**MINISTERUL EDUCAŢIEI**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Tehnologia Informației**

# Raport

**Disciplina:** Metode numerice

**Lucrarea de laborator nr. 3**

**Tema:** *“LUCRAREA DE LABORATOR NR. 3*

*INTERPOLAREA FUNCŢIILOR CU AJUTORUL POLINOMULUI LAGRANGE*”

Varianta: 23

**Student:** _____ **Raevschi Grigore TI-231**

**Coordonator:** _____ **Conf. univ. Pațiuc Vladimir**

Chișinău 2024

# Cuprins

**Scopul lucrării**

Pentru funcţia $f: [a, b] \rightarrow R$ se cunosc valorile $y0, y1, y2, \ldots, yn$ în nodurile distincte $x0, x1, x2, \ldots, xn$, adică $yi = f(xi), i = 0,1,2, \ldots, n$

1. Să se construiască polinomul de interpolare Lagrange $L\,n(x)$ ce aproximează funcţia dată.
2. Să se calculeze valoarea funcţiei $f(x)$ într-un punct $x = \alpha$ utilizând polinomul de interpolare Lagrange $L\,n(x)$.
3. Să se aproximeze valoarea funcţiei $f(x)$ pentru $x = \alpha$ cu eroarea $\varepsilon = 10^{-4}$ (sau cu cea mai bună exactitate posibilă), calculînd polinomul de interpolare Lagrange $L\,m(x)$, unde $m < n$
4. Să se compare şi să se explice rezultatele obţinute în 2) şi 3).

## 1 PROBLEMĂ PROPUSĂ SPRE REZOLVARE

23. $\alpha$ =1,276,

| $x$ | 0.765 | 1.867 | 3.987 | 5.601 | 7.043 | 9.231 | 10.987 |
|---|---|---|---|---|---|---|---|
| $y$ | 2.87611 | 4.18432 | 1.09673 | -1.4587 | -3.5729 | 0.9876 | 2.87644 |

## 2 REZOLVAREA MATEMATICĂ

**Calculul pentru $L_0(\alpha)$:**

$$L_0(\alpha) = \frac{(1.276 - 1.867)}{(0.765 - 1.867)} \cdot \frac{(1.276 - 3.987)}{(0.765 - 3.987)} \cdot \frac{(1.276 - 5.601)}{(0.765 - 5.601)} \cdot \frac{(1.276 - 7.043)}{(0.765 - 7.043)} \cdot \frac{(1.276 - 9.231)}{(0.765 - 9.231)} \cdot \frac{(1.276 - 10.987)}{(0.765 - 10.987)}$$

Calculăm fiecare fracţie separat:

1. $\frac{1.276-1.867}{0.765-1.867} = \frac{-0.591}{-1.102} = 0.536$
2. $\frac{1.276-3.987}{0.765-3.987} = \frac{-2.711}{-3.222} = 0.841$
3. $\frac{1.276-5.601}{0.765-5.601} = \frac{-4.325}{-4.836} = 0.894$
4. $\frac{1.276-7.043}{0.765-7.043} = \frac{-5.767}{-6.278} = 0.918$
5. $\frac{1.276-9.231}{0.765-9.231} = \frac{-7.955}{-8.466} = 0.94$
6. $\frac{1.276-10.987}{0.765-10.987} = \frac{-9.711}{-10.222} = 0.951$

Produsul acestor termeni:

$$L_0(\alpha) = 0.536 \cdot 0.841 \cdot 0.894 \cdot 0.918 \cdot 0.94 \cdot 0.951 \approx 0.314$$

Termenul corespunzător:

$$y_0 \cdot L_0(\alpha) = 2.87611 \cdot 0.314 \approx 0.902$$

# 3 Codul programului în limbajul C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double lagrange_interpolation(double *x, double *y, int n, double xi) {
    double result = 0.0;
    for (int i = 0; i < n; i++) {
        double term = y[i];
        for (int j = 0; j < n; j++) {
            if (j != i) {
                term *= (xi - x[j]) / (x[i] - x[j]);
            }
        }
        result += term;
    }
    return result;
}

int main() {
    int n;
    printf("Introduceti numarul de puncte: ");
    scanf("%d", &n);

    double *x = (double *)malloc(n * sizeof(double));
    double *y = (double *)malloc(n * sizeof(double));

    if (x == NULL || y == NULL) {
        printf("Eroare la alocarea memoriei!\n");
        return 1;
    }

    printf("Introduceti valorile lui x si y:\n");
    for (int i = 0; i < n; i++) {
        printf("x[%d]= ", i);
        scanf("%lf", &x[i]);
        printf("y[%d]= ", i);
        scanf("%lf", &y[i]);
    }

    double alpha;
    printf("Introduceti valoarea pentru x=alpha: ");
    scanf("%lf", &alpha);

    double result_interpolation = lagrange_interpolation(x, y, n, alpha);
    printf("Valoarea f(x) pentru x=%.3f este: %.6f\n", alpha, result_interpolation);

    // Aproximarea valorii funcţiei f(x) pentru x=alpha cu eroarea ε=10^(-4)
    double epsilon = 1e-4;
    int m = n;

    while (1) {
        double *xm = (double *)malloc(m * sizeof(double));
        double *ym = (double *)malloc(m * sizeof(double));

        if (xm == NULL || ym == NULL) {
            printf("Eroare la alocarea memoriei!\n");
```

```
            free(x);
            free(y);
            return 1;
        }

        for (int i = 0; i < m; i++) {
            xm[i] = x[i];
            ym[i] = y[i];
        }

        double result_high_precision = lagrange_interpolation(xm, ym, m, alpha);
        double error = fabs(result_high_precision - result_interpolation);

        free(xm);
        free(ym);

        if (error < epsilon) {
             printf("Aproximarea cu eroarea ε=10^(-4) pentru x=%.3f este: %.6f\n",
alpha, result_high_precision);
            break;
        }

        // Mărește m pentru o aproximare mai bună
        m++;
    }

    // Eliberarea memoriei
    free(x);
    free(y);

    return 0;
```

# 4 Rezultatul programului

```
Introduceti numarul de puncte: 7
Introduceti valorile lui x si y:
x[0]= 0.765
y[0]= 2.87611
x[1]= 1.867
y[1]= 4.18432
x[2]= 3.987
y[2]= 1.09673
x[3]= 5.601
y[3]= -1.4587
x[4]= 7.043
y[4]= -3.5729
x[5]= 9.231
y[6]= 2.87644
Introduceti valoarea pentru x=alpha: 1.276
Valoarea f(x) pentru x=1.276 este: 4.286062
Aproximarea cu eroarea ε=10^(-4) pentru x=1.276 este: 4.286062
```

## Concluzie

Prin această lucrare, s-a demonstrat eficiența și flexibilitatea metodei de interpolare Lagrange în aproximarea funcțiilor necunoscute pe baza unui set de puncte discrete. Metoda oferă o soluție precisă și adaptabilă pentru estimarea valorilor funcției și pentru realizarea de aproximități cu erori controlate. Rezultatele obținute arată că interpolarea cu un subset de puncte poate fi suficientă pentru a obține o precizie dorită, fiind totodată mai eficientă din punct de vedere al calculului.

În urma aplicării celor două metode (interpolarea cu toți cei n termeni și interpolarea cu mai puțini termeni pentru a atinge precizia cerută), s-a observat că utilizarea unui număr mai mic de puncte poate oferi un rezultat la fel de precis, în timp ce reduce complexitatea și riscul de erori numerice. Acest aspect este important în cazurile în care datele sunt numeroase sau când stabilitatea numerică este critică.