Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы численного анализа»

**ОТЧЕТ**

к лабораторной работе №6

на тему:

**«ИНТЕРПОЛЯЦИОННЫЕ МНОГОЧЛЕНЫ»**

БГУИР 1-40-04-01

|  |
| --- |
| Выполнил студент группы 253504  Дмитрук Богдан Ярославович |
| 31.10.2023 |
| (дата, подпись студента) |
| Проверил  Анисимов Владимир Яковлевич |
|  |
| (дата, подпись преподавателя) |

Минск 2023

**СОДЕРЖАНИЕ**

[1. Цели выполнения задания 3](#_Toc147528122)

[2. Краткие теоретические сведения 4](#_Toc147528123)

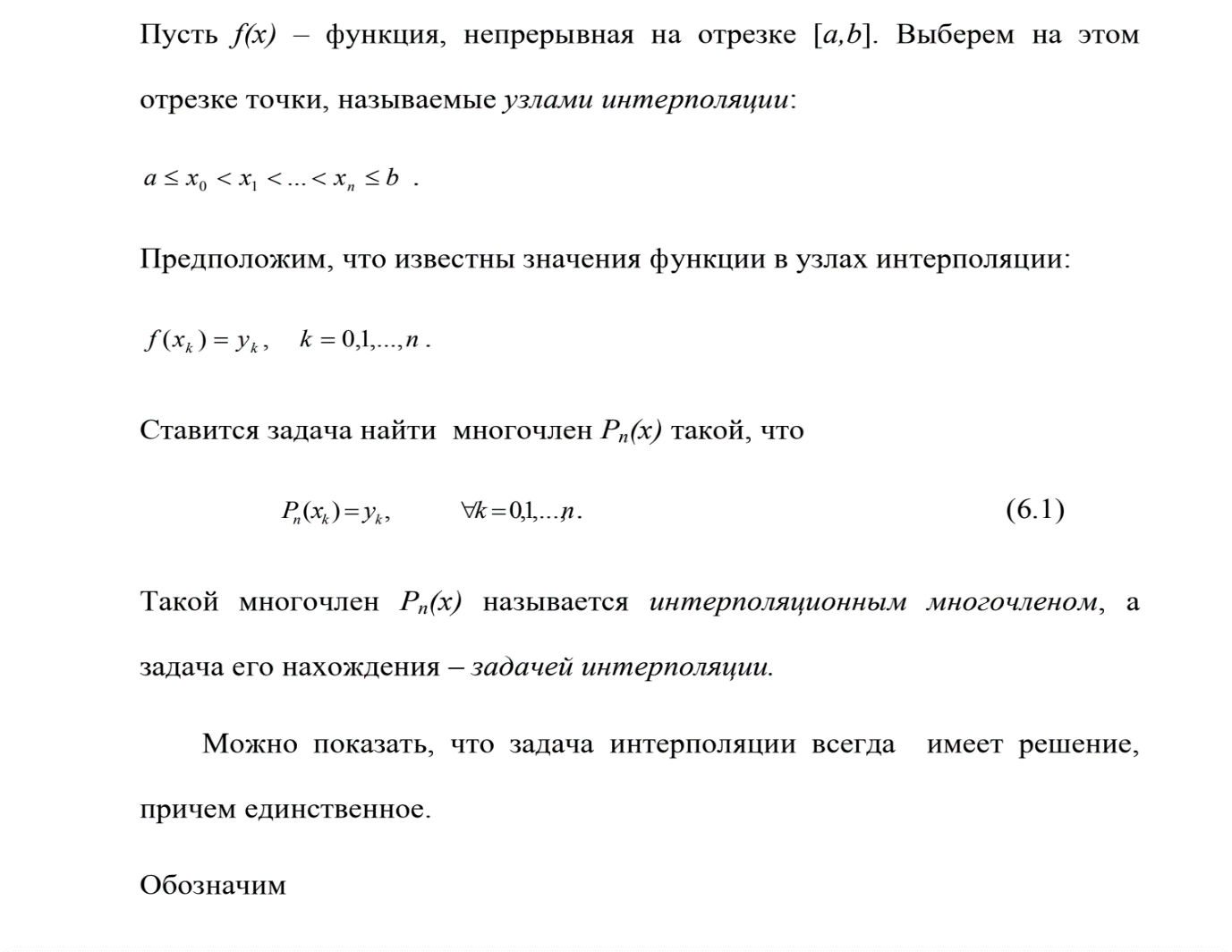
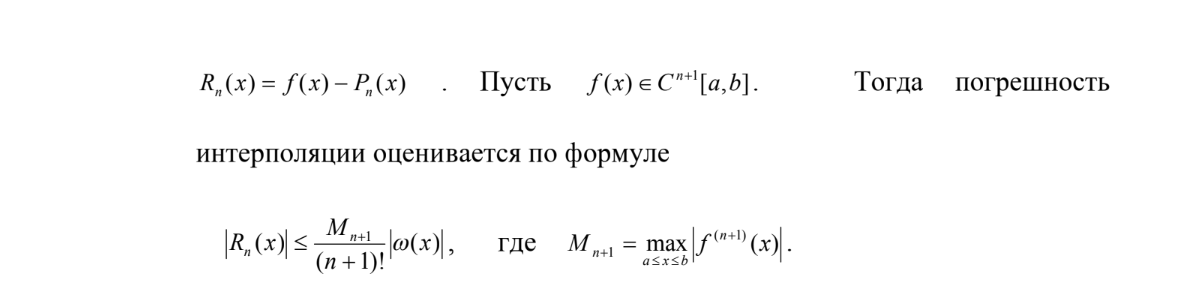
[3. Задание 11](#_Toc147528124)

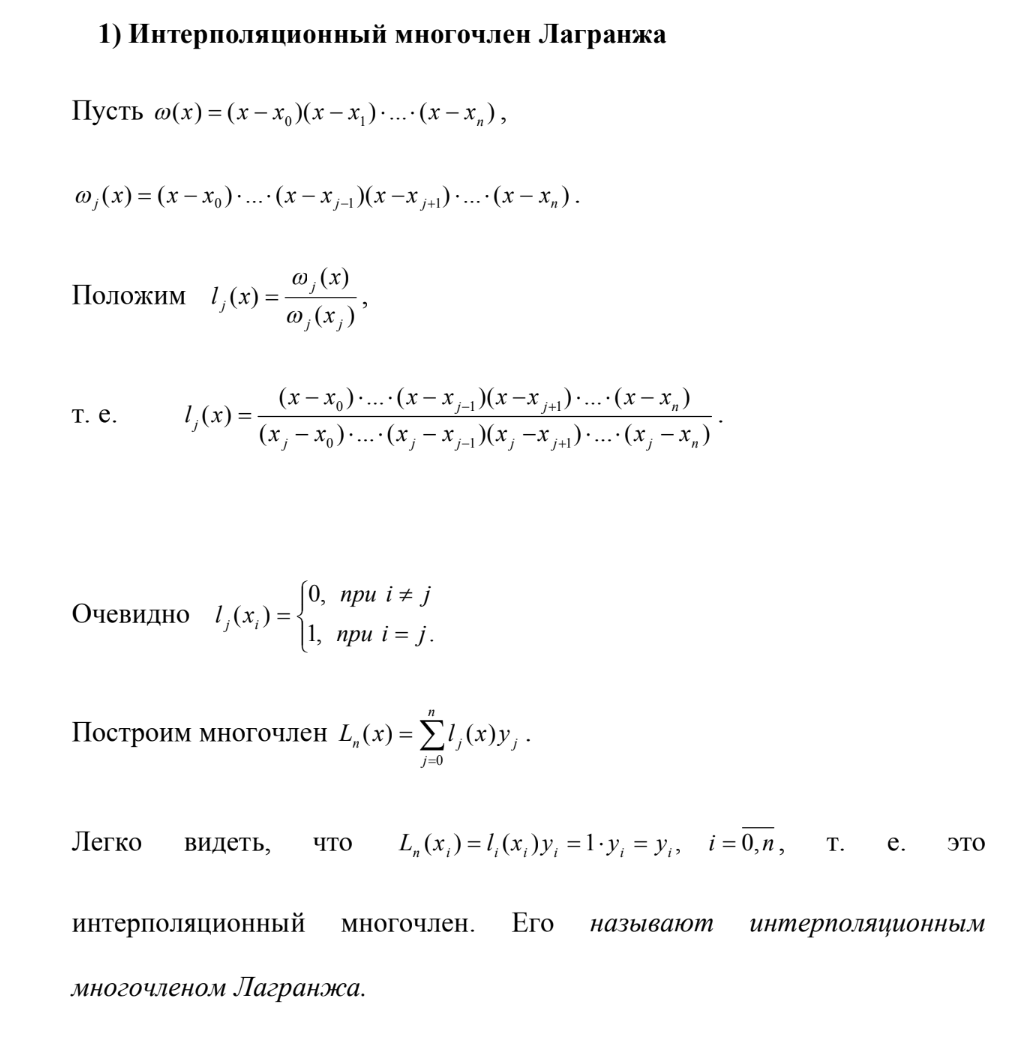
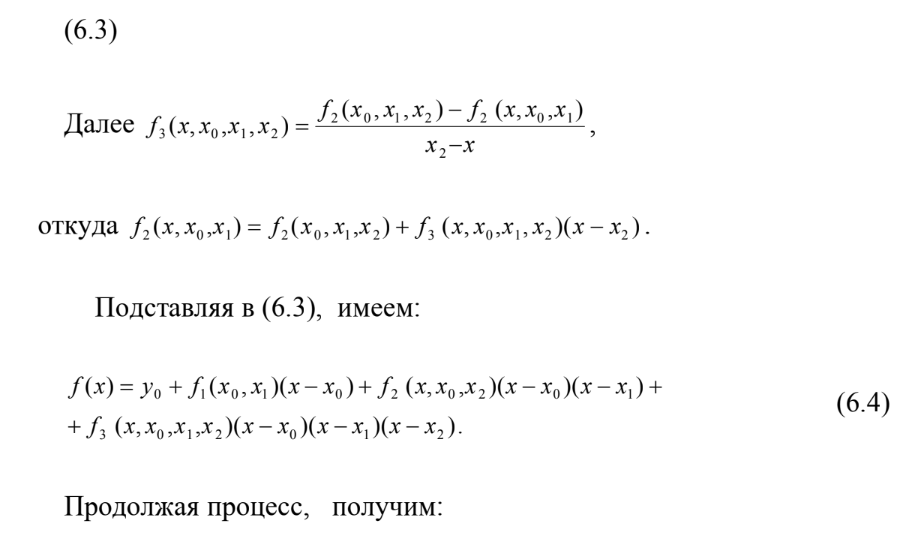
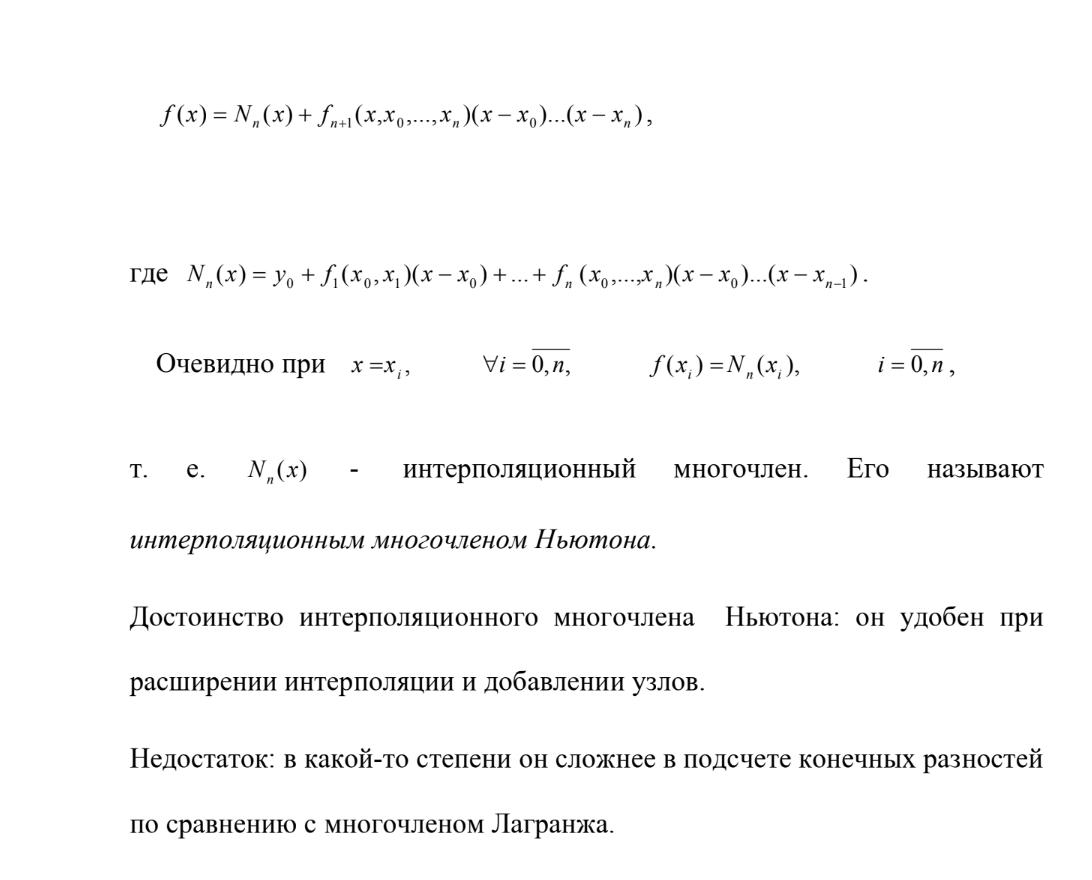
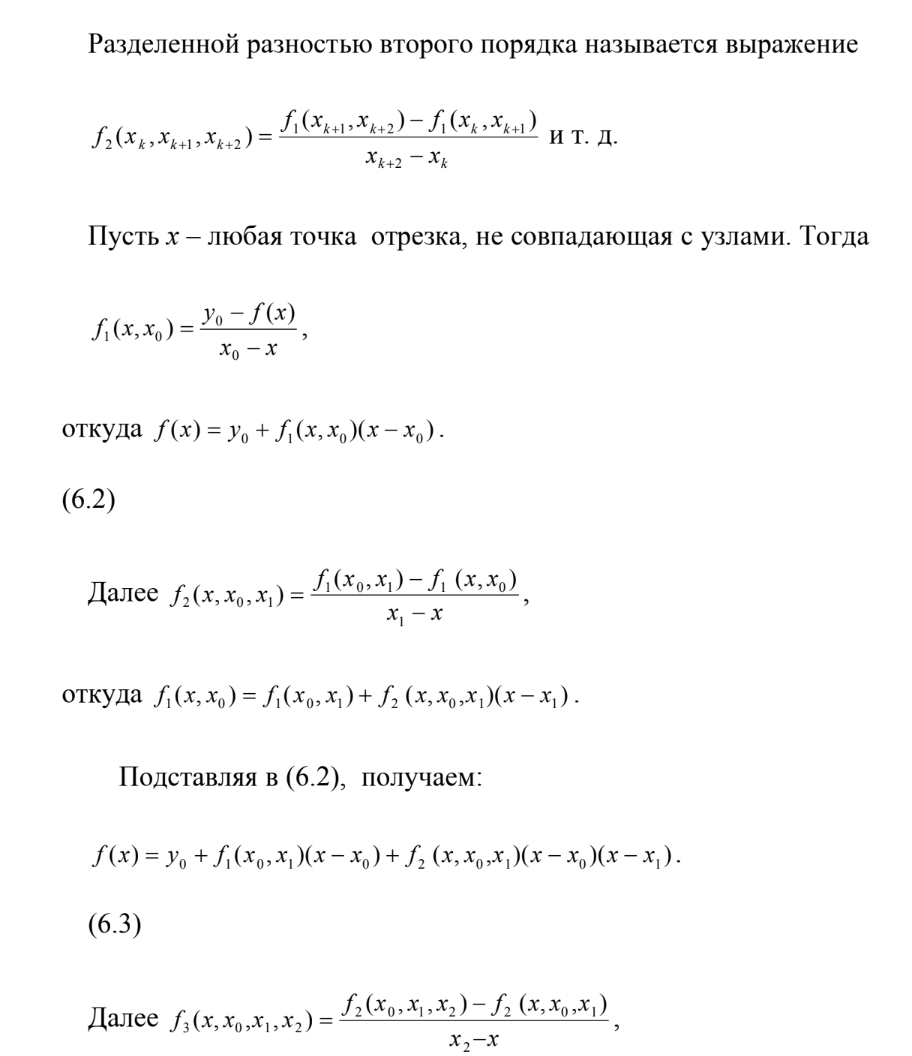
[4. Программная реализация 12](#_Toc147528125)

[5. Полученные результаты 16](#_Toc147528126)

[6. Выводы 18](#_Toc147528127)

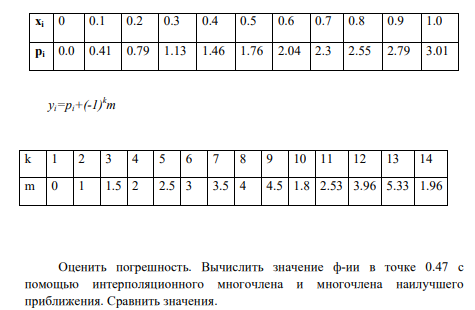
1. ЦЕЛИ ВЫПОЛНЕНИЯ ЗАДАНИЯ
2. Изучить интерполяцию функций с помощью интерполяционных многочленов Лагранжа и Ньютона.
3. Составить программу, позволяющую получить интерполяционные многочлены Лагранжа и Ньютона
4. Построить интерполяционные многочлены в Форме Лагранжа и Ньютона в соответствии с вариантом.
5. Оценить погрешность. Вычислить значение функции в точке с помощью интерполяционного многочлена наилучшего приближения. Сравнить значения.
6. **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

1. **ЗАДАНИЕ**

Построить интерполяционные многочлены в форме Лагранжа и Ньютона используя номер варианта k и соответствующие значения параметров m и p из таблиц:



1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

**Метод Лагранжа**

def lagrange\_l(x\_list: list, x\_cur):

x = symbols('x')

lg = 1

n = len(x\_list)

for i in range(n):

if x\_cur == x\_list[i]:

continue

lg \*= (x - x\_list[i]) / (x\_cur - x\_list[i])

return lg

def lagrange\_poly(x, y):

if len(x) != len(y):

raise ValueError("x and y are not same length")

n = len(x)

L = 0

for i in range(n):

L += y[i] \* lagrange\_l(x, x[i])

return L

**Метод Ньютона**

def get\_a(x: list, y: list):

a\_table = np.zeros(shape=(len(y) - 1, len(y) - 1))

n = a\_table.shape[0]

for i in range(n):

a\_table[0, i] = (y[i + 1] - y[i]) / (x[i+1] - x[i])

for i in range(1, n):

for j in range(n - i):

a\_table[i, j] = (a\_table[i - 1, j + 1] - a\_table[i - 1, j]) / (x[i+1+j] - x[j])

return a\_table

def newton\_poly(x\_list: list, y\_list: list):

if len(x\_list) != len(y\_list):

raise TypeError("x and y must be same size")

n = len(x\_list)

x = symbols('x')

N = 0

a\_table = get\_a(x\_list, y\_list)

for i in range(n):

if i == 0:

N += y\_list[0]

continue

x\_poly = 1

for j in range(i):

x\_poly \*= (x-x\_list[j])

N += a\_table[i-1, 0] \* x\_poly

return N

**Многочлен полученный методом наименьших квадратов** def sum\_of\_pow(x\_list, pow):

n = x\_list.shape[0]

sum = 0

for i in range(n):

sum += np.power(x\_list[i], pow)

return sum

def sum\_x\_y\_pows(x\_list, y\_list, pow):

n = x\_list.shape[0]

sum = 0

for i in range(n):

sum += y\_list[i] \* np.power(x\_list[i], pow)

return sum

def min\_square(x\_list, y\_list, m):

n = x\_list.shape[0]

A = np.zeros(shape=(m+1, m+1))

for i in range(m+1):

for j in range(m+1):

A[i, j] = sum\_of\_pow(x\_list, i+j)

b = np.zeros(shape=m + 1)

for i in range(m+1):

b[i] = sum\_x\_y\_pows(x\_list, y\_list, i)

a\_vec = np.linalg.solve(A, b)

return a\_vec

def get\_polynomial(x\_list, y\_list):

a\_vec = min\_square(x\_list, y\_list, y\_list.shape[0])

x = sp.symbols('x')

polynomial = 0

for i in range(len(a\_vec)):

polynomial += a\_vec[i] \* x\*\*i

return polynomial

1. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Результаты, полученные программным продуктом по окончанию обработки данных из условия:

Interpolation point: x = 0.47

Lagrange poly result on x = 0.47:

-1.8273479202911966

Newton poly result on x = 0.47:

-1.827347920291197

Polynomial of best approximation(least squares poly) result on x = 0.47

-1.8272769999629404

Как мы можем заметить, значения полиномов полученных методом лагранжа и методом ньютона практически не отличаются друг от друга в данной точке, однако расхождения наблюдаются с методом наименьших квадратов.

В результате выполнения программного продукта были получены следующие погрешности:

1. ВЫВОДЫ

Таким образом, в ходе выполнения лабораторной работы была освоена интерполяция функций с помощью интерполяционных многочленов Лагранжа и Ньютона. Была также составлена программа, на трех тестовых примерах проверена правильность её работы. Для тестовых примеров и вычисленных по условию многочленов были построены графики. Также, бычислено значение функции в точке x=0.47 согласно заданному варианту. В заданном варианте задания погрешность в многочленах разных типов не превышает 10-12.