**Аннотация**

Пояснительная записка курсовой работы "Интерполяция функции одной переменной методом Ньютона" содержит в себе введение, анализ задания описанием входных и выходных данных, обзор литературных источников, описание математической модели и методов вычислительной математики, пояснения к алгоритму, текст программы, инструкцию. При изучении дисциплины "Информатика" для написания курсовой работы использовались различные литературные источники, которые перечислены в настоящем документе. В данной курсовой работе приведена программа, которая применяется для интерполяции таблично заданной функции методом Ньютона. В ней был использован метод структурного программирования для облегчения написания и отладки программы, а также повышения ее наглядности и читаемости. Целью написания данной работы было получение и закрепление практических навыков разработки алгоритмов различными методами. Представленная программа реализована на языке программирования Pascal. Пояснительная записка содержит 25 листов, на которых размещено два рисунка, текст программы и описание программы и алгоритма.

**Содержание**

Введение

Анализ задания

Математическая модель задачи

Программирование функции формулы Ньютона

Обзор литературных источников

Разработка программы по схеме алгоритма

Инструкция пользования программой

Текст программы

Исходные данные и результат решения контрольного примера

Заключение

Список использованных источников

**Введение**

Современное развитие физики и техники тесно связано с использованием электронных вычислительных машин (ЭВМ). В настоящее время ЭВМ стали обычным оборудованием многих институтов и конструкторских бюро. Это позволило от простейших расчетов и оценок различных конструкций или процессов перейти к новой стадии работы - детальному математическому моделированию (вычислительному эксперименту), которое существенно сокращает потребность в натурных экспериментах, а в ряде случаев может их заменить.

Сложные вычислительные задачи, возникающие при исследовании физических и технических проблем, можно разбить на ряд элементарных -таких как вычисление интеграла, решение дифференциального уравнения и т. п. Многие элементарные задачи являются несложными и хорошо изучены. Для этих задач уже разработаны методы численного решения, и нередко имеются стандартные программы решения их на ЭВМ. Есть и достаточно сложные элементарные задачи; методы решения таких задач сейчас интенсивно разрабатываются.

В связи с этим современный специалист с высшим образованием должен обладать не только высоким уровнем подготовки по профилю своей специальности, но и хорошо знать математические методы решения инженерных задач, ориентироваться на использование вычислительной техники, практически освоить принципы работы на ЭВМ.

**Анализ задания**

В качестве входных данных использованы:

1. Количество узлов.
2. Табличные значения функции.

Выходными данными, т.е. результатом программы является:

1. Значения таблично заданной функции в промежуточных значениях.
2. График полинома.

**Математическая модель задачи**

При выполнении курсовой работы была выбрана следующая математическая модель:

Интерполяция и приближение функций.

*1. Постановка задачи.*

Одной из основных задач численного анализа является задача об интерполяции функций. Часто требуется восстановить функцию для всех значений на отрезке если известны ее значения в некотором конечном числе точек этого отрезка. Эти значения могут быть найдены в результате наблюдений (измерений) в каком-то натурном эксперименте, либо в результате вычислений. Кроме того, может оказаться, что функция задается формулой и вычисления ее значений по этой формуле очень трудоемки, поэтому желательно иметь для функции более простую (менее трудоемкую для вычислении) формулу, которая позволяла бы находить приближенное значение рассматриваемой функции с требуемой точностью в любой точке отрезка. В результате возникает следующая математическая задача.



Пусть и» отрезке задана сетка со



и в ее узлах заданы значения функции , равные



.



Требуется построить интерполянту — функцию , совпадающую с функцией в узлах сетки:



.



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



*2. Интерполяция по Ньютону*

Дана табличная функция:

|  |  |  |
| --- | --- | --- |
| i |  |  |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |
| .. | .. | .. |
| n |  |  |

Или

, (1)



Точки с координатами называются узловыми точками или узлами.



Количество узлов в табличной функции равно N=n+1.

Необходимо найти значение этой функции в промежуточной точке, например, , причем . Для решения задачи используется интерполяционный многочлен.



Интерполяционный многочлен по формуле Ньютона имеет вид:



где n – степень многочлена,

Интерполяционная формула Ньютона формула позволяет выразить интерполяционный многочлен через значение в одном из узлов и через разделенные разности функции , построенные по узлам .



Сначала приведем необходимые сведения о разделенных разностях.

Пусть в узлах

,



известны значения функции . Предположим, что среди точек , , нет совпадающих. Разделенными разностями первого порядка называются отношения



, ,.



Будем рассматривать разделенные разности, составленные по соседним узлам, т. е. выражения

.



По этим разделенным разностям первого порядка можно построить разделенные разности второго порядка:

,



,



Таким образом, разделённая разность -го порядка на участке может быть определена через разделённые разности -го порядка по рекуррентной формуле:



. (3)



где , , - степень многочлена.



Максимальное значение равно . Тогда и разделенная разность n-го порядка на участке равна



,



т.е. равна разности разделенных разностей -го порядка, разделенной на длину участка .



Разделенные разности



являются вполне определенными числами, поэтому выражение (1) действительно является алгебраическим многочленом -й степени. При этом в многочлене (1) все разделенные разности определены для участков , .



При вычислении разделенных разностей принято записывать их в виде таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  | • |  |  |
|  |  | • | • | • |  |
| ■ | • | • | • |  |  |
| • | • | • |  |  |  |
| • | • |  |  |  |  |
|  |  |  |  |  |  |

Разделенная разность -го порядка следующим образом выражается через значения функции в узлах:



. (1)



Эту формулу можно доказать методом индукции. Нам потребуется частный случай формулы (1):



Интерполяционным многочленом Ньютона называется многочлен



Рассмотренная форма полинома Ньютона носит название первой интерполяционной формулы Ньютона, и используется, обычно, при интерполировании вначале таблицы.

Заметим, что решение задачи интерполяции по Ньютону имеет некоторые преимущества по сравнению с решением задачи интерполяции по Лагранжу. Каждое слагаемое интерполяционного многочлена Лагранжа зависит от всех значений табличной функции yi, i=0,1,…n. Поэтому при изменении количества узловых точек N и степени многочлена n (n=N-1) интерполяционный многочлен Лагранжа требуется строить заново. В многочлене Ньютона при изменении количества узловых точек N и степени многочлена n требуется только добавить или отбросить соответствующее число стандартных слагаемых в формуле Ньютона (2). Это удобно на практике и ускоряет процесс вычислений.

**Программирование функции формулы Ньютона**

Для построения многочлена Ньютона по формуле (1) организуем циклический вычислительный процесс по . При этом на каждом шаге поиска находим разделенные разности k-го порядка. Будем помещать разделенные разности на каждом шаге в массив Y.



Тогда рекуррентная формула (3) будет иметь вид:



(4)



В формуле Ньютона (2) используются разделенные разности -го порядка, подсчитанные только для участков т.е. разделенные разности -го порядка для . Обозначим эти разделенные разности k-го порядка как . А разделенные разности, подсчитанные для , используются для расчетов разделенных разностей более высоких порядков.



Используя (4), свернем формулу (2). В результате получим

(5)



где

– значение табличной функции (1) для .



– разделенная разность -го порядка для участка .



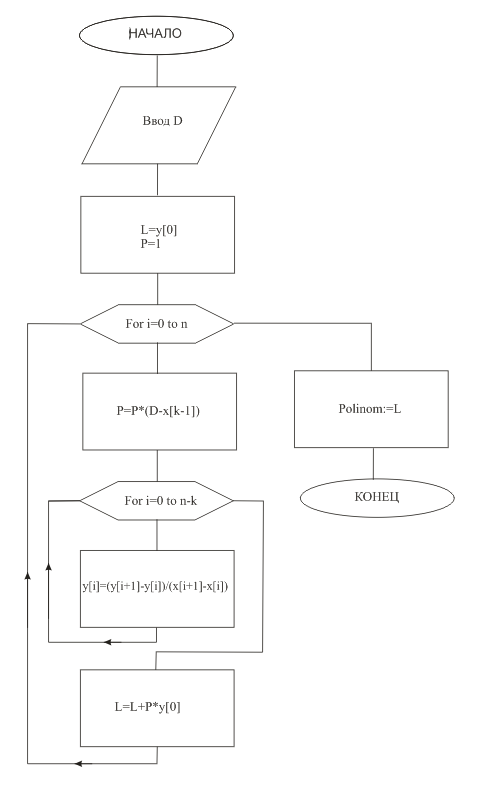
.



Для вычисления Р удобно использовать рекуррентную формулу внутри цикла по .



Схема алгоритма интерполяции по Ньютону представлена на рисунке:



Function POlinom(n: integer; d:real; x,y :per):real;

var

l:real;

k,i:integer;

p: real;

begin

L:=y[0];

P:=1;

for k:=1 to n do begin

P:=P\*(D-X[k-1]);

for i:=0 to (n-k) do begin

Y[i]:=(y[i+1]-y[i])/(x[i+k]-x[i]);

end;

L:=L+P\*y[0];

end;

Polinom:=l;

end;

где

n – количество узлов

x[i],y[i] – табличные значения функции

D – точка, в которой необходимо вычислить значение l

**Обзор литературных источников**

*1. Численные методы*

Численные методы являются одним из мощных математических средств решения задачи. Простейшие численные методы мы используем всюду, например» извлекая квадратный корень на листке бумаги. Есть задачи, где без достаточно сложных численных методов не удалось бы получить ответа; классический пример—открытие Нептуна по аномалиям движения Урана.

В современной физике таких задач много- Более того, часто требуется выполнить огромное число действий за короткое время, иначе ответ будет не нужен. Например, суточный прогноз погоды должен быть вычислен за несколько часов; коррекцию траектории ракеты надо рассчитать за несколько минут (напомним, что для расчета орбиты Нептуна Леверье потребовалось полгода); режим работы прокатного стана должен исправляться за секунды. Это немыслимо без мощных ЭВМ, выполняющих тысячи или даже миллионы операций в секунду.

Современные численные методы и мощные ЭВМ дали возможность решать такие задачи, о которых полвека назад могли только мечтать. Но применять численные методы далеко не просто. Цифровые ЭВМ умеют выполнять только арифметические действия и логические операции. Поэтому помимо разработки математической модели, требуется еще разработка алгоритма, сводящего все вычисления к последовательности арифметических и логических действий. Выбирать модель и алгоритм надо с учетом скорости и объема памяти ЭВМ: чересчур сложная модель может оказаться машине не под силу, а слишком простая — не даст физической точности.

Сам алгоритм и программа для ЭВМ должны быть тщательно проверены. Даже проверка программы нелегка, о чем свидетельствует популярное утверждение: «В любой сколь угодно малой программе есть, по меньшей мере, одна ошибка». Проверка алгоритма еще более трудна, ибо для сложных алгоритмов не часто удается доказать сходимость классическими методами. Приходится использовать более или менее надежные «экспериментальные» проверки, проводя пробные расчеты на ЭВМ и анализируя их.

Строгое математическое обоснование алгоритма редко бывает исчерпывающим исследованием. Например, большинство доказательств сходимости итерационных процессов справедливо только при точном выполнении всех вычислений; практически же число сохраняемых десятичных знаков редко происходит 5 — 6 при «ручных» вычислениях и 10—12 при вычислениях на ЭВМ. Плохо поддаются теоретическому исследованию «маленькие хитрости» — незначительные на первый взгляд детали алгоритма, сильно влияющие на его эффективность. Поэтому окончательную оценку метода можно дать только после опробования его в практических расчетах.

К чему приводит пренебрежение этими правилами — видно из принципа некомпетентности Питера: «ЭВМ многократно увеличивает некомпетентность вычислителя».

Для сложных задач разработка численных методов и составление программ для ЭВМ очень трудоемки и занимают от нескольких недель до нескольких лет. Стоимость комплекса отлаженных программ нередко сравнима со стоимостью экспериментальной физической установки. Зато проведение отдельного расчета по такому комплексу много быстрей и дешевле, чем проведение отдельного эксперимента. Такие комплексы позволяют подбирать оптимальные параметры исследуемых конструкций, что не под силу эксперименту.

Однако численные методы не всесильны. Они не отменяют все остальные математические методы. Начиная исследовать проблему, целесообразно использовать простейшие модели, аналитические методы и прикидки. И только разобравшись в основных чертах явления, надо переходить к полной модели и сложным численным методам; даже в этом случае численные методы выгодно применять в комбинации с точными и приближенными аналитическими методами.

Современный физик или инженер-конструктор для успешной работы должен одинаково хорошо владеть и «классическими» методами, и численными методами математики.

*2. Турбо Паскаль*

Язык Паскаль с момента своего создания Н. Виртом в 1971 году играет особую роль и в практическом программировании, и в его изучении. С непревзойденной четкостью в нем реализованы принципы структурного программирования. Паскаль стал первым языком, с которым знакомиться большинство будущих программистов.

Трансляторы для программ, написанных на Паскале, разработаны для различных компьютеров и в настоящее время имеют множество разновидностей. Они являются компиляторами, обрабатывающие разработанные программистами тексты программ.

Схематически программа представляется в виде последовательности восьми разделов:

1. Заголовок программы
2. Описание внешних модулей, процедур и функций
3. Описание меток
4. Описание констант
5. Описание типов переменных
6. Описание переменных
7. Описание функций и процедур
8. Раздел операторов

**Разработка программы по схеме алгоритма**

При разработке программы в данной работе используются следующие операторы и стандартные процедуры:

Program - Заголовок программы

Uses – раздел подключения модулей

Begin – открывающая логическая скобка

End – закрывающая логическая скобка

:= - оператор присваивания

Crt - (Cathod ray tube - электронно-лучевая трубка) один из наиболее часто используемых модулей. Он содержит процедуры обслуживания процессов вывода информации на экран, ввода с клавиатуры, а также процедуры и функции вывода звуковых сигналов, работы с окнами на экране и вывода цветных текстовых строк на экран.

Graph – графический модуль для вывода базовых графических элементов, таких как точки, отрезки прямых линий, дуги и целые окружности и других графических элементов, называемых графическими примитивами

Var – раздел описания переменных

Writeln, Write – операторы вывода информации

Readln, Read – операторы ввода информации

If <условие> then <оператор>– оператор условного перехода

For <параметр>:=<нач.знач.> to <конечн.знач.> do <оператор> – оператор цикла с параметром

Repeat <оператор> until <условие> - оператор цикла с постусловием

Clrscr – очистка экрана

Initgraph – процедура инициализации графического режима

Closegraph – процедура закрытия графического режима

Line (x1, y1, x2, y2) – соединение двух точек отрезком

Putpixel (x, y, c) – построение точки (x, y) цветом с

Readkey – оператор считывание кода клавиш

Outtextxy (x, y, st) – вывод строки st, начиная с точки (x,y)

Getmaxx – результатом этой функции будет max значение x в данном видеорежиме

Goto – перейти к

+ - арифметическая операция сложения

- - арифметическая операция вычитания

\* - арифметическая операция умножения

/ - арифметическая операция деления

*Описание переменных и констант используемых в алгоритме*

n – количество узлов в таблице, не считая начальную точку ;



i, j – счётчики;

- значения узлов записанных в одномерные массивы;



D – переменная, используемая для нахождения значения полинома Ньютона в этой точке;

L – переменная значения полинома Ньютона

k, step – константы используемые для построения графика полинома;

u – переменная шага деления графика;

Для описания алгоритма в данной курсовой работе были пронумерованы символы.

**Инструкция пользования программой**

Для запуска программы необходимо дважды щелкнуть на ярлыке с именем Niton.exe. После этого на экран будет выведен титульный лист. Чтобы продолжить надо нажать клавишу Enter.

Следующим шагом в окне программы будет показана строка с текстом «Показать пояснения к программе (1/0)?», чтобы увидеть их следует нажать 1 и подтвердить ввод нажатием клавиши Enter. Чтобы продолжить надо нажать клавишу Enter. Сразу после этого в диалоговом окне появится строка «Введите количество уpлов n (N=n+1)», где нужно указать количество (N-1) узлов таблицы и нажать Enter. Далее надо будет ввести значения из таблицы, по окончанию ввода нажать Enter.

На экран будет выведена введённая таблица значений. Затем пользователю будет предложено «Введите x ». Нужно ввести x для которого необходимо найти приближённое значение. После этого программа вычислит значение и предложит найти значения для другого x.

Дальше программа попросит ввести шаг деления графика. После ввода шага программа построит график полинома. Для продолжения нужно нажать Enter.

Потом программа спросит «повторить вычисления и построения графика полинома для другой функции?» Чтобы начать заново нужно нажать 1, чтобы закончить работу с программой нажать 0 и после ввода подтвердить выбор клавишей Enter.

**Текст программы**

program interpol;

uses crt,graph;

const

MAXCOUNT=30;

type

per = array [0..MAXCOUNT] of real;

var

X,y :per;

n,i :integer;

l,D,f :real;

label Lp, Lt;

{Процедура вывода титульного листа}

Procedure Titul;

begin

Clrscr;

GoToXY(23,2);

Writeln(‘Федеральное агентство по образованию');

GoToXY(22,3);

Writeln('Тульский государственный университет');

GoToXY(28,4);

Writeln('КАФЕДРА РАДИОЭЛЕКТРОНИКИ');

GoToXY(14,8);

Writeln('Интерполяция функции одной переменной методом Ньютона.');

GoToXY(27,9);

Writeln('Построение графика полинома.');

GoToXY(34,12);

Writeln('Вариант #7');

GoToXY(24,17);

Writeln('Студент гр. 220371 Поляков A.M.');

GoToXY(20,19);

Writeln('Руководитель доцент, K.T.H. Давыдов B.B.');

GoToXY(33,23);

Writeln('Тула, 2008 g.');

readkey;

clrscr;

end;

{Процедура вывода пояснения к программе}

Procedure help;

begin

clrscr;

writeln (Эта программа по значениям функции f(x) заданной таблично в нескольких точках отрезка находит ее значения в ' +

+ остальных точках данного отрезка. Точки с координатами (xi, yi) называются узловыми точками или узлами.');

writeln ('Количество узлов в табличной функции должно быть равно N=n+1. ');

writeln (' После ввода количества узлов n (начальная точка (x[0],y[0]) не является узлом) нужно вводить узловые точки +

+' функции. После этого программа сможет находить значения данной функции в остальных точках отрезка (x[0]..x[n]).');

writeln (После этого на экран будут выведен график полинома.');

readkey;

clrscr;

end;

{Процедура ввод табличных значений}

procedure Enter(var X,y: per);

var

i: integer;

label mp;

begin

mp: for i:=0 to n do

begin

write('X[',i,'] = '); readln(x[i]);

write('y[',i,'] = '); readln(y[i]);

end;

for i:=0 to n-1 do

if x[i+1]-x[i]<=0 then

begin

writeln ('Ошибка. Повторите ввод.');

goto mp

end;

end;

{процедура вывода табличных значений}

procedure Print(n: integer; X,y: per); var

i: integer;

begin

for i:=0 to n do

begin

write(x[i]:12:6);

end;

writeln;

for i:=0 to n do

begin

write(y[i]:12:6);

end;

writeln;

end;

{Функция формулы Ньютона}

Function Polinom(n: integer; d:real; X,y :per):real;

var

l:real;

k,i:integer;

p: real;

begin

L:=y[0];

P:=1;

for k:=1 to n do begin

P:=P\*(D-X[k-1]);

for i:=0 to (n-k) do begin

Y[i]:=(y[i+1]-y[i])/(x[i+k]-x[i]);

end;

L:=L+P\*y[0];

end;

POlinom:=l;

end;

{ процедура построение графика}

procedure Grafik(n: integer; D :real ; X,Y: per; L:real);

const

step=10;

var

driver,mode: integer;

i:longint;

st:string;

u,k:integer;

begin

writeln('Введите шаг деления графика');

readln(u);

k:=26;

driver:=detect;

initgraph (driver,mode,'');

setcolor (1);

line (320,0,320,480);

line (0,240,640,240);

for i:=0 to 32 do begin

setlineStyle (1,0,0);

line (0,i\*k+6,640,i\*k+6);

line (i\*k+8,0,i\*k+8,480);

end;

setcolor (3);

outtextxy (310,15,'y');

outtextxy (620,240,'x');

for i:=0 to getmaxx div (2\*k) do

begin

str (i\*u,st);

outtextxy(getmaxx div 2+i\*(k),getmaxy div 2+step,st);

str (-i\*u,st);

outtextxy (getmaxx div 2-i\*k,getmaxy div 2+step,st);

end;

for i:=1 to getmaxy div (2\*k) do

begin

str (-i\*u,st);

outtextxy (getmaxx div 2+step,getmaxy div 2+i\*k,st);

str (i\*u,st);

outtextxy (getmaxx div 2+step,getmaxy div 2-i\*k,st);

end;

d:=-u\*12;

repeat

d:=d+0.002;

putpixel (round(320+d\*k/u),round(240+(-POlinom(n,d,x,y))\*k/u),10);

until d>u\*12;

readkey;

end;

{Основной текст программы}

begin

TextMode(3);

TextBackground(1);

TextColor(14);

Titul;

writeln ('Вывести пояснение к программе?? (Да-1,Нет-0)');

read (f);

if f=1 then help else

lp:clrscr;

writeln('Введите количество узлов n (N=n+1)');

read(n);

Enter(X,y);

Print(n,X,y);

repeat

lt:Writeln('BbBedite X (ot ',x[0]:4:2,' do ',x[n]:4:2,')');

read(d);

if d<x[0] then begin

writeln('Ошибка. x не может быть меньше ',x[0]:4:2);

goto lt; end;

if d>x[n] then begin

writeln('Ошибка. x не может быть больше ',x[n]:4:2);

goto lt; end;

writeln(Polinom (n,d,X,y):6:3);

writeln('Найти значения для другой точки X?(ДА-1,НЕТ-0)');

read(f)

until f=0;

Grafik(n,D,X,Y,l);

readkey;

CloseGraph;

clrscr;

writeln('Повторить для другой функции? (Да-1,Нет-0)');

read(f);

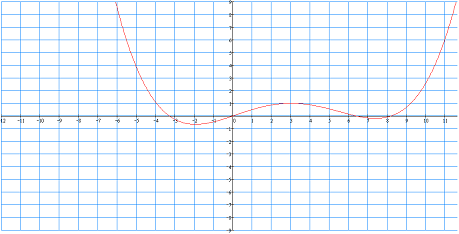
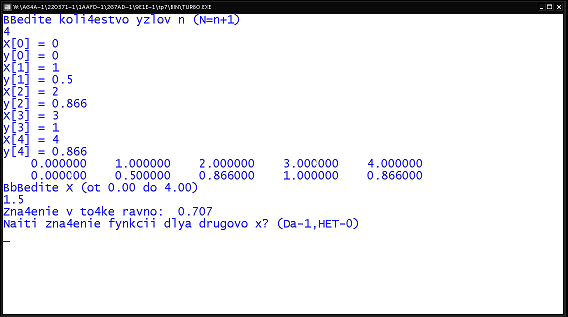
if f=1 then goto lp else end.

**Исходные данные и результат решения контрольного примера**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
|  | 0 | 0.5 | 0.866 | 1 | 0.866 |

# Вычислим значение таблично заданной функции в точке x=1.5



# Ми получили значение 0.707 которое мало отличается от точного значения

# .



# **Заключение**

В курсовой работе я рассмотрел только первую формулу полинома Ньютона, которая используется вблизи начала таблицы. Интерполяционный полином в форме Ньютона удобно использовать, если точка интерполирования находится вблизи начала таблицы. Этот полином интересен тем, что каждая частичная сумма первых m слагаемых есть интерполяционный полином m-1 степени, построенный по m первым табличным точкам. Поэтому интерполяционные полиномы Ньютона удобно использовать при последовательном увеличении степени интерполяционного многочлена.

К недостатку формулы Ньютона можно отнести то, что при вычислениях в таблице с постоянным шагом при увеличении количества узлов не всегда удается добиться повышения точности вычислений. Это обусловлено тем, что равноотстоящие узлы не являются лучшими с точки зрения уменьшения погрешности интерполирования. Если имеется возможности выбора узлов интерполирования, то их следует выбирать так, чтобы обеспечить минимум погрешности интерполяции.

В процессе выполнения курсовой работы были закреплены приобретенные за период обучения навыки и умения самостоятельного составления алгоритмов и программ на языке программирования Turbo Pascal 7.0 для решения простых типовых математических задач. Эта работа ещё раз подтвердила полезность использования ЭВМ для решения прикладных математических задач. Полученные знания и накопленный опыт решения простых задач в будущем позволят разрабатывать гораздо более сложные программы и алгоритмы, облегчат разбиение сложных задач на простые элементы.

**Список использованных источников**

1. Введение в численные методы/ А.А. Самарский – М.: наука, 1982.
2. Начала программирования на языке Паскаль/С.А. Абрамов – М., 1987.
3. Практическое руководство по методам вычислений с приложением программ для персональных компьютеров/ В.И. Ракитин – М.: Высш. шк., 1998.
4. Программирование в среде Турбо Паскаль/Д.Б. Поляков – М., 1992.
5. Справочник по алгоритмам и программам на языке бейсик для персональных ЭВМ/ В.П. Дьяконов – М.: Наука, 1987.
6. Турбо Паскаль 7.0/В.В. Фаронов – М., 1998.
7. Численные методы анализа/Б.П. Демидович – М.: Государственное издательство физико-математической литературы, 1962.
8. Численные методы /Калиткин Н.Н. – М.: 1996
9. Немнюгин С.A. Turbo Pascal - СПб.: Питер, 2002.- 496 с,