Министерство образования Республики Беларусь
Учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ
к лабораторной работе 6
на тему

Интерполяционные многочлены

Выполнил: студент группы 053502
Герчик Артём Вадимович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Оглавление

Цели выполнения задания 3
Краткие теоретические сведения 4
Программная реализация 11
Полученные результаты 16
Выводы 19

Цели выполнения задания

1) Изучить интерполяцию функций с помощью интерполяционных многочленов Лагранжа и Ньютона

Краткие теоретические сведения

АППРОКСИМАЦИЯ И ИНТЕРПОЛЯЦИЯ ФУНКЦИЙ

Из математического анализа известно, что в окрестности точки x_0 любую n раз непрерывно дифференцируемую функцию можно аппроксимировать (приблизить) ее многочленом Тейлора:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)(x - x_0)^k}{k!},$$

причем

$$f(x_0) = P_n(x_0),$$
$$f'(x_0) = P_n'(x_0),$$
....
$$f^{(n)}(x_0) = P_n^{(n)}(x_0).$$

Очевидно, такая аппроксимация во многих отношениях является очень хорошей, но она имеет локальный характер, т.е. хорошо аппроксимирует функцию только вблизи точки x_0 . Это главный недостаток аппроксимации с помощью многочлена Тейлора.

Если речь идет об аппроксимации функции на отрезке, применяются другие методы.

Пусть $f(x) \in C[a,b]$ – непрерывная функция. Рассмотрим задачу аппроксимации (приближения) ее более простой функцией (обычно многочленом).

Известно из математического анализа, что в силу теоремы Вейерштрасса, любую функцию можно с какой угодно точностью приблизить многочленом по норме $\|f(x)\| = \max\limits_{a \le x \le b} |f(x)|$ пространства C[*a, b*], т. е. в смысле равномерной сходимости. Но существуют и другие нормы:

$$\|f(x)\| = \int_a^b |f(x)|dx \quad \text{или} \quad \|f(x)\| = \sqrt{\int_b^a |f(x)|^2 dx}\,.$$

Тогда $\|f(x) - P(x)\| < \varepsilon$ означает, что площадь или усредненная площади фигуры, заключенной между графиками функции *f(x)* и многочлена *P(x)*, должна быть меньше ε (заданной точности).

Возможен и другой подход, когда в качестве аппроксимирующей функции берут многочлен или другую достаточно простую функцию, значения которых совпадают со значениями исходной функции в заданных заранее точках, так называемых узлах. Такого рода приближение функций имеет свое собственное название - интерполяция.

Интерполяционный многочлен

Пусть *f(x)* – функция, непрерывная на отрезке [*a,b*].

Выберем на этом отрезке точки, называемые *узлами интерполяции*:

$a \le x_0 < x_1 < ... < x_n \le b$.

Предположим, что известны значения функции в узлах интерполяции:

$f(x_k) = y_k, \quad k = 0,1,...,n$.

Ставится задача найти многочлен $P_n(x)$ такой, что

$$P_n(x_k) = y_k, \qquad \forall k = 0,1,...,n. \tag{7.1}$$

Такой многочлен $P_n(x)$ называется *интерполяционным многочленом*, а задача его нахождения – *задачей интерполяции.*

Покажем, что задача интерполяции имеет решение, причем единственное.

Пусть $P_n(x) = \sum_{k=0}^{n} a_k x^{n-k}$.

Тогда для определения коэффициентов многочлена из условия (7.1) получаем систему:

$$\begin{cases} a_0 x_0^n + a_1 x_0^{n-1} + ... + a_n = y_0 \\ a_0 x_1^n + a_1 x_1^{n-1} + ... + a_n = y_1 \\ .... \\ a_0 x_n^n + a_1 x_n^{n-1} + ... + a_n = y_n . \end{cases}$$

Ее определитель Δ с точностью до знака совпадает с так называемым определителем Вандермонда.

$$W(x_0,...,x_n) = \begin{vmatrix} 1 & 1 & ... & 1 \\ x_0 & x_1 & ... & x_n \\ x_0^2 & x_1^2 & ... & x_n^2 \\ ... & ... & ... & ... \\ x_0^n & x_1^n & ... & x_n^n \end{vmatrix} = \prod_{i<j}(x_j - x_i) \ne 0 .$$

Поскольку все x_i различны, определитель Δ отличен от нуля, и, следовательно, система имеет единственное решение. Отсюда вытекает существование и единственность интерполяционного многочлена.

Погрешность интерполяции.

Обозначим

$R_n(x) = f(x) - P_n(x)$ и будем искать ее оценку.

Пусть $f(x) \in C^{n+1}[a,b]$. Положим $R_n(x) = \omega(x) r(x)$,

где $\omega(x) = (x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_n)$.

Зафиксируем произвольную точку *x*, отличную от узлов интерполяции x_i, $i = \overline{0,n}$, и построим вспомогательную функцию:

$$F(t) = P_n(t) + \omega(t) r(x) - f(t), \qquad a \le t \le b \ . \tag{7.2}$$

Очевидно, $F(x) = 0$ и, кроме того $F(x_k) = 0, \qquad k = \overline{0,n}$.

Таким образом, функция *F(t)* имеет по крайней мере (*n+2*) нуля на отрезке [*a,b*]. Применим теорему Ролля, по которой между каждой парой нулей функции находится по крайней мере один нуль производной этой функции. Тогда производная $F'(t)$ имеет по крайней мере (*n+1*) нулей на данном интервале (a,b). Продолжая рассуждение, получим в итоге, что $F^{(n)}(t)$ имеет, по крайней мере, два нуля, а $F^{(n+1)}(t)$ − один нуль в некоторой точке ξ на (a,b).

Продифференцируем равенство (7.2) (*n+1*) раз и подставим $t = \xi$. Получим

$$F^{(n+1)}(\xi) = (n+1)! \cdot r(x) - f^{(n+1)}(\xi) = 0.$$

Откуда $r(x) = \dfrac{f^{(n+1)}(\xi)}{(n+1)!}$.

Тогда

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega(x),$$

где $\xi \in [a,b]$ (очевидно формула напоминает остаток формулы Тейлора в форме Лагранжа). В итоге имеем оценку погрешности интерполяции:

$$|R_n(x)| \le \frac{M_{n+1}}{(n+1)!} |\omega(x)|, \qquad \text{где} \qquad M_{n+1} = \max_{a \le x \le b} \left| f^{(n+1)}(x) \right|.$$

Интерполяционный многочлен Лагранжа

Пусть даны узлы на отрезке $[a,b]$, $a \le x_0 < x_1 < ... < x_n \le b$, и значения функции $F(x)$ в узлах

$$f(x_i) = y_i, \qquad i = \overline{0,n}.$$

Пусть $\omega(x) = (x - x_0)(x - x_1) \cdot ... \cdot (x - x_n)$,

$$\omega_j(x) = (x - x_0) \cdot ... \cdot (x - x_{j-1})(x - x_{j+1}) \cdot ... \cdot (x - x_n),$$

т. е. $\omega_j(x) = \dfrac{\omega(x)}{x - x_j}$.

Положим $l_j(x) = \dfrac{\omega_j(x)}{\omega_j(x_j)}$,

т. е. $$l_j(x) = \frac{(x - x_0) \cdot ... \cdot (x - x_{j-1})(x - x_{j+1}) \cdot ... \cdot (x - x_n)}{(x_j - x_0) \cdot ... \cdot (x_j - x_{j-1})(x_j - x_{j+1}) \cdot ... \cdot (x_j - x_n)}.$$

Очевидно $l_j(x_i) = \begin{cases} 0, & при\ i \ne j \\ 1, & при\ i = j. \end{cases}$

Построим многочлен $L_n(x) = \sum_{j=0}^{n} l_j(x) y_j$.

Легко видеть, что $L_n(x_i) = l_i(x_i) y_i = 1 \cdot y_i = y_i$, $i = \overline{0,n}$, т.е. это интерполяционный многочлен. Его называют интерполяционным многочленом Лагранжа.

Интерполяционный многочлен Ньютона

Пусть $x_0, x_1, ..., x_n$ − набор узлов интерполирования,

$y_0, y_1, ..., y_n$ − значения функции $f(x)$ в узлах.

Величину $\Delta y_k = y_{k+1} - y_k$ называют конечной разностью первого порядка в *к*-м узле.

Аналогично определяются конечные разности высших порядков.

$$\Delta^2 y_k = \Delta y_{k+1} - \Delta y_k = y_{k+2} - y_{k+1} - (y_{k+1} - y_k) = y_{k+2} - 2y_{k+1} + y_k$$

....

$$\Delta^i y_k = \Delta^{i-1} y_{k+1} - \Delta^{i-1} y_k = \sum_{i=0}^{n} (-1)^{n-i} C_n^i y_{k+i} \ \Delta^i y_k = \Delta^{i-1} y_{k+1} - \Delta^{i-1} y_k = \sum_{i=0}^{n} (-1)^{n-i} C_n^i y_{k+i}.$$

Конечные разности обычно считают по схеме:

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$
x_0 x_1 x_2 x_3	y_0 y_1 y_2 y_3	$\Delta y_0 = y_1 - y_0$ $\Delta y_1 = y_2 - y_1$ $\Delta y_2 = y_3 - y_2$	$\Delta^2 y_0 = \Delta y_1 - \Delta y_0$ $\Delta^2 y_1 = \Delta y_2 - \Delta y_1$	$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0$

Разделенной разностью первого порядка называется выражение

$$f_1(x_k, x_{k+1}) = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{\Delta y_k}{\Delta x_k}.$$

Разделенной разностью второго порядка называется выражение

$$f_2(x_k, x_{k+1}, x_{k+2}) = \frac{f_1(x_{k+1}, x_{k+2}) - f_1(x_k, x_{k+1})}{x_{k+2} - x_k}$$ и т. д.

Пусть x – любая точка отрезка, не совпадающая с узлами. Тогда

$$f_1(x, x_0) = \frac{y_0 - f(x)}{x_0 - x},$$

откуда $f(x) = y_0 + f_1(x, x_0)(x - x_0)$. (7.3)

Далее $f_2(x, x_0, x_1) = \dfrac{f_1(x_0, x_1) - f_1\ (x, x_0)}{x_1 - x}$,

откуда $f_1(x, x_0) = f_1(x_0, x_1) + f_2\ (x, x_0, x_1)(x - x_1)$.

Подставляя в (7.3), получаем

$$f(x) = y_0 + f_1(x_0, x_1)(x - x_0) + f_2\ (x, x_0, x_1)(x - x_0)(x - x_1). \qquad (7.4)$$

Далее $f_3(x, x_0, x_1, x_2) = \dfrac{f_2(x_0, x_1, x_2) - f_2\ (x, x_0, x_1)}{x_2 - x}$,

откуда $f_2(x, x_0, x_1) = f_2(x_0, x_1, x_2) + f_3\ (x, x_0, x_1, x_2)(x - x_2)$.

Подставляя в (4), имеем:

$$f(x) = y_0 + f_1(x_0, x_1)(x - x_0) + f_2\ (x, x_0, x_2)(x - x_0)(x - x_1) +$$
$$+ f_3\ (x, x_0, x_1, x_2)(x - x_0)(x - x_1)(x - x_2). \qquad (7.5)$$

Продолжая процесс, получим:

$$f(x) = N_n(x) + f_{n+1}(x, x_0, ..., x_n)(x - x_0)...(x - x_n),$$

где $N_n(x) = y_0 + f_1(x_0, x_1)(x - x_0) + ... + f_n\ (x_0, ..., x_n)(x - x_0)...(x - x_{n-1})$.

Очевидно, при $x = x_i, \quad \forall i = \overline{0, n}, \quad f(x_i) = N_n(x_i), \quad i = \overline{0, n}$,

т. е. $N_n(x)$ − интерполяционный многочлен. Его называют интерполяционным многочленом Ньютона.

Достоинство интерполяционного многочлена Ньютона: он удобен при расширении интерполяции и добавлении узлов.

Недостаток: в какой-то степени он сложнее в подсчете конечных разностей по сравнению с многочленом Лагранжа.

Аппроксимация методом наименьших квадратов

Пусть дана функция $f(x)$ на отрезке [*a, b*].

Разобьем отрезок с помощью узлов

$a \le x_0 < x_1 < ... < x_n \le b$.

Пусть $y_0, y_1,..., y_n$ − значение функции *f(x)* в узлах.

Если n − большое число, то интерполяционный $L_n(x)$ − многочлен высокой степени. Зачастую неудобно использовать многочлены очень высокой степени. Очевидно, мы можем отказаться от использования части узлов и тем самым понизить степень интерполяционного многочлена, но тогда теряется часть информации. Поэтому вместо интерполяционного многочлена будем искать многочлен $P_m(x)$ меньшей степени *(m<n)*, такой что сумма

$$\sum_{i=0}^{n}[f(x_i) - P_m(x_i)]^2$$

принимает наименьшее значение. Данный многочлен называется многочленом наилучшего приближения по методу наименьших квадратов.

Положим

$$P_m(x) = a_0 x^m + ... + a_m$$

и будем искать решение задачи

$$S(a_0,..., a_m) = \sum_{i=0}^{n}[a_0 x_i^m + ... + a_{m-1}x_i + a_m - y_i]^2 \to \min.$$

Приравнивая к нулю производные S, получим систему линейных уравнений для определения коэффициентов a_i:

$$\frac{\partial S}{\partial a_0} = 2\sum_{i=0}^{n}[a_0 x_i^m + ... + a_m - y_i]\cdot x_i^m = 0$$

$$\frac{\partial S}{\partial a_1} = 2\sum_{i=0}^{n}[a_0 x_i^m + ... + a_m - y_i]\cdot x_i^{m-1} = 0$$

....

$$\frac{\partial S}{\partial a_{m-1}} = 2\sum_{i=0}^{n}[a_0 x_i^m + ... + a_m - y_i]\cdot x_i = 0$$

$$\frac{\partial S}{\partial a_m} = 2\sum_{i=0}^{n}[a_0 x_i^m + ... + a_m - y_i]\cdot 1 = 0 \ .$$

Отсюда получается

$$\begin{cases} a_0(\sum_{i=0}^{n} x_i^{2m}) + a_1 \sum_{i=0}^{n} x_i^{2m-1} + ... + a_m \sum_{i=0}^{n} x_i^m = \sum_{i=0}^{n} y_i x_i^m \\ a_0(\sum_{i=0}^{n} x_i^{2m-1}) + ... + a_m \sum_{i=0}^{n} x_i^{m-1} = \sum_{i=0}^{n} y_i x_i^{m-1} \\ .... \\ a_0(\sum_{i=0}^{n} x_i^n) + ... + a_m \sum_{i=0}^{n} 1 = \sum_{i=0}^{n} y_i \end{cases}$$

− нормальная система для определения коэффициентов $a_0, a_1, ..., a_n$.

Для решения нормальной системы обычно используется следующая таблица:

i	x_i	x_i^2	………	x_i^{2m}	y_i	$y_i x_i$	………	$y_i x_i^m$
0	x_0	x_0^2		x_0^{2m}	y_0	$y_0 x_0$		$y_0 x_0^m$
1	x_1	x_1^2		x_1^{2m}	y_1	$y_1 x_1$		$y_1 x_1^m$
.	.	.		.	.	.		.
.	.	.		.	.	.		.
.	.	.		.	.	.		.
n	x_n	x_n^2		x_n^{2m}	y_n	$y_n x_n$		$y_n x_n^m$
	$\sum_{i=0}^{n} x_i$	$\sum_{i=0}^{n} x_i^2$	...	$\sum_{i=0}^{n} x_i^{2m}$	$\sum_{i=0}^{n} y_i$	$\sum_{i=0}^{n} y_i x_i$	...	$\sum_{i=0}^{n} y_i x_i^m$

Когда $m \le n$, можно показать, что нормальная система имеет единственное решение, которое действительно дает минимальное значение для функции *S*. Получив решения нормальной системы $a_0, ..., a_n$, строим многочлен наилучшего приближения по методу наименьших квадратов.

В частном случае, когда $m=n$, многочлен $P_n(x)$ переходит в интерполяционный многочлен.

Программная реализация

```
# import matplotlib.pyplot as plt
import numpy as np


def input_values():
    x = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8,
0.9, 1.0]
    p = [0.0, 0.41, 0.79, 1.13, 1.46, 1.76, 2.04, 2.3,
2.55, 2.79, 3.01]
    k = 7
    m = 3.5
    y = [(p[i] + ((-1) ** k) * m) for i in range(len(x))]
    dots = list(zip(x, y))

    # dots = [(0, 0), (1, 0)]
    # dots = [(-1, 0), (0, 1), (1, 0)]
    # dots = [(1, 1), (2, 2), (3, 3), (4, 4), (5, 6)]

    return dots


dots = input_values()
(x, y) = map(list, zip(*dots))


def calculate_lagrange_polynom(dots):
    n = len(dots)
    (x, y) = map(list, zip(*dots))
    polynom = np.poly1d([0])
    for i in range(n):
        p = np.poly1d([1])
        for j in range(n):
            if j != i:
```

```
                p *= np.poly1d([1, -x[j]]) / (x[i] -
x[j])
        polynom += y[i] * p
    return polynom


def calculate_divided_differences(xs):
    n = len(xs)
    diffs = [[None for j in range(n - i)] for i in
range(n)]
    for i in range(n):
        diffs[i][0] = y[i]
    for j in range(1, n):
        for i in range(n - j):
            diffs[i][j] = ((diffs[i][j - 1] - diffs[i +
1][j - 1]) / (xs[i] - xs[i + j]))
    return diffs


def calculate_inaccurany(xs, xdot):
    n = len(xs)
    diffs = calculate_divided_differences(xs)
    maxdiff = 0.0
    for i in range(len(diffs)):
        for j in range(len(diffs[i])):
            maxdiff = max(maxdiff, abs(diffs[i][j]))
    w = 1
    for i in range(n):
        w *= xdot - xs[i]
    f = 1
    for i in range(1, n + 1 + 1):
        f *= i
    R = maxdiff * w / f
    return R
```

```
def calculate_newton_polynom(dots):
    n = len(dots)
    (x, y) = map(list, zip(*dots))

    diffs = calculate_divided_differences(x)

    polynom = np.poly1d([0])
    for i in range(n):
        p = np.poly1d([1])
        for j in range(i):
            p *= np.poly1d([1, -x[j]])
        polynom += p * diffs[0][i]

    return polynom


def calculate_least_square_polynom(dots, m=None):
    n = len(dots) - 1
    if m is None:
        m = n
    assert 0 <= m <= n
    return np.poly1d(np.polyfit(*map(list, zip(*dots)),
m))


def main():
    print("Интерполяционные многочлены \n")

    print("Исходные (x,y) = ", dots, '\n')

    lagrange = calculate_lagrange_polynom(dots)
    print("Многочлен Лагранджа = ")
    print(lagrange, '\n')
```

```
    newton = calculate_newton_polynom(dots)
    print("Многочлен Ньютона = ")
    print(newton, '\n')

    squares = calculate_least_square_polynom(dots)
    print("Многочлен наименьших квадратов = ")
    print(squares, '\n')

    dot_to_calculate = 0.47
    print(f"Многочлен Лагранджа от ({dot_to_calculate}) =
", lagrange(dot_to_calculate))
    print(f"Многочлен Ньютона от ({dot_to_calculate}) =
", newton(dot_to_calculate))
    print(f"Многочлен наименьших квадратов от
({dot_to_calculate}) = ", squares(dot_to_calculate))
    print(f"|Лаграндж({dot_to_calculate}) -
Ньютон({dot_to_calculate})| =",
          abs(lagrange(dot_to_calculate) -
newton(dot_to_calculate)))
    print(f"|Лаграндж({dot_to_calculate}) - Наименьшие
квадраты({dot_to_calculate})| =",
          abs(lagrange(dot_to_calculate) -
squares(dot_to_calculate)))
    print(f"|Ньютон({dot_to_calculate}) - Наименьшие
квадраты({dot_to_calculate})| =",
          abs(newton(dot_to_calculate) -
squares(dot_to_calculate)))

    print(f"Погрешность от ({dot_to_calculate}) = ",
calculate_inaccurany(x, dot_to_calculate))

    # plotdots = 10 ** 4
    #
    # plt.plot(x, y, 'og')
    #
```

```
    # xplot = np.linspace(min(x), max(x), plotdots)
    #
    # yplot = [squares(xdot) for xdot in xplot]
    # plt.plot(xplot, yplot, 'y')
    #
    # yplot = [lagrange(xdot) for xdot in xplot]
    # plt.plot(xplot, yplot, 'b')
    #
    # yplot = [newton(xdot) for xdot in xplot]
    # plt.plot(xplot, yplot, 'r--')
    #
    # plt.show()


if __name__ == '__main__':
    main()
```

Полученные результаты

Тестовый пример 1

Исходные (x,y) = [(-1, 0), (0, 1), (1, 0)]

Многочлен Лагранджа =

$-1\,x^{2} + 1$

Многочлен Ньютона =

$-1\,x^{2} + 1$

Многочлен наименьших квадратов =

$-1\,x^{2} + 1$

Многочлен Лагранджа от (0.47) = 0.7791
Многочлен Ньютона от (0.47) = 0.7791
Многочлен наименьших квадратов от (0.47) = 0.7790999999999999
|Лаграндж(0.47) - Ньютон(0.47)| = 0.0
|Лаграндж(0.47) - Наименьшие квадраты(0.47)| = 1.1102230246251565e-16
|Ньютон(0.47) - Наименьшие квадраты(0.47)| = 1.1102230246251565e-16
Погрешность от (0.47) = -0.015257374999999998

Тестовый пример 2

Исходные (x,y) = [(1, 1), (2, 2), (3, 3), (4, 4), (5, 6)]

Многочлен Лагранджа =

$0.04167\,x^{4} - 0.4167\,x^{3} + 1.458\,x^{2} - 1.083\,x + 1$

Многочлен Ньютона =

$0.04167\,x^{4} - 0.4167\,x^{3} + 1.458\,x^{2} - 1.083\,x + 1$

Многочлен наименьших квадратов =

$$0.04167 x^4 - 0.4167 x^3 + 1.458 x^2 - 1.083 x + 1$$

Многочлен Лагранджа от (0.47) = 0.7717527837500028
Многочлен Ньютона от (0.47) = 0.7717527837500001
Многочлен наименьших квадратов от (0.47) = 0.7717527837499569
|Лаграндж(0.47) - Ньютон(0.47)| = 2.6645352591003757e-15
|Лаграндж(0.47) - Наименьшие квадраты(0.47)| = 4.5852210917018965e-14
|Ньютон(0.47) - Наименьшие квадраты(0.47)| = 4.318767565791859e-14
Погрешность от (0.47) = -0.27338802207750007

ЗАДАНИЕ
Вариант 7

Построить интерполяционные многочлены в форме Лагранжа и Ньютона. Оценить погрешность. Вычислить значение функции в точке 0.47 с помощью интерполяционного многочлена и многочлена наилучшего приближения. Сравнить значения.

x	*0.0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
y	*-3.5*	*-3.09*	*-2.71*	*-2.37*	*-2.04*	*-1.74*	*-1.46*	*-1.2*	*-0.95*	*-0.71*	*-0.49*

Результаты:

Интерполяционные многочлены

Исходные (x,y) = [(0.0, -3.5), (0.1, -3.09), (0.2, -2.71), (0.3, -2.37), (0.4, -2.04), (0.5, -1.74), (0.6, -1.46), (0.7, -1.2000000000000002), (0.8, -0.9500000000000002), (0.9, -0.71), (1.0, -0.4900000000000002)]

Многочлен Лагранджа =

$3279 x^{10} - 1.682e{+}04 x^9 + 3.714e{+}04 x^8 - 4.611e{+}04 x^7 + 3.532e{+}04 x^6$
$- 1.719e{+}04 x^5 + 5268 x^4 - 966.3 x^3 + 92.75 x^2 + 0.6282 x - 3.5$

Многочлен Ньютона =

$3279 x^{10} - 1.682e{+}04 x^9 + 3.714e{+}04 x^8 - 4.611e{+}04 x^7 + 3.532e{+}04 x^6$
$- 1.719e{+}04 x^5 + 5268 x^4 - 966.3 x^3 + 92.75 x^2 + 0.6282 x - 3.5$

Многочлен наименьших квадратов =

$3279 x^{10} - 1.682e{+}04 x^9 + 3.714e{+}04 x^8 - 4.611e{+}04 x^7 + 3.532e{+}04 x^6$
$- 1.719e{+}04 x^5 + 5268 x^4 - 966.3 x^3 + 92.75 x^2 + 0.6282 x - 3.5$

Многочлен Лагранджа от (0.47) = -1.8273479203003964
Многочлен Ньютона от (0.47) = -1.8273479202912304
Многочлен наименьших квадратов от (0.47) = -1.8273479202919984
|Лаграндж(0.47) - Ньютон(0.47)| = 9.166001291305292e-12
|Лаграндж(0.47) - Наименьшие квадраты(0.47)| = 8.397949002869609e-12
|Ньютон(0.47) - Наименьшие квадраты(0.47)| = 7.680522884356833e-13
Погрешность от (0.47) = 2.58052446071294e-13

Выводы

Таким образом, в ходе выполнения лабораторной работы была освоена интерполяция функций с помощью интерполяционных многочленов Лагранжа и Ньютона. Составлена компьютерная программа, на тестовых примерах проверена правильность её работы, вычислено значение функции в точке согласно заданному варианту.