Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий Кафедра прикладной математики

Отчет защищен с оценкой

Преподаватель

(подпись)

«\_ » 2023 г.

Отчет

По лабораторной работе №3

по дисциплине «Вычислительные алгоритмы»

Студент группы ПИ-02 Чередов Р.А.

Преподаватель Проскурин А.В.

Барнаул 2023

# Задание

Составить программу для построения интерполяционного многочлена Лагранжа (Ньютона). Программа должна работать в двух режимах:

а) по заданной таблице значений функции определять приближенное значение функции в некоторой точке, вводимой пользователем;

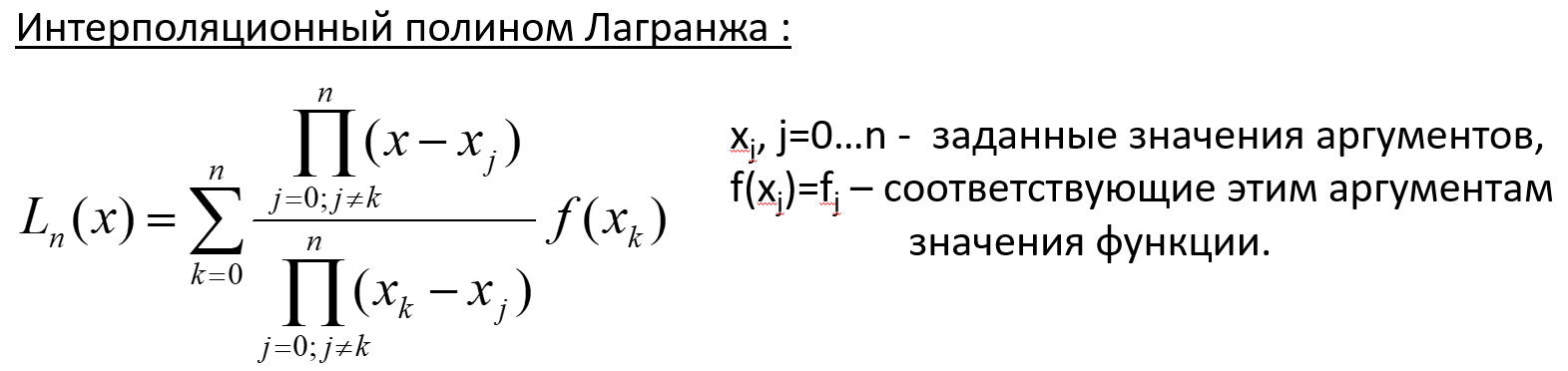
б) по заданной аналитически функции y = f (x) и массиву значений аргумента (массив читается из файла) вычислить таблицу значений функции. Используя полученную таблицу, построить интерполяционный многочлен после чего нарисовать графики функции y = f (x) и интерполяционного многочлена. Найти величину максимального отклонения заданной функции от интерполирующей в точках, в которых вычислялись значения функций для построения графиков (для изображения графиков число точек, в которых вычисляются функции должно быть существенно больше, чем число узлов интерполяции).

Исследовать путем проведения вычислительных экспериментов влияние количества и расположения узлов интерполирования, участков интерполирования на величину погрешности интерполирования. В качестве функций, для которых проводится анализ, помимо придуманных Вами функций рекомендуется рассмотреть y = |x| при |x| <= 1, y = e

-x при |x| <= 4, y = sinx при |x| <= Pi.

# Теория

Интерполяционный многочлен **Лагранжа** — многочлен минимальной степени, принимающий заданные значения в заданном наборе точек, то есть решающий задачу интерполяции.



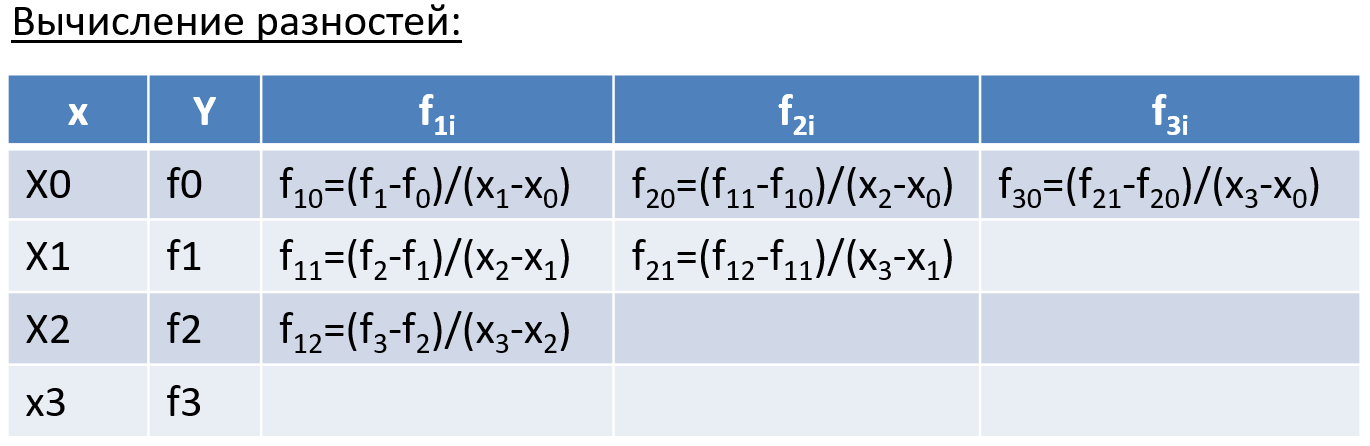
**Примечание**: при изменении количества узлов интерполяции n все слагаемые полинома Лагранжа необходимо пересчитывать.

Интерполяционные формулы **Ньютона** — формулы вычислительной математики, применяющиеся для полиномиального интерполирования.

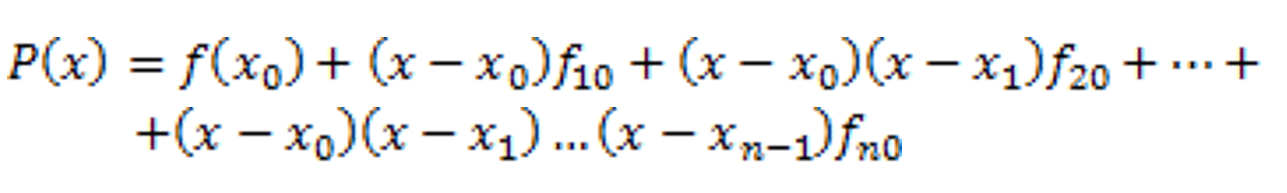
По разделенным разностям (k-1)-го порядка вычисляем разделенные разности порядка k:



Эта формула удобна для последовательных вычислений в таблице.



Интерполяционный полином Ньютона:



**Интерполяционный многочлен Ньютона является другой формой записи интерполяционного многочлена Лагранжа**.

При увеличении n ранее вычисленные коэффициенты не изменяются.

Погрешность интерполяции:







При равномерном шаге

(h = xi+1 – xi = const) :

# 

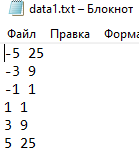
Погрешность можно оценить через разность заданной функции и полиномиального многочлена:

# 

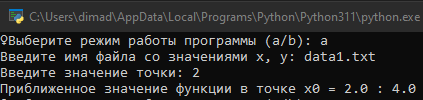
# Отклонение разность между полученной функции и интерпаляционной.

# Решение

import numpy as np  
import matplotlib.pyplot as plt  
from numpy import \*  
from os import system  
def lagrange\_interpolation(x, y, x\_interp):  
 *"""  
 Вычисляет интерполяционный многочлен Лагранжа.  
 Аргументы:  
 x -- массив узлов интерполяции  
 y -- массив значений функции в узлах интерполяции  
 x\_interp -- точка, в которой требуется вычислить значение интерполяционного  
многочлена  
 Возвращает:  
 Значение интерполяционного многочлена в точке x\_interp  
 """* # Вычисляем количество узлов интерполяции  
 n=len(x)  
 # Инициализируем массив базисных многочленов Лагранжа  
 lj=np.ones(n)  
 # Вычисляем базисные многочлены Лагранжа  
 for i in range(n):  
 for j in range(n):  
 if i != j:  
 lj[i] \*= (x\_interp - x[j]) / (x[i] - x[j])  
 # Вычисляем значение интерполяционного многочлена Лагранжа  
 y\_interp = np.sum(y \* lj)  
 return y\_interp  
def read\_file\_x(filename):  
 *"""  
 Функция для чтения массива значений аргумента из файла.  
 """* with open(filename, 'r') as f:  
 x = [float(line) for line in f]  
 return np.array(x)  
def read\_file\_xy(filename):  
 *"""  
 Функция для чтения значений x, y из файла.  
 """* x = []  
 y = []  
 with open(filename, 'r') as f:  
 for line in f:  
 data = line.split()  
 x.append(float(data[0]))  
 y.append(float(data[1]))  
 return np.array(x), np.array(y)  
def plot\_graphs(x, y, x\_interp, y\_interp, x\_intersect, y\_intersect):  
 *"""  
 Функция для построения графиков и вычисления максимального отклонения.  
 """* # Строим график исходной функции  
 plt.plot(x, y, label='Исходная функция')  
 # Строим график интерполяционного многочлена  
 plt.plot(x\_interp, y\_interp, label='Интерполяционный многочлен')  
 # Отмечаем точки пересечения  
 plt.plot(x\_intersect, y\_intersect, 'o', color='red', label='Узлы')  
 # Определяем максимальное отклонение  
 max\_err = max\_error(y\_interp, y)  
 print('Максимальное отклонение: ', max\_err)  
 # Добавляем легенду и выводим графики на экран  
 plt.legend()  
 plt.show()  
def max\_error(y1, y2):  
 *"""  
 Функция для вычисления максимального отклонения между двумя массивами.  
 """* max\_err = 0  
 for i in range(len(y1)):  
 err = abs(y1[i] - y2[i])  
 if err > max\_err:  
 max\_err = err  
 return max\_err  
while (True):  
 \_ = system('cls')  
 # Выбираем режим работы программы  
 mode = input('Выберите режим работы программы (a/b): ')  
 if mode == 'a':  
 # Считываем данные из файла  
 filename = input('Введите имя файла с массивом значений аргумента: ')  
 x1 = read\_file\_x(filename)  
 # Запрашиваем ввод функции f(x)  
 func = input('Введите выражение: ')  
 # Вычисляем значения функции  
 y1 = []  
 for x in x1:  
 y1.append(eval(func))  
 # Создаем массивы с множеством точек для построения графика функции y = f(x)  
 x\_forgraph = np.arange(x1[0], x1[-1], 0.001)  
 y\_forgraph = []  
 for x in x\_forgraph:  
 y\_forgraph.append(eval(func))  
 # Строим интерполяционный многочлен  
 x\_interp = np.arange(x1[0], x1[-1], 0.001)  
 y\_interp = []  
 for xi in x\_interp:  
 y\_interp.append(lagrange\_interpolation(x1, y1, xi))  
 # Строим графики  
 plot\_graphs(x\_forgraph, y\_forgraph, x\_interp, y\_interp, x1, y1)  
 elif mode == 'b':  
 # Считываем данные из файла  
 filename = input('Введите имя файла со значениями x, y: ')  
 # Задаем таблицу точек  
 x2, y2 = read\_file\_xy(filename)  
 # Задаем точку, в которой нужно вычислить значение многочлена  
 x0 = float(input('Введите значение точки: '))  
 # Строим интерполяционный многочлен  
 y0 = lagrange\_interpolation(x2, y2, x0)  
 print('Приближенное значение функции в точке x0 =', x0, ':', y0)  
 else:  
 print('Некорректный ввод.')

* 1. Тест №1 Исходные данные:

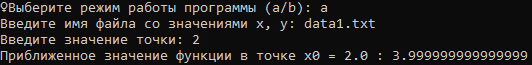
Результат:



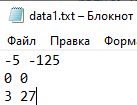
* 1. Тест №2 Исходные данные:



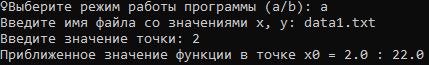
Результат:



* 1. Тест №3 Исходные данные:



Результат:



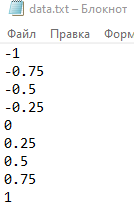
# Второй режим работы

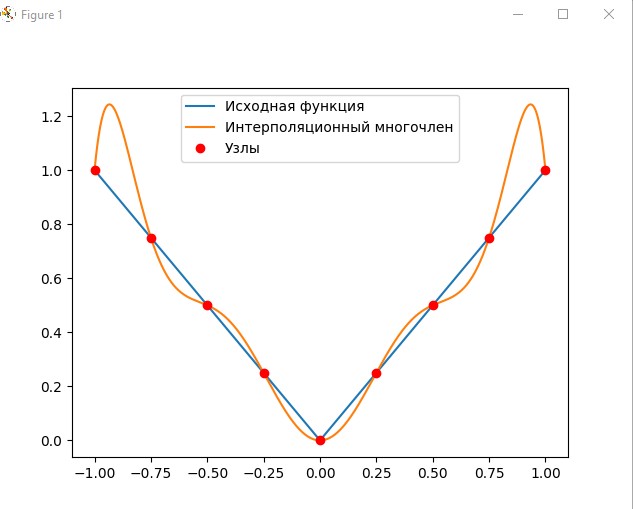
**3.1** | | | |

Исходные данные:

Результат:

Тест №1

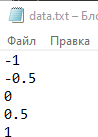




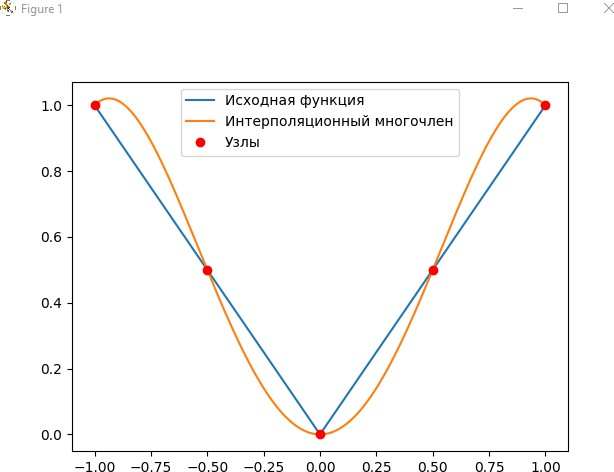


Тест №2

Исходные данные:



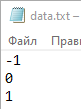
Результат:



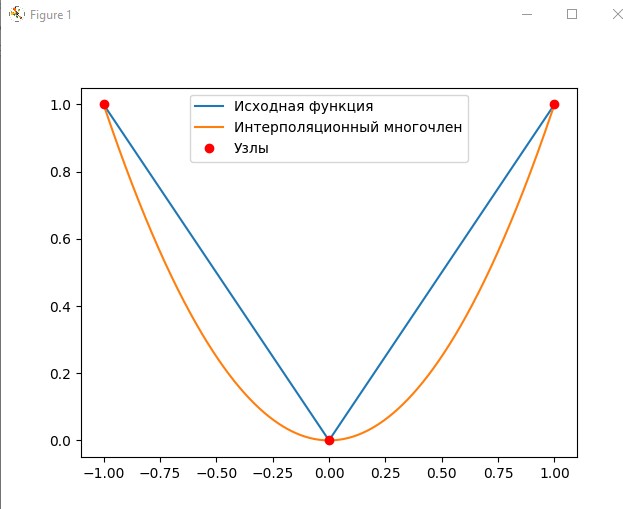


Тест №3

Исходные данные:



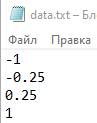
Результат:



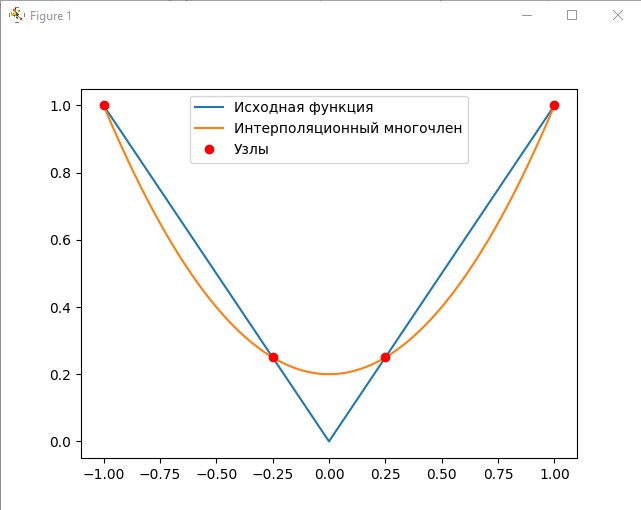


Тест №4

Исходные данные:



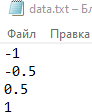
Результат:



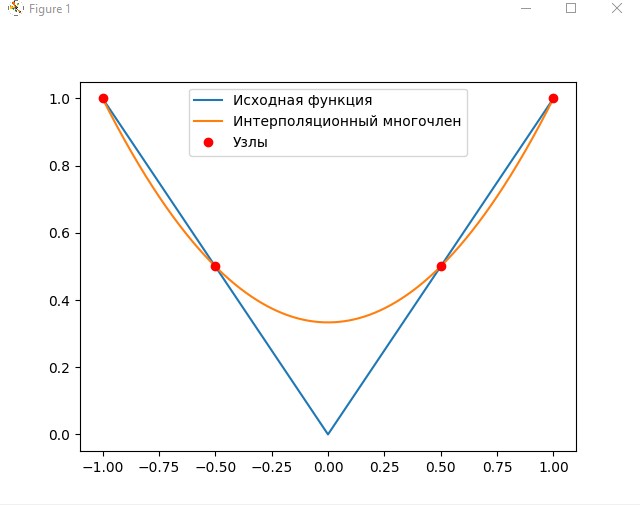


Тест №5

Исходные данные:



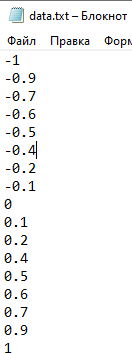
Результат:



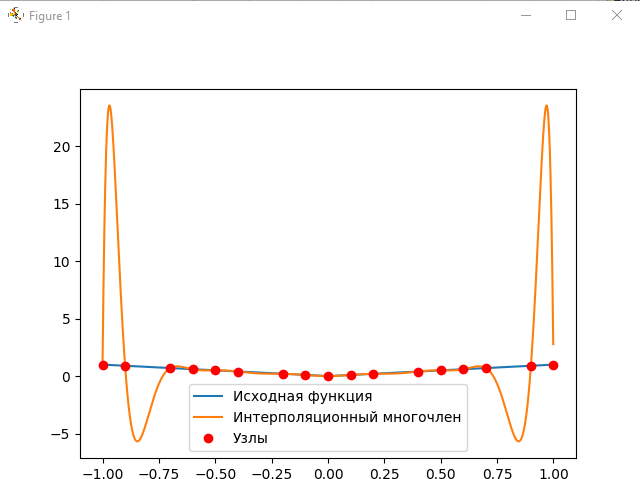


Тест №6

Исходные данные:



Результат:



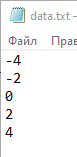


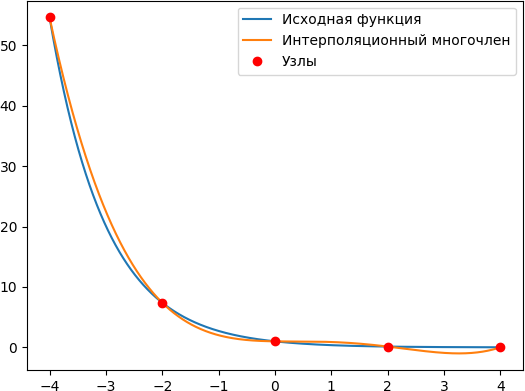
**3.2** | |

Исходные данные:

Результат:

Тест №1

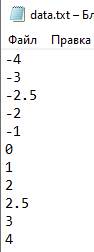




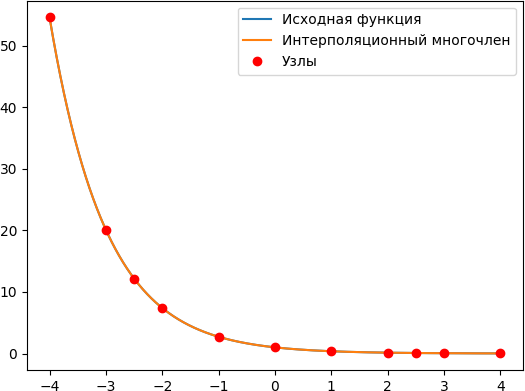


Тест №2

Исходные данные:

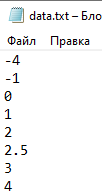


Результат:

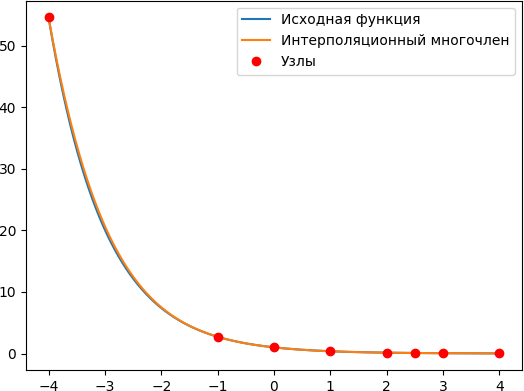


Тест №3

Исходные данные:



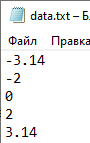
Результат:



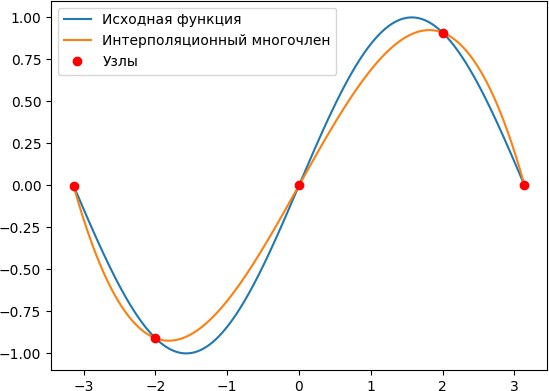
# 3.3 | |

Исходные данные:

Тест №1

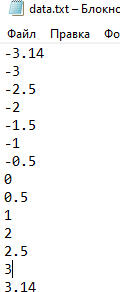


Результат:

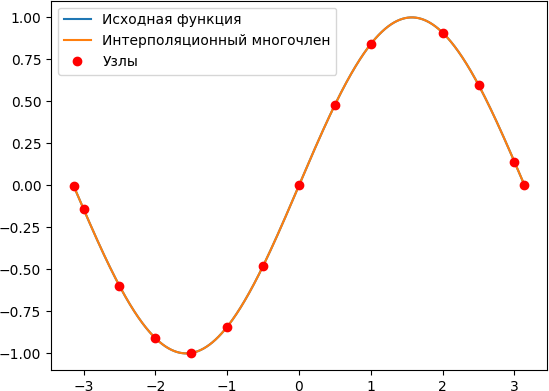


Тест №2

Исходные данные:



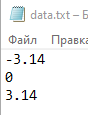
Результат:





Тест №3

Исходные данные:



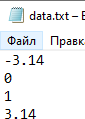
Результат:



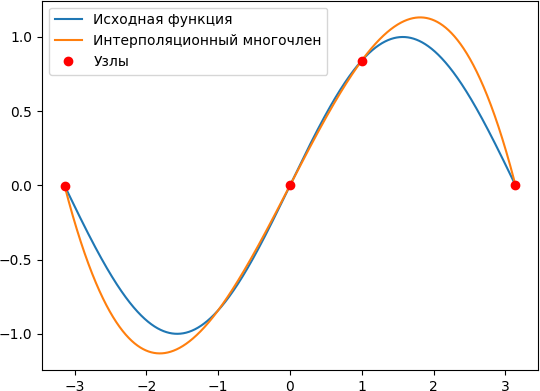


Тест №3

Исходные данные:



Результат:





# Вывод

Для различных функций увеличение или уменьшение точек интерполяции по разному влияло на результат и максимальное отклонение. Например, для функции:



увеличение точек интерполяции привело к значительному увеличению максимального отклонения, а для функции:



наоборот, уменьшило величину максимального отклонения.

Изменение расположения точек интерполирования уменьшило максимальное отклонение на одном участке, но увеличило на другом.