МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра вычислительной математики

Отчёт

Лабораторная работа №1-4

“ Интерполирование функций”

Вариант № 12

Снежко Льва Владимировича
студента 3 курса, 3 группы
специальности «Информатика»
дисциплина «Численные методы»
Преподаватель: Будник А.М.

Минск, 2025

Постановка задачи

Рассмотрим набор различный точек на отрезке [a, b]:

$$x_0 < x_1 < \dots < x_n\ ,\ x_i \in [a, b],\ \text{i} = \overline{0, n}.$$

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

Требуется восстановить значение функции $\text{f}(x) = \alpha_j e^x + (1 - \alpha_j)\text{sinx}$ в других точках $x^*, x^{**}, x^{***} \in [a, b]$:

$$[\text{a, b}] = [\alpha_j,\ 1 + \alpha_j],\ \text{где}$$

$\alpha_j = 0.1 + 0.05 \cdot$ j = [j = 12 – номер варианта] = $0.1 + 0.05 \cdot 12 = 0.7.$

Таким образом имеем:

$$[\text{a, b}] = [0.7, 1.7]$$

$$\text{f}(x) = 0.7e^x + 0.7\text{sinx}$$

Необходимо интерполировать эту функцию:

1. Многочленом Ньютона $P_{10}(x)$ на равномерной сетке;
2. Многочленом Ньютона $P_{10}(x)$ на Чебышёвской сетке;
3. Кубическим сплайном;
4. Методом наименьших квадратов.

Для каждого из методов необходимо:

- Вычислить значения интерполяционного многочлена в точках $x^*,\ x^{**},\ x^{***}$;
- Вычислить остаток интерполирования в точках $x^*,\ \ x^{**},\ x^{***}$;
- Вычислить истинную погрешность $r_{n,\text{ист.}}$;
- Сравнить и проанализировать полученные результаты.

Алгоритм решения

1. Интерполирование многочленом Ньютона на равномерной сетке

Пусть x_i заданы равномерно на отрезке [0.5,1.5]:

$$x_i = \alpha_i + \mathrm{i}\mathrm{h},\ \textit{где}$$

$$\mathrm{h} = \frac{1}{n},\ \ \mathrm{i} = \overline{0,\ n},\ \ \mathrm{n} = 10,\ \textit{т.е.}$$

$\mathrm{h} = 0.1$

$x_i = 0.7 + 0.1 \cdot i,\ ,\ \mathrm{i} = \overline{0,\ 10}$

3 точки восстановления:

$$x^* = x_0 + \frac{2}{3}\mathrm{h} = 0.5 + \frac{2}{3} \cdot 0.1 = 0.766667,$$

$$x^{**} = x_{\frac{n}{2}} + \frac{1}{2}\mathrm{h} = x_5 + \frac{1}{2}\mathrm{h} = 1 + 0.5 \cdot 0.1 = 0.65,$$

$$x^{***} = x_n - \frac{1}{3}\mathrm{h} = 1.5 - \frac{1}{3} \cdot 0.1 = 1.666667$$

Интерполяционный многочлен в форме Ньютона будем искать в следующем виде:

$$P_n(x) = \sum_{i=0}^{n} \alpha_i \omega_i(x) = \alpha_0 + \alpha_1\,(x - x_0) + \alpha_2\,(x - x_0)(x - x_1) \dots \alpha_n\,(x - x_0) \dots (x - x_{n-1}),$$

где $\omega_i(x) = \prod_{j=0}^{i-1}(x - x_j),\ i = \overline{0{,}10}$

Коэффициенты ИМ удобно вычислять по определению разделённых разностей:

$$\alpha_i = \mathrm{f}[x_0, \dots, x_i],\ \ i = \overline{0{,}10}$$

$$\mathrm{f}[\mathrm{x}_0,\ \dots,\ \mathrm{x}_\mathrm{i}] = \frac{\mathrm{f}[\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{i}] - \mathrm{f}[\mathrm{x}_0, \dots, \mathrm{x}_{\mathrm{i}-1}]}{\mathrm{x}_\mathrm{i} - \mathrm{x}_0},\quad \mathrm{i} = \overline{1, 10};$$

$$\mathrm{f}[\mathrm{x}_\mathrm{j}] = \mathrm{f}(\mathrm{x}_\mathrm{j}),\quad \mathrm{j} = \overline{0, 10}$$

путём построения треугольной таблицы следующего вида:

$$
\begin{array}{cccccc}
x_0, f[x_0] & & & & & \\
 & f[x_0, x_1] & & & & \\
x_1, f[x_1] & & f[x_0, x_1, x_2] & & & \\
 & f[x_1, x_2] & & \ddots & & \\
x_2, f[x_2] & & f[x_0, x_1, x_2] & & f[x_0, \ldots, x_{n-1}] & \\
\vdots & f[x_2, x_3] & \vdots & & & f[x_0, \ldots, x_n] \\
\vdots & \vdots & \vdots & & f[x_1, \ldots, x_n] & \\
\vdots & \vdots & \vdots & \cdot^{\cdot^{\cdot}} & & \\
x_{n-1}, f[x_{n-1}] & \vdots & f[x_{n-2}, x_{n-1}, x_n] & & & \\
 & f[x_{n-1}, x_n] & & & & \\
x_n, f[x_n] & & & & &
\end{array}
$$

Остаток интерполирования в точках x^*, x^{**}, x^{***} вычислим по следующей формуле:

$$r_n(x) = f[x_0, \ldots, x_n, x]\, \omega_{n+1}(x), \quad x \in \{x^*, x^{**}, x^{***}\}$$

Истинную погрешность вычислим так:

$$r_{n,\text{ист}}(x^*) = f(x^*) - P_n(x^*)$$

Листинг кода

```
%pip install tabulate
%pip install sympy
import numpy as np
import math
from tabulate import tabulate
import matplotlib.pyplot as plt
import pandas as pd
import sympy as sp

j = 12
n = 10
alpha_j = 0.1 + 0.05*j
h = 1/n
def f(x):
    return alpha_j * math.exp(x) + (1 - alpha_j) * math.sin(x)

# Шаг 1. Построим исходную таблицу
x_vals = np.array([alpha_j + i * h for i in range(n+1)])
f_vals = np.array([f(x_) for x_ in x_vals])
x_star = np.array([x_vals[0] + 2/3*h, x_vals[n // 2] / 2 + 0.5 * h, x_vals[-1] -
1/3*h])
f_star = np.array([f(x_) for x_ in x_star])
print(tabulate(zip(x_vals, y_vals), headers=['x', 'f(x)']))
print('\nСпециальные точки')
print(tabulate(zip(x_star, f_star), headers=['x*', 'f(x*)']))

# Таблица значений функции
table = pd.DataFrame({"x_i": x_vals, "f(x_i)": f_vals})
table_transposed = table.T

# Точки для проверки интерполяции
x_star1 = x_star[0]
x_star2 = x_star[1]
x_star3 = x_star[2]

f_x_star = f_star[0]
f_x_star2 = f_star[1]
f_x_star3 = f_star[2]

def compute_newton_coefficients(x_vals, y_vals):
    """
    Возвращает список коэффициентов интерполяционного многочлена Ньютона
    с использованием рекурсивного определения разделённых разностей.
    """
    n = len(x_vals)
    # Создаём таблицу размером n x n
    dd_table = [y_vals.copy()]  # f[x_i]

    for level in range(1, n):
        prev_column = dd_table[-1]
        curr_column = []
```

```
        for i in range(n - level):
            numerator = prev_column[i + 1] - prev_column[i]
            denominator = x_vals[i + level] - x_vals[i]
            curr_column.append(numerator / denominator)
        dd_table.append(curr_column)

    # Коэффициенты Ньютона — это верхние элементы каждого столбца
    return [dd_table[i][0] for i in range(n)]

newton_coeffs = compute_newton_coefficients(x_vals, f_vals)
def newton_interpolation(x_vals, y_vals, x, coef):
    """
    Вычисляет значение интерполяционного многочлена Ньютона в точке x
    с использованием рекурсивной формулы:
    P_{n+1}(x) = P_n(x) + alpha_{n+1} * omega_{n+1}(x)
    """
    result = coef[0]
    omega = 1.0
    for i in range(1, len(coef)):
        omega *= (x - x_vals[i - 1])
        result += coef[i] * omega
    return result


P_x_star = newton_interpolation(x_vals, f_vals, x_star1, newton_coeffs)
P_x_star2 = newton_interpolation(x_vals, f_vals, x_star2, newton_coeffs)
P_x_star3 = newton_interpolation(x_vals, f_vals, x_star3, newton_coeffs)

# Результаты интерполяции
data = {
    "Точка": ["x*", "x**", "x***"],
    "Значение x": [x_star1, x_star2, x_star3],
    "f(x)": [f_x_star, f_x_star2, f_x_star3],
    "P(x) (полином)": [P_x_star, P_x_star2, P_x_star3]
}

df = pd.DataFrame(data)

# Производная (n+1)-го порядка
x = sp.Symbol('x')
f_sym = alpha_j * sp.exp(x) + (1 - alpha_j) * sp.sin(x)
f_derivative = sp.diff(f_sym, x, n + 1)

# Максимум абсолютного значения производной на отрезке [0.7, 1.7]
f_derivative_abs = sp.lambdify(x, sp.Abs(f_derivative), 'numpy')
x_test = np.linspace(0.7, 1.7, 1000)
M_max = np.max(f_derivative_abs(x_test))

# Истинная погрешность
r_x_star = f_x_star - P_x_star
r_x_star2 = f_x_star2 - P_x_star2
r_x_star3 = f_x_star3 - P_x_star3

# Оценка погрешности по неравенству
factorial = math.factorial(n + 1)
x_stars = [x_star1, x_star2, x_star3]
r_x_stars = [r_x_star, r_x_star2, r_x_star3]
error_bound_stars = []
```

```
for x_val in x_stars:
    prod_term = np.prod([abs(x_val - xi) for xi in x_vals])
    error_bound = M_max / factorial * prod_term
    error_bound_stars.append(error_bound)

# Проверка выполнения неравенства
is_error_bound_stars_valid = [
    abs(r_x_stars[i]) <= error_bound_stars[i] for i in range(3)
]

# Таблица ошибок
error_table = pd.DataFrame({
    "Точка": ["x*", "x**", "x***"],
    "Значение x": [x_star1, x_star2, x_star3],
    "r истинная": [abs(r) for r in r_x_stars],
    "оценка погрешности": error_bound_stars,
    "M = max|f^(n+1)(x)|": [M_max] * 3,
    "Неравенство выполняется?": is_error_bound_stars_valid
})

# Вывод таблиц
display(table_transposed)
display(df)
display(error_table)
```

Результаты

Таблица значений:

	0	1	2	3	4	5	6	7	8	9	10
x_i	0.700000	0.800000	0.90000	1.000000	1.100000	1.200000	1.300000	1.400000	1.500000	1.600000	1.700000
f(x_i)	1.602892	1.773085	1.95672	2.155239	2.370278	2.603694	2.857575	3.134275	3.436431	3.766995	4.129263

	Точка	Значение x	f(x)	P(x) (полином)
0	x*	0.766667	1.714927	1.714927
1	x**	0.650000	1.522435	1.522435
2	x***	1.666667	4.004765	4.004765

	Точка	Значение x	r истинная	оценка погрешности	M = max|f^(n+1)(x)|	Неравенство выполняется?
0	x*	0.766667	1.021405e-13	1.343614e-13	2.78765	True
1	x**	0.650000	3.516742e-12	4.688495e-12	2.78765	True
2	x***	1.666667	2.433609e-13	2.863766e-13	2.78765	True

Коэффициенты интерполяционного многочлена (разделенные разности):

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_10
0	1.602892	1.701933	0.672075	0.240342	0.081633	0.016566	0.0023	0.000369	0.000059	0.000006	5.661216e-07

Таблица разделённых разностей

	x_i	f[x0..x0]	f[x0..x1]	f[x0..x2]	f[x0..x3]	f[x0..x4]	f[x0..x5]	f[x0..x6]	f[x0..x7]	f[x0..x8]	f[x0..x9]	f[x0..x10]
0	0.7	1.602892e+00	1.773085	1.956720	2.155239	2.370278	2.603694	2.857575	3.134275	3.436431	3.766995	4.129263
1	0.8	1.701933e+00	1.836348	1.985183	2.150398	2.334151	2.538816	2.766998	3.021559	3.305639	3.622678	NaN
2	0.9	6.720749e-01	0.744178	0.826076	0.918765	1.023320	1.140911	1.272809	1.420399	1.585195	NaN	NaN
3	1.0	2.403423e-01	0.272996	0.308962	0.348518	0.391970	0.439659	0.491968	0.549321	NaN	NaN	NaN
4	1.1	8.163336e-02	0.089916	0.098889	0.108629	0.119224	0.130772	0.143382	NaN	NaN	NaN	NaN
5	1.2	1.656594e-02	0.017946	0.019480	0.021189	0.023095	0.025221	NaN	NaN	NaN	NaN	NaN
6	1.3	2.299651e-03	0.002558	0.002849	0.003176	0.003543	NaN	NaN	NaN	NaN	NaN	NaN
7	1.4	3.685241e-04	0.000416	0.000467	0.000524	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	1.5	5.893869e-05	0.000065	0.000071	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	1.6	6.451632e-06	0.000007	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	1.7	5.661216e-07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Анализ

Порядок $r_{ист}$ не превышает погрешности интерполяции в контрольных точках. Видно, что погрешность возрастает в точках, близких к краям отрезка и имеет наиболее точное значение в точке находящейся близко к середине отрезка. Разница погрешностей в зависимости от точки восстановления связана с возрастанием многочлена ω(x) и расположением контрольной точки относительно ближайшего узла.

$\omega_{n+1}(x^{*})$ = -0.000002;

$\omega_{n+1}(x^{**})$ = 0.000064;

$\omega_{n+1}(x^{***})$ = 0.000123.

2. Интерполирование многочленом Ньютона на Чебышёвской сетке

Пусть теперь x_i заданы на отрезке [0.5,1.5] следующим образом :

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} cos\left(\frac{\pi(2i+1)}{2n+1}\right) \text{ , } где$$

$i = \overline{0, n}$,

a = 0,7,

b = 1,7,

n = 10, т. е.

$$x_i = 1.2 + \frac{1}{2} cos\left(\frac{\pi(2i+1)}{21}\right) \text{ , } i = \overline{0,\ 10},$$

Точки восстановления те же:

- $x^* = 0.770944,$
- $x^{**} = 1.184737,$
- $x^{***} = 1.662389$

Остаток интерполирования в точках x^*, x^{**}, x^{***} оценим по следующим формулам:

$$|r_n(x)| \le \frac{M}{(n+1)!} |\omega_{n+1}(x)| \le \frac{M}{(n+1)!} \cdot 2\left(\frac{b-a}{4}\right)^{n+1}, \quad (v. 1),$$

$$|r_n(x)| \le |f[x_0, \ \ldots, x_n, x]| \cdot |\, \omega_{n+1}(x)| \le$$

$$\le |f[x_0, \ \ldots, x_n, x]| \cdot 2\left(\frac{b - a}{4}\right)^{n+1}, \qquad (v. 2), \qquad \text{где}$$

$x \in \{x^*, \ x^{**}, x^{***}\}$

$M = \left\|f^{(n+1)}\right\|_{C[a,b]} = \max\limits_{a \le x \le b} |f^{(n+1)}(x)|,$

n = 10,

a = 0.5, b = 1.5.

Листинг кода

```
import numpy as np
import math
from tabulate import tabulate
import matplotlib.pyplot as plt
import pandas as pd
import sympy as sp
j = 12
n = 10
a = 0.7
b = 1.7
alpha_j = 0.1 + 0.05*j
h = 1/n

def f(x):
    return alpha_j * math.exp(x) + (1 - alpha_j) * math.sin(x)

x_vals = np.array(sorted([(a+b) / 2 + (b-a) / 2 * math.cos(math.pi * (2*i+1) /
(2*n+2)) for i in range(n+1)]))
f_vals = np.array([f(x_) for x_ in x_vals])
x_star = np.array([x_vals[0] + 2/3*h, x_vals[n // 2] + h / 2, x_vals[n] - h / 3])
f_star = np.array([f(x_) for x_ in x_star])

print(tabulate(zip(x_vals, f_vals), headers=['x', 'f(x)']))
print('\nСпециальные точки')
print(tabulate(zip(x_star, f_star), headers=['x*', 'f(x*)']))
# Таблица значений функции
table = pd.DataFrame({"x_i": x_vals, "f(x_i)": f_vals})
table_transposed = table.T

# Точки для проверки интерполяции
x_star1 = x_star[0]
x_star2 = x_star[1]
x_star3 = x_star[2]

f_x_star = f_star[0]
f_x_star2 = f_star[1]
f_x_star3 = f_star[2]

def compute_newton_coefficients(x_vals, y_vals):
    """
    Возвращает список коэффициентов интерполяционного многочлена Ньютона
    с использованием рекурсивного определения разделённых разностей.
    """
    n = len(x_vals)
    # Создаём таблицу размером n x n
    dd_table = [y_vals.copy()]  # f[x_i]

    for level in range(1, n):
        prev_column = dd_table[-1]
```

```python
        curr_column = []
        for i in range(n - level):
            numerator = prev_column[i + 1] - prev_column[i]
            denominator = x_vals[i + level] - x_vals[i]
            curr_column.append(numerator / denominator)
        dd_table.append(curr_column)

    # Коэффициенты Ньютона — это верхние элементы каждого столбца
    return dd_table, [dd_table[i][0] for i in range(n)]

dd_table, newton_coeffs = compute_newton_coefficients(x_vals, f_vals)


def newton_interpolation(x_vals, y_vals, x, coef):
    """
    Вычисляет значение интерполяционного многочлена Ньютона в точке x
    с использованием рекурсивной формулы:
    P_{n+1}(x) = P_n(x) + alpha_{n+1} * omega_{n+1}(x)
    """
    result = coef[0]
    omega = 1.0
    for i in range(1, len(coef)):
        omega *= (x - x_vals[i - 1])
        result += coef[i] * omega
    return result, omega

omegas = [1.0,1.0,1.0]
P_x_star, omegas[0] = newton_interpolation(x_vals, f_vals, x_star1,
newton_coeffs)
P_x_star2, omegas[1] = newton_interpolation(x_vals, f_vals, x_star2,
newton_coeffs)
P_x_star3, omegas[2] = newton_interpolation(x_vals, f_vals, x_star3,
newton_coeffs)

omega_frame = pd.DataFrame(omegas, index=[f'omega_{i}' for i in
range(len(omegas))])
# Результаты интерполяции
data = {
    "Точка": ["x*", "x**", "x***"],
    "Значение x": [x_star1, x_star2, x_star3],
    "f(x)": [f_x_star, f_x_star2, f_x_star3],
    "P(x) (полином)": [P_x_star, P_x_star2, P_x_star3]
}

n = len(newton_coeffs)
dd_frame = pd.DataFrame(dd_table, columns=[f"f[x0..x{i}]" for i in
range(len(newton_coeffs))])
dd_frame.insert(0, "x_i", x_vals)
coeff_frame = pd.DataFrame(newton_coeffs, index=[f"a_{i}" for i in range(n)]).T
df = pd.DataFrame(data)
```

```python
# Производная (n+1)-го порядка
x = sp.Symbol('x')
f_sym = alpha_j * sp.exp(x) + (1 - alpha_j) * sp.sin(x)
f_derivative = sp.diff(f_sym, x, n + 1)

# Максимум абсолютного значения производной на отрезке [0.7, 1.7]
f_derivative_abs = sp.lambdify(x, sp.Abs(f_derivative), 'numpy')
x_test = np.linspace(0.7, 1.7, 1000)
M_max = np.max(f_derivative_abs(x_test))

# Истинная погрешность
r_x_star = f_x_star - P_x_star
r_x_star2 = f_x_star2 - P_x_star2
r_x_star3 = f_x_star3 - P_x_star3

# Оценка погрешности по неравенству
factorial = math.factorial(n + 1)
x_stars = [x_star1, x_star2, x_star3]
r_x_stars = [r_x_star, r_x_star2, r_x_star3]
error_bound_stars = []

for x_val in x_stars:
    prod_term = np.prod([abs(x_val - xi) for xi in x_vals])
    error_bound = M_max / factorial * prod_term
    error_bound_stars.append(error_bound)

# Проверка выполнения неравенства
is_error_bound_stars_valid = [
    abs(r_x_stars[i]) <= error_bound_stars[i] for i in range(3)
]

# Таблица ошибок
error_table = pd.DataFrame({
    "Точка": ["x*", "x**", "x***"],
    "Значение x": [x_star1, x_star2, x_star3],
    "r истинная": [abs(r) for r in r_x_stars],
    "оценка погрешности": error_bound_stars,
    "M = max|f^(n+1)(x)|": [M_max] * 3,
    "Неравенство выполняется?": is_error_bound_stars_valid
})

# Вывод таблиц
display(table_transposed)
display(df)
display(error_table)
display(dd_frame)
display(coeff_frame)
display(omega_frame)
```

Результаты

Таблица значений:

	0	1	2	3	4	5	6	7	8	9	10
x_i	0.705089	0.745184	0.822125	0.929680	1.059134	1.200000	1.340866	1.470320	1.577875	1.654816	1.694911
f(x_i)	1.611250	1.678212	1.812509	2.014017	2.280290	2.603694	2.967752	3.343927	3.691247	3.961424	4.110004

Коэффициенты интерполяционного многочлена (разделенные разности):

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_10
1.61125	1.670114	0.643691	0.225181	0.077884	0.016057	0.002234	0.000362	0.000059	0.000006	5.671354e-07

Интерполяция в контрольных точках:

Точка	Значение x	f(x)	P(x) (полином)
x*	0.771756	1.723712	1.723712
x**	1.250000	2.727935	2.727935
x***	1.661577	3.986094	3.986094

Точка	Значение x	r истинная	оценка погрешности v1	оценка погрешности v2	M = max|f^(n+1)(x)|
x*	0.771756	2.442491e-14	4.623513e-14	0.000002	3.870417
x**	1.250000	2.442491e-14	4.623513e-14	0.000002	3.870417
x***	1.661577	1.065814e-14	4.623513e-14	0.000002	3.870417

Анализ

Нетрудно видеть, что для x^*, x^{**}, x^{***} $r_{\text{ист}}$ не превосходит оценки сверху. Также можно заметить, что истинная погрешность в контрольных точках на чебышевской сетке улучшилась по сравнению с равномерной сеткой.

Многочлен $\omega_{n+1}(x)$ в контрольных точках принимает значения:

omega_0	4.907741e-07
omega_1	9.560534e-07
omega_2	5.198465e-06

Давайте проанализируем расположение точек восстановления относительно ближайших узлов:

```
Минимальное расстояние от контрольных точек до узлов интерполяции:

    x*_i    \delta_i
--------  ----------
0.771756  0.0265719
1.25      0.05
1.66158   0.00676139
```

Наименьшая погрешность наблюдается в точке x^{***}, для которой контрольная точка находится ближе остальных к ближайшему узлу интерполирования, что снижает интерполяционную ошибку.

Наибольшая погрешность соответствует точкам $x *, x^{**}$, Заметим, что погрешность интерполяции в этих точках – самая высокая. Можно сделать следующий вывод: чем дальше контрольная точка от узла, тем выше ошибка интерполяции.