (y) от располагаемого дохода (x);

и другие зависимости.

Если приближающая функция $\varphi(x)$ выбрана в виде показательной функции $\varphi(x) = a_0 e^{a_1 x}$, то выражение можно светсти к линейному, логарифмируя левую и правую части равенства:

$$\ln \varphi(x) = \ln a_0 e^{a_1 x} \text{ или } \ln \varphi(x) = \ln a_0 + a_1 x.$$

После введения обозначений: $\ln \varphi(x) = \psi(x)$, $\ln a_0 = b_0$; $a_1 = b_1$, функция $\psi(x)$ записывается как линейная по аргументу x:

$$\psi(x) = b_0 + b_1 x$$

Сумма отклонений определяется формулой:

$$S = S(b_0, b_1) = \sum_{i=0}^{n} (b_0 + b_1 x_i - y_i)^2,$$

а коэффициенты b_0 и b_1 находятся из решения системы:

$$\begin{cases} 2\sum_{i=0}^{n} (b_0 + b_1 x_i - y_i) = 0 \\ 2\sum_{i=0}^{n} (b_0 + b_1 x_i - y_i) \cdot x_i = 0 \end{cases}$$

после чего осуществляется обратный переход к параметрам a_0 и a_1.

3. Описание программы

3.1. Руководство пользователю

Интерфейс программы приведен на рис. 3.1:

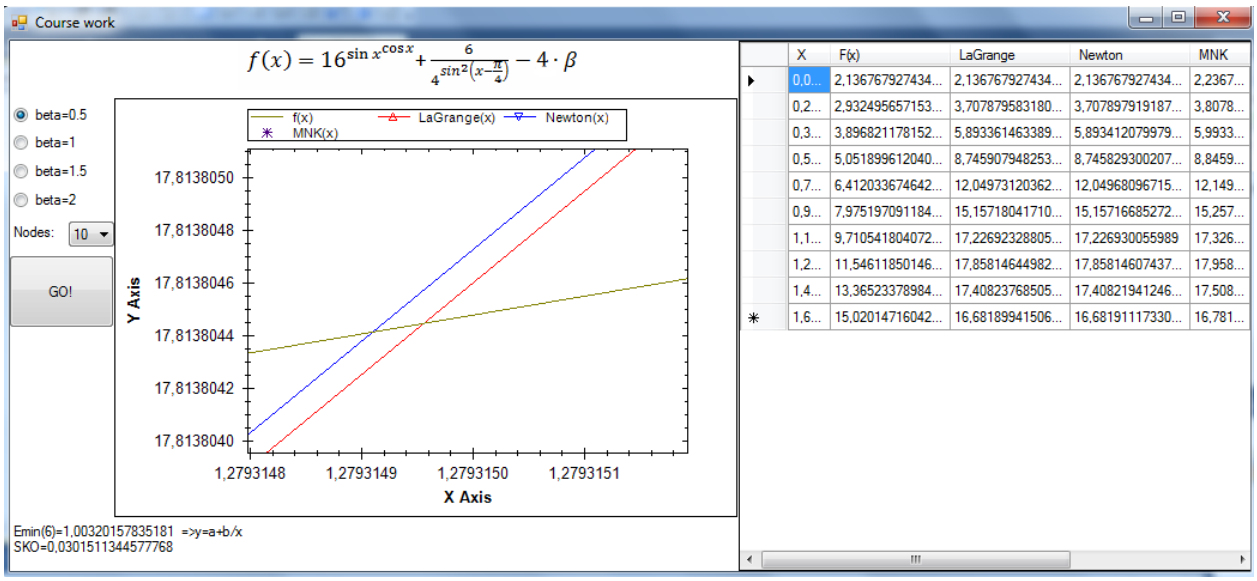

Рисунок 3.1 – Интерфейс программы

При работе с программой пользователь может увидеть результаты интерполирования методом Лагранжа, Ньютона и аппроксимации функции в графическом виде и в табличном представлении, что позволяет наглядно сравнить результаты интерполирования и аппроксимации. Для этого необходимо нажать кнопку “GO!”, после чего на форме появится график, табличное значение функций.

Также в программе реализовано увеличение масштаба графика, ибо расходимость методов от функции довольно тяжело увидеть без приближения, а уж тем более методов друг от друга. Для увеличения масштаба графиков необходимо зажать левую клавишу мыши, появиться черный прямоугольник, который будет служить определителем насколько масштабировать график. Имеется и второй вариант масштабирования, который заключается в прокрутке колесика.

По умолчанию выбрано минимальное количество узлов для функции, т.е. 10 и $\beta = 0.5$.

В левом нижнем углу отображается минимальная погрешность, которая позволяет определить какую эмпирическую функцию стоит использоваться для аппроксимации, и среднее квадратическое отклонение(СКО), которое демонстрирует насколько аппроксимация правдоподобна. В случае недостоверности аппроксимации, внизу появится надпись, сообщающая пользователя о том, что аппроксимация неверна.

3.2. Описание программы

Функции:

`double Newton_f(double[] xval, double[] node, double x)` – функция для интерполирования функции методом Ньютона

`int beta()` – функция для передачи значения β в зависимости от выбранного значения. С список внесены радиобатоны, в зависимости от выбранного компонента β присваивается значение +2, пока не будет выполнено условие нажатия нужного нам радиобаттона.

`public double m(double x)` – функция, возвращающая данную нам функцию.

`public double PnLagrange(double x, double[] nodes,double []xval)` – функция для интерполирования функции методом Лагранжа.

`public void DrawGraph()` – функция для изображения графиков на компоненте класса ZedGraph, для занесения данных, для сравнения, в DataGridView значений зависимости от значения x.

public double Apr(double x, double[] nodes, int k, double[] xval) – функция, проводящая аппроксимацию данной нам функции.

double SKO(double x, double[] nodes, int k) – функция для нахождения СКО

double Ee(double[] nodes, double[] xval, int xmax, ref int outcount) – функция передает значение погрешности и номер погрешности по ссылке

Переменные, массивы, объекты классов:

`GraphPane pane` – объект служит для работы с классом `GraphPane`.

`var mlist, varLGList, var NTList, var MNKList, var SKOList` - объекты класса `PointPairList`, служат для записи и отображения значений интерполирования, аппроксимации при различных значениях x.

```
int xmax =Convert.ToInt16(comboBox1.Text) – переменная,
отвечающая за выбранное количество узлов, от которой зависит
шаг.
double step – шаг, задается делением нашего интервала на
количества узлов, соответственно.
double[] nodes = new double[xmax] – массив из значений в
функциях.
double[] xval = new double[xmax] – массив из x.
int k  - счетчик для определения элемента массивов.
```

4. **Численные эксперименты**

Перед программированием данной мне функции $f\ x\ = 16^{\sin x^{\cos x}} + \frac{6}{4^{sin^2\ x-\frac{\pi}{4}}} - 4 \cdot \beta$ я счел нужным провести исследование ее в Mathcad, т.е. посмотреть, что из себя представляет график функции. Осознав, что без преобразования ее, я не смогу получить желаемый результат пришлось преобразовать ее. Ниже приведены формулы и элементарные преобразования функции. Тригонометрические формулы, используемые для преобразования:

$$sin^2\ x\ = \frac{1 - \cos(2 \cdot x)}{2}$$

$$\cos\ \alpha - \beta\ = \cos\alpha \cdot \cos\beta + \sin\alpha \cdot \sin\beta$$

Пошаговое преобразование функции:

1. $sin^2\ \ x - \frac{\pi}{4}\ = \frac{1-\cos\ 2\cdot x-\frac{\pi}{2}}{2} =$

$$\frac{1}{2} - \frac{\cos\ 2\cdot x-\frac{\pi}{2}}{2} = \frac{1}{2} - \frac{\cos\ 2\cdot x\ \cdot\cos\frac{\pi}{2}+\sin\ 2\cdot x\ \cdot\sin\frac{\pi}{2}}{2} = \frac{1}{2} - \frac{\sin\ 2\cdot x}{2} = \frac{1-\sin\ 2\cdot x}{2}$$

2. $\frac{6}{4^{\frac{1-\sin\ 2\cdot x}{2}}} = \frac{6\cdot 4^{\frac{\sin(2\cdot x)}{2}}}{2} = 3 \cdot 4^{\frac{\sin 2\cdot x}{2}}$

$$f\ x\ = 16^{\sin x^{\cos x}} + 3 \cdot 4^{\frac{\sin 2\cdot x}{2}} - 4 \cdot \beta$$

3. Также, мне показалось, что переменную β, умноженную на константу 4, с шагом 0.5 можно преобразовать , т.е. параметр β=2,4,6.

Итоговая функция получает вид:

$$f\ x\ = 16^{\sin x^{\cos x}} + 3 \cdot 4^{\frac{\sin 2\cdot x}{2}} - \beta$$

где β=2,4,6,8

Т.к. вид данной функции не явно выражен, график не дает четкого понимания к какой из эмпирических функций относится моя функция. Для определения вида эмпирической функции стоит определить минимальную погрешность соответствующую построенному графику. Для этого использовались следующие формулы:

$$x_{arith} = \frac{x_1+x_n}{2},$$

$$x_{heom} = \overline{x_1 x_n}$$

$$x_{harm} = \frac{2x_1 x_n}{x_1 + x_n}$$

По вычисленным значениям независимой переменной x найдем из построенного графика соответствующее значение зависимой переменной y для пока еще неизвестной зависимости:

$$x_{arith} => y_1^*$$

$$x_{heom} => y_2^*$$

$$x_{harm} => y_3^*$$

Далее вычисляют среднее арифметическое крайних значений, среднее геометрическое крайних значений и среднее гармоническое крайних значений, соответственно:

$$y_{arith} = \frac{y_1+y_n}{2},$$

$$y_{heom} = \overline{y_1 y_n}$$

$$y_{harm} = \frac{2y_1 y_n}{y_1 + y_n}$$

После чего сравнивают найденные из графика величины с вычисленными значениями и составляют набор отличительных критериев(погрешностей) между расчетными значениями и табличными значениями для зависимой переменной. После чего из 7 погрешностей выбирают минимальную, которая соответствует определенной эмпирической функции. В моем случае:

$$\varepsilon_6 = |y_3^* - y_{arith}|$$

$\varepsilon_{min} = \varepsilon_6$, которой соответствует функция вида $y = a + \frac{b}{x}$.

Приведем систему нормальних уравнений для моей зависимости:

$$aN + b \sum_{n=1}^{N} x_n^{-1} = \sum_{n=1}^{N} y_n$$

$$a \sum_{n=1}^{N} x_n^{-1} + b \sum_{n=1}^{N} x_n^{-2} = \sum_{n=1}^{N} \frac{y_n}{x_n}$$

В результате выражения одного параметра через другой получаем значения параметров и эмпирическую функцию:

$$b = \sum_{i=1}^{n} f\ x_i\ x_n,$$

$$a = \frac{1}{N}$$

$$f\ x\ = \frac{1}{N} + \frac{\sum_{i=1}^{n} f\ x_i\ x_n}{x},$$

Ниже приведены таблицы, для сравнения и анализа полученных результатов, в которой изображены по порядку: значение x, значения функции в узлах, интерполирование методом Лагранжа, интерполирование методом Ньютона, аппроксимация методом наименьших квадратов, их погрешность.

x	f(x)	Lagrange	Newton	MNK	f(x)-Lagr	f(x)-Newt	f(x)-MNK
0,01936	2,136768	2,136768	2,136768	2,236768	0	0	-0,1
0,201178	2,932496	3,70788	3,707898	3,80788	-0,77538	-0,7754	-0,87538
0,382996	3,896821	5,893361	5,893412	5,993361	-1,99654	-1,99659	-2,09654
0,564815	5,0519	8,745908	8,745829	8,845908	-3,69401	-3,69393	-3,79401
0,746633	6,412034	12,04973	12,04968	12,14973	-5,6377	-5,63765	-5,7377
0,928451	7,975197	15,15718	15,15717	15,25718	-7,18198	-7,18197	-7,28198
1,110269	9,710542	17,22692	17,22693	17,32692	-7,51638	-7,51639	-7,61638
1,292087	11,54612	17,85815	17,85815	17,95815	-6,31203	-6,31203	-6,41203
1,473905	13,36523	17,40824	17,40822	17,50824	-4,043	-4,04299	-4,143
1,655724	15,02015	16,6819	16,68191	16,7819	-1,66175	-1,66176	-1,76175

В таблице, как мы можем заметить, минимальное количество узлов, т.е. 10, а $\beta = 0.5$. На первой итерации, т.е. в первом узле, методы интерполяции сходятся с нашей функцией. При последующих итерациях погрешность

методов довольно высока, метод Ньютона совсем незначительно уступает методу Лагранжа. Аппроксимация же данной нам функции проигрывает интерполяции.

x	f(x)	Lagrange	Newton	MNK	f(x)-Lagr	f(x)-Newt	f(x)-MNK
0,01936	-3,86323	-3,86323	-3,86323	-3,81323	0	0	-0,05
0,114598	-3,0675	-3,10902	-3,10901	-3,05902	0,04152	0,041504	-0,00848
0,209836	-2,10318	-2,20293	-2,20289	-2,15293	0,099751	0,099711	0,049751
0,305074	-0,9481	-1,12546	-1,12551	-1,07546	0,177359	0,177411	0,127359
0,400312	0,412034	0,13693	0,136896	0,18693	0,275104	0,275138	0,225104
0,49555	1,975197	1,585643	1,585635	1,635643	0,389554	0,389562	0,339554
0,590789	3,710542	3,200493	3,200518	3,250493	0,510049	0,510023	0,460049
0,686027	5,546119	4,929468	4,92953	4,979468	0,61665	0,616589	0,56665
0,781265	7,365234	6,683198	6,683112	6,733198	0,682035	0,682122	0,632035
0,876503	9,020147	8,340906	8,34086	8,390906	0,679241	0,679288	0,629241
0,971741	10,36436	9,771085	9,771072	9,821085	0,593277	0,59329	0,543277
1,066979	11,29265	10,86224	10,86225	10,91224	0,430414	0,430405	0,380414
1,162217	11,77087	11,55187	11,55188	11,60187	0,219004	0,218989	0,169004
1,257455	11,84248	11,84148	11,84149	11,89148	0,000997	0,000992	-0,049
1,352693	11,61221	11,79285	11,79286	11,84285	-0,18064	-0,18065	-0,23064
1,447931	11,21831	11,50994	11,50994	11,55994	-0,29163	-0,29163	-0,34163
1,54317	10,80752	11,1166	11,11659	11,1666	-0,30908	-0,30908	-0,35908
1,638408	10,52195	10,73923	10,73922	10,78923	-0,21728	-0,21727	-0,26728
1,733646	10,50122	10,49944	10,49944	10,54944	0,001771	0,001777	-0,04823
1,828884	10,90092	10,51843	10,51842	10,56843	0,38249	0,382498	0,33249

Следующий эксперимент отличается от предыдущего количеством узлов. Теперь взято максимальное количество, т.е. 20, а $\beta = 2$. Из таблицы можно сделать вывод, что от количества узлов зависит погрешность интерполяции, т.е., чем больше узлов, тем меньше погрешность, соответственно. Также, анализируя данные, хочу заметить, что погрешность методов на каждой итерации, напрямую зависит от функции. Аппроксимация дает нам более точный результат, нежели интерполяция. Переменная β служит лишь для изменения расположения графика по оси OY.

При увеличении количества узлов среднее квадратическое отклонение уменьшается, а от варьирования параметра β функции никак не зависит.

Вывод

Интерполяционную формулу Ньютона удобнее применять в том случае, когда интерполируется одна и та же функция *f(x),* но число узлов интерполяции постепенно увеличивается. Обязательное условие для использования формулы Ньютона – наличие равноотстоящих узлов. Если узлы интерполяции фиксированы и интерполируется не одна, а несколько функций, то удобнее пользоваться формулой Лагранжа. Полиномы, полученные методами Лагранжа и Ньютона, практически совпадают.

В ходе курсовой работы была разрабатана программа, которая позволяет строить интерполяционные полиномы Лагранжа и Ньютона функции, проводить аппроксимацию методом наименьших квадратов.

Список использованной литературы

1. Мазманішвілі, О.С., Ю.В. Шкварко Практикум з чисельних методів: Навч. посібник. /– К.: ІСДО, 1994. – 160 с.

2. Бодров, В.И. Численные методы и программирование: учебное пособие / В.И. Бодров, С.И. Дворецкий, В.Ф. Калинин. – М.: МИХМ, 1986. – 92 с.

3. Крылов, В.И. Вычислительные методы / В.И. Крылов, В.В. Бобков, П.И. Монастырский. – М. : Наука, 1976. – 304 с.

Приложение

Текст программы

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using ZedGraph;

namespace TestZ
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int beta()
        {
            int beta = 0;
            List<RadioButton> RB = new List<RadioButton>();
            RB.Add(rb1);
            RB.Add(rb2);
            RB.Add(rb3);
            RB.Add(rb4);
            for (int i = 0; i < RB.Count; i++)
            {
                beta += 2;
                if (RB[i].Checked)
                    return beta;
            }
            return 0;
        }

        public double m(double x) { return Math.Pow(16, Math.Pow(Math.Sin(x),
Math.Cos(x))) + 3 * Math.Pow(4, Math.Sin(2 * x) / 2) - beta();}

#region Approximation
       double MethodInterval(double x, double[] nodes, int k) { return  x *
Math.Pow(nodes[k], x); }

       double SKO(double x, double[] nodes, int k, double[] xval)
       {
           return  Math.Pow(Math.Pow(m(x) - MethodPoint(x, nodes, k, xval),
2)*Math.Pow(nodes.Count(),-1),0.5);
       }
        double MethodPoint(double x, double[] nodes, int k, double[] xval)
        {
            return Math.Pow( nodes.Count(),-1) + nodes[k] * xval[k] / x;
        }
        public double Apr(double x, double[] nodes, int k, double[] xval)
        {
            return MethodPoint(x, nodes, k,xval);
        }


        double Ee(double[] nodes, double[] xval, int xmax, ref int outcount)
        {
```

```
            int k = 0;
            double ma=0;
            double[] yStar = new double[3];
            double[] y = new double[3];
            double[] Em = new double[7];
            yStar[0] = (xval[0] + xval[xmax-1]) / 2;
            yStar[1] = Math.Sqrt(xval[0] * xval[xmax-1]);
            yStar[2] = 2 * (xval[0] * xval[xmax-1]) / xval[0] + xval[xmax-1];
            y[0] = (nodes[0] + nodes[xmax-1]) / 2;
            y[1] = Math.Sqrt(nodes[0] * nodes[xmax-1]);
            y[2] = 2 * (nodes[0] * nodes[xmax-1]) / nodes[0] + nodes[xmax-1];

            for (int i = 0; i < 7; i++)
            {
                if(i>=0 && i<=2)
                    Em[i] = Math.Abs(yStar[0] - y[i]);

                if (i==3)
                {
                    Em[i] = Math.Abs(yStar[1] - y[0]);
                }
                if (i > 3 && i<7)
                {

                    Em[i] = Math.Abs(yStar[2] - y[k]);
                    k++;
                }
                ma = Em.Min();

            }

            for (int i = 0; i < 7; i++)
            {
                if (ma == Em[i])
                {
                    outcount = 5;
                }
            }
            return ma;
        }

#endregion

        public double PnLagrange(double x, double[] nodes, double[] xval)
        {
            x += 0.005;
            double p = 1;
            double pn = 0;

           for (int i = 0; i < xval.Count(); i++)
           {
               double currentX = xval[i];
               for (int j = 0; j < xval.Count(); j++)
               {
                   if (currentX != xval[j])
                   {
                       p*=(x-xval[j])/(currentX-xval[j]);
                   }
               }
               pn += nodes[i]*p;
               p = 1;
           }
```

```
            return pn;
          }
        public double Newton(double[] xval, double[] node, double x)
        {
            x += 0.01;
            double res = node[0], F, den;
            int i, j, k;
            for (i = 1; i < node.Count(); i++)
            {
                F = 0;
                for (j = 0; j <= i; j++)
                {                   den = 1;

                    for (k = 0; k <= i; k++)
                    {
                        if (k != j) den *= (xval[j] - xval[k]);
                    }

                    F += node[j] / den;
                }
                            for (k = 0; k < i; k++) F *= (x - xval[k]);
                res += F;
            }
            return res;
        }
#region Drawing
          public void DrawGraph()
          {
              GraphPane pane = zedGraph1.GraphPane;
              pane.CurveList.Clear();
              pane.Title.Text = "";

              var mlist = new PointPairList();
              var LGList = new PointPairList();
              var NTList = new PointPairList();
              var MNKList = new PointPairList();


              double step =2;
              int xmax =Convert.ToInt16(comboBox1.Text);
              dataGridView1.RowCount = xmax-1;
              double[] nodes = new double[xmax];
              double[] xval = new double[xmax];
              step /= xmax;
              int k = 0;
              string tempsko = "";
              for (double i = 0.1; i <= 2; i+=step)
              {
                  nodes[k] = m(i);
                  xval[k] = i;
                  k++;
              }
              k = 0;
              for (double x =0.1; x <= 2; x +=step)
              {
                  mlist.Add(x, m(x));
                  LGList.Add(x, PnLagrange(x, nodes,xval));
                  NTList.Add(x, Newton(xval,nodes,x));
                  MNKList.Add(x, Apr(x, nodes,k,xval));
                  tempsko = SKO(x, nodes, k, xval).ToString();

                  k++;
```

```
            }

            for (int x = 0; x <xmax-1; x++)
            {
                dataGridView1[0, x].Value = mlist[x].X;
                dataGridView1[1, x].Value = mlist[x].Y;
                dataGridView1[2, x].Value = LGList[x].Y;
                dataGridView1[3, x].Value = NTList[x].Y;
                dataGridView1[4, x].Value = MNKList[x].Y;
            }



            pane.AddCurve("f(x)", mlist, Color.Olive, SymbolType.XCross);
            pane.AddCurve("LaGrange(x)", LGList, Color.Red, SymbolType.Triangle);
            pane.AddCurve("Newton(x)", NTList, Color.Blue, SymbolType.TriangleDown);
            LineItem curvemy=  pane.AddCurve("MNK(x)", MNKList, Color.Indigo,
SymbolType.Star);
            curvemy.Line.IsVisible = false;

            int outcount=0;
            string outE = Ee(nodes, xval, xmax, ref outcount).ToString();


            pane.XAxis.Scale.Min = 0;
            pane.XAxis.Scale.Max = 2;
            pane.YAxis.Scale.Min = -5;
            pane.YAxis.Scale.Max = 20;
            label2.Text = "Emin(" + outcount + ")=" + outE + "
=>y=a+b/x\nSKO="+tempsko;
            if (Convert.ToDouble(tempsko) < Convert.ToDouble(outE) && outE!="Nan")
            label3.Text ="SKO<<Emin=> Аппроксимация физически недостоверна!";
            else
            {
                label3.Text = "";
            }

            zedGraph1.AxisChange();
            zedGraph1.Invalidate();
        }
#endregion
        private void button1_Click(object sender, EventArgs e)
        {
            dataGridView1.Rows.Clear();

            if(comboBox1.Text=="")
                return;

           zedGraph1.Visible = true;
           DrawGraph();

        }
    }
}
```