|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**Лабораторная работа 1.1**

**«Приближение функций»**

**ДИСЦИПЛИНА: «Методы обработки информации»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_Петроченков И. А.\_\_)  (Подпись) (Ф.И.О.) |
| Проверил(а): | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_Никитенко У. В.\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2023 г. | | |

# Цели работы:

Сформировать практические навыки описания и анализа используемых алгоритмов; создания программной реализации системы с заданными свойствами.

**Задачи:**

Восстановление (доопределение) функции, заданной на дискртеном множестве точек.

**Вариант №**

**Задание 1.** Исследовать ходимость интерполяционного процесса в точке x\* ∈[a,b] для функции y= f(x), x∈[a,b], если функцию интерполировать одним из способов:

1) многочленом Лагранжа с произвольными узлами;

2) многочленом Лагранжа с равноотстоящими узлами;

3) многочленом Лагранжа, записанным в барицентрическом виде;

4) многочленом Ньютона через разделенные разности;

**Задание 2.** Вычислить практическую величину погрешности и теоретическую величину погрешности в п.1)-4) задания 1. Минимизировать погрешность интерполяции, если функцию **y= f(x),** на [a,b] приблизить многочленами Лагранжа 1, 2, 3-й степени.

**Результаты работы:**

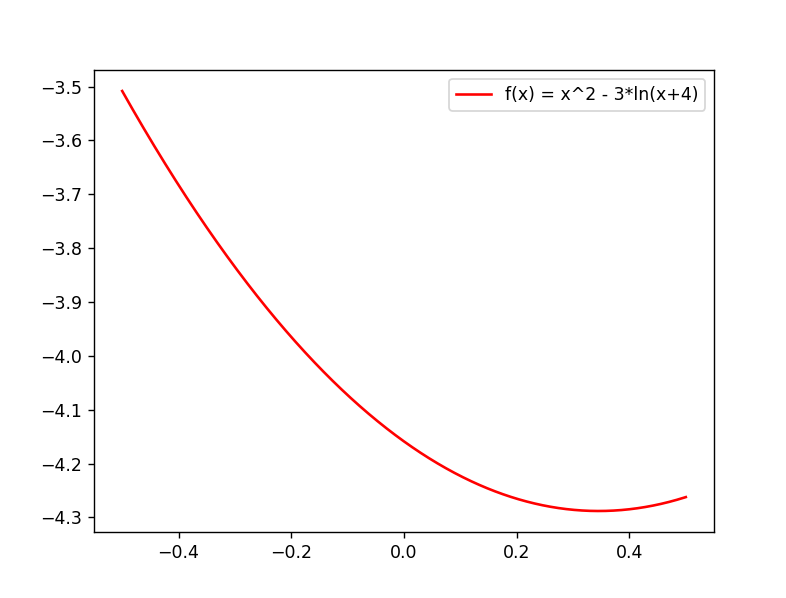
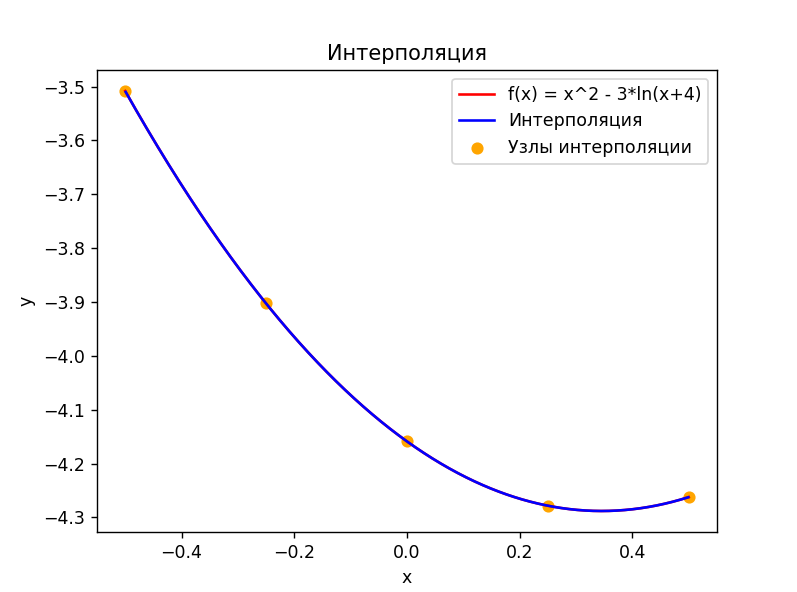
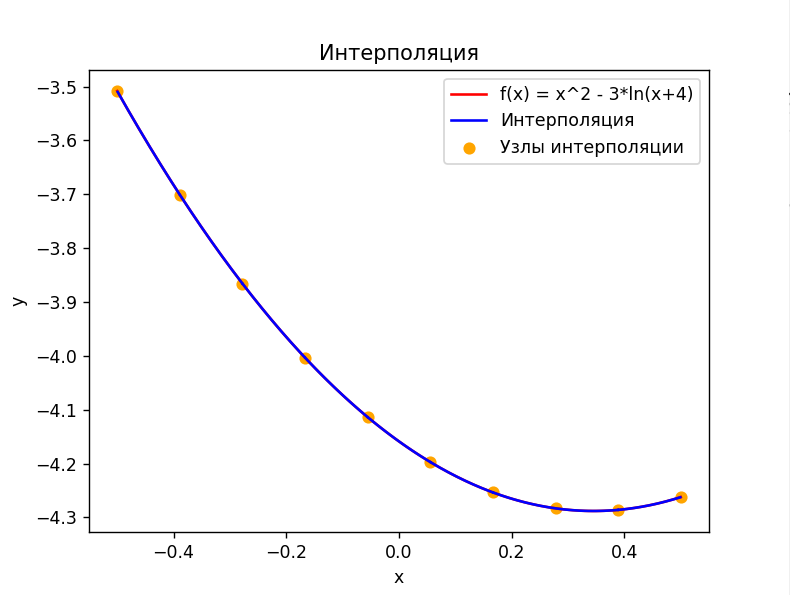
****

Рис. 1 График исходной функции





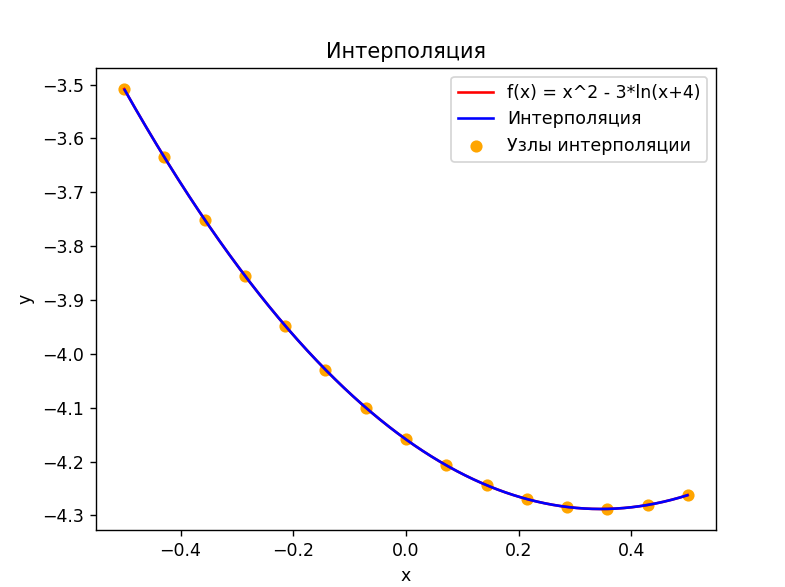


Рис. 2 Интерполяция в барицентрической форме с разным числом узлов

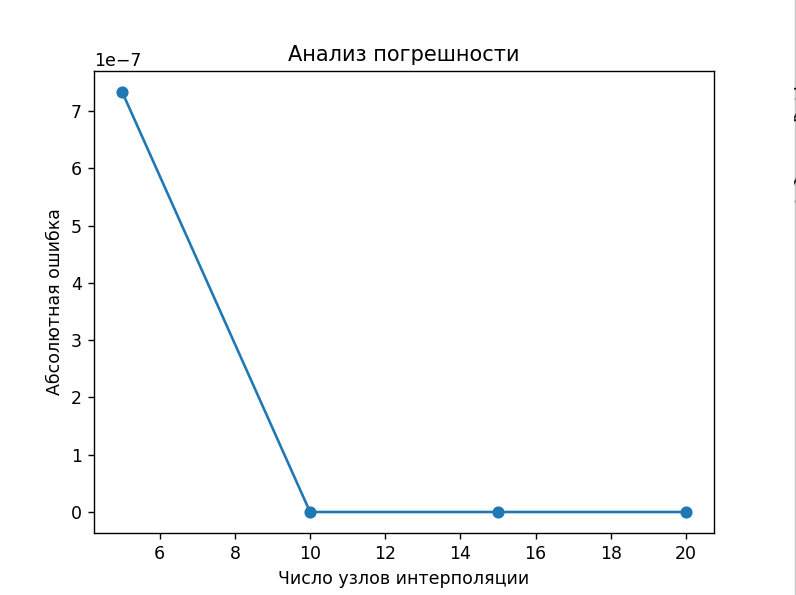


Рис. 3 Анализ сходимости интерполяционной функции

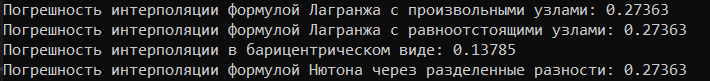


Рис. 4 Оценка погрешности вычислений

**Листинг кода:**

# def lagrange\_interpolation(x\_current, nodes, values)->float:

# res = 0;

# for i in range(len(values)):

# cur = 1;

# for j in range(len(nodes)):

# if i != j:

# cur \*= (x\_current - nodes[j]) / (nodes[i] - nodes[j]);

# res += values[i] \* cur;

# return res;

import numpy as np;

import matplotlib.pyplot as plt;

from scipy.interpolate import lagrange

def lagrange\_interpolation(x\_current, nodes, function)->float:

res = 0;

for i in range(len(nodes)):

cur = 1;

for j in range(len(nodes)):

if i != j:

cur \*= (x\_current - nodes[j]) / (nodes[i] - nodes[j]);

res += function(nodes[i]) \* cur;

return res;

def newton\_interpolation(x\_current, nodes, function)->float:

def divided\_difference(nodes, function):

summ = 0;

for i in range(len(nodes)):

mult = function(nodes[i]);

for j in range (len(nodes)):

if i != j:

mult /= (nodes[i] - nodes[j]);

summ += mult;

return summ;

res = function(nodes[0]);

for i in range(1, len(nodes)):

mult = divided\_difference(nodes[0:i+1], function);

for j in range(i):

mult \*= (x\_current - nodes[j]);

res += mult;

return res;

def barycentric\_interpolation(x\_current, nodes, function)->float:

def find\_beta(current\_index, nodes):

divider = 1;

for i in range(len(nodes)):

if current\_index != i:

divider \*= (nodes[current\_index] - nodes[i]);

return (1/divider);

numinator = 0;

denominator = 0;

for i in range(len(nodes)):

numinator += (find\_beta(i, nodes) \* function(nodes[i])) / (x\_current - nodes[i]);

denominator += find\_beta(i, nodes) / (x\_current - nodes[i]);

return numinator/denominator;

if \_\_name\_\_ == "\_\_main\_\_":

def f(x): return x\*\*2 - 3\*np.log(x+4)

a = -0.5;

b = 0.5;

num\_nodes\_list = [5, 10, 15, 20];

x\_start = 0.1;

errors = [];

x\_values = np.linspace(a, b, 1000)

plt.plot(x\_values, f(x\_values), color = 'red', label='f(x) = x^2 - 3\*ln(x+4)')

plt.legend();

plt.show();

for num\_nodes in num\_nodes\_list:

nodes = np.linspace(a, b, num\_nodes)

values = f(nodes)

y\_start = barycentric\_interpolation(x\_start, nodes, f)

errors.append(abs(f(x\_start) - y\_start))

nodes = np.linspace(a, b, num\_nodes)

#values = f(nodes)

x\_values = np.linspace(a, b, 1000)

interpolated\_values = barycentric\_interpolation(x\_values, nodes, f)

y\_start = barycentric\_interpolation(x\_start, nodes, f)

plt.plot(x\_values, f(x\_values), color = 'red', label='f(x) = x^2 - 3\*ln(x+4)')

plt.plot(x\_values, interpolated\_values, color = 'blue', label='Интерполяция')

plt.scatter(nodes, values, color='orange', label='Узлы интерполяции')

plt.xlabel('x')

plt.ylabel('y')

plt.title('Интерполяция')

plt.legend()

#plt.grid(True)

plt.show()

plt.plot(num\_nodes\_list, errors, marker='o')

plt.xlabel('Число узлов интерполяции')

plt.ylabel('Абсолютная ошибка')

plt.title('Анализ погрешности')

#plt.grid(True)

plt.show()

n = 1;

x\_nodes = np.linspace(a, b, n+1)

y\_nodes = f(x\_nodes)

#lag\_poly\_arbitrary = lagrange(x\_nodes, y\_nodes)

lag\_poly\_arbitrary = lambda x: lagrange\_interpolation(x, x\_nodes, f);

newton\_poly = lambda x: newton\_interpolation(x, x\_nodes, f)

x\_values = np.linspace(a, b, 1000)

f\_values = f(x\_values)

lagrange\_interpolated\_values = lag\_poly\_arbitrary(x\_values)

lagrange\_practical\_error = np.max(np.abs(f\_values - lagrange\_interpolated\_values))

x\_uniform = np.linspace(a, b, n+1)

y\_uniform = f(x\_uniform)

#lag\_poly\_uniform = lagrange(x\_uniform, y\_uniform)

lag\_poly\_uniform = lambda x: lagrange\_interpolation(x, x\_uniform, f);

x\_chebyshev = 0.5 \* (a + b) + 0.5 \* (b - a) \* np.cos((2\*np.arange(n+1) + 1) / (2\*(n+1)) \* np.pi)

y\_chebyshev = f(x\_chebyshev)

#lag\_poly\_barycentric = lagrange(x\_chebyshev, y\_chebyshev)

lag\_poly\_barycentric = lambda x: barycentric\_interpolation(x, x\_chebyshev, f);

newton\_interpolated\_values = newton\_poly(x\_values)

newton\_practical\_error = np.max(np.abs(f\_values - newton\_interpolated\_values))

x\_values = np.linspace(a, b, 1000)

f\_values = f(x\_values)

lagrange\_uniform\_interpolated\_values = lag\_poly\_uniform(x\_values)

lagrange\_uniform\_practical\_error = np.max(np.abs(f\_values - lagrange\_uniform\_interpolated\_values))

lagrange\_barycentric\_interpolated\_values = lag\_poly\_barycentric(x\_values)

lagrange\_barycentric\_practical\_error = np.max(np.abs(f\_values - lagrange\_barycentric\_interpolated\_values))

print("Погрешность интерполяции формулой Лагранжа с произвольными узлами:", "{:.5f}".format(lagrange\_practical\_error))

print("Погрешность интерполяции формулой Лагранжа с равноотстоящими узлами:", "{:.5f}".format(lagrange\_uniform\_practical\_error))

print("Погрешность интерполяции в барицентрическом виде:", "{:.5f}".format(lagrange\_barycentric\_practical\_error))

print("Погрешность интерполяции формулой Нютона через разделенные разности:", "{:.5f}".format(newton\_practical\_error))

**Вывод:**

Сформированы практические навыки описания и анализа используемых алгоритмов; создана программная реализация системы с заданными свойствами.