

Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра вычислительных методов и программирования

Отчет по лабораторной работе №7

Аппроксимация функций

Вариант 11

Выполнил:

студент 1 курса

группы № 348602

Трошкин Дмитрий Сергеевич

Проверил:

Матюшкин Светослав Иванович

Минск 2023

1 АППРОКСИМАЦИЯ ФУНКЦИЙ

Цель работы: изучить алгоритмы аппроксимации функций; освоить методику построения и использования алгебраических интерполяционных многочленов Лагранжа и Ньютона

1.1 Условие

Написать и отладить программу (задания из табл. 7.1) аппроксимации функции $f(x)$ на интервале $[a, b]$ заданным методом, m – количество точек, в которых известна функция (размер таблицы). Вид функции задан для получения значений таблицы (x_i, y_i) , $i=1, 2, \dots, m$ и проверки качества аппроксимации. Решение задачи оформить отдельной функцией, алгоритм которой описать в виде блок-схемы.

1.2 Исходный код

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

typedef double (*f_type)(double x);
typedef double (*f_inter_type)(double* x_arr, double* y_arr, double x, int m);

double calc_derivative(f_type f, double x, double dx)
{
    return (f(x + dx) - f(x)) / dx;
}

double func_max(f_type f, double a, double b, double eps)
{
    double max = 0, y;
    for(double x = a - eps; x < b + eps; x += eps)
    {
        y = f(x);
        max = max < y ? y : max;
    }
    return max;
}

double func_min(f_type f, double a, double b, double eps)
{
    double min = 0, y;
    for(double x = a - eps; x < b + eps; x += eps)
    {
        y = f(x);
        min = min > y ? y : min;
    }
}
```

```

    }
    return min;
}

double *gen_val_table(f_type f, double a, double b, double *max, double *min, int size)
{
    double h = (b - a) / (size+1) ;
    double *arr = malloc(size * sizeof(double));

    for(int i = 0; i < size; i++, a += h)
    {
        arr[i] = f(a);
        if(isinf(arr[i])) arr[i] = f(a-h);
        *min = (*min > arr[i]) ? arr[i] : *min;
        *max = (*max < arr[i]) ? arr[i] : *max;
    }
    return arr;
}

double *gen_inter_val_table(f_inter_type f, double a, double b, double *max,
                           double *min, double *x_arr, double *y_arr, int m, int size)
{
    double h = (b - a) / (size+1) ;
    double *arr = malloc(size * sizeof(double));

    for(int i = 0; i < size; i++, a += h)
    {
        arr[i] = f(x_arr, y_arr, a, m);
        if(isinf(arr[i])) arr[i] = f(x_arr, y_arr, a-h, m);
        *min = (*min > arr[i]) ? arr[i] : *min;
        *max = (*max < arr[i]) ? arr[i] : *max;
    }
    return arr;
}

void to_inter(f_type f, int m, double a, double b, double **x_arr, double **y_arr)
{
    double step = (b - a) / (m - 1);
    *x_arr = malloc(m*sizeof(double));
    *y_arr = malloc(m*sizeof(double));
    for(int i = 0; i < m; i++, a+= step)
    {
        (*x_arr)[i] = a;
        (*y_arr)[i] = f(a);
    }
}

double func_lin_inter(double* x_arr, double* y_arr, double x, int m)
{
    if(m < 3) return 0;
    int t = ceil((m-1) * (x - x_arr[0]) / (x_arr[m-1] - x_arr[0]));
    if(t <= 0) t = 1;

```

```

    if(t >= m) t = m - 1;
    return y_arr[t - 1] + (x - x_arr[t - 1])*(y_arr[t] - y_arr[t - 1])/(x_arr[t] - x_arr[t - 1]);
}

double func_kvad_inter(double* x_arr, double* y_arr, double x, int m)
{
    if(m < 3) return 0;
    int t = ceil((m-1) * (x - x_arr[0]) / (x_arr[m-1] - x_arr[0]));
    if(t < 1) t = 1;
    if(t >= m - 1) t = m - 2;
    return func_lin_inter(x_arr, y_arr, x, m-1) + (x - x_arr[t - 1])*(x - x_arr[t]) *
        ((y_arr[t - 1] - y_arr[t])/(x_arr[t - 1] - x_arr[t]) -
        (y_arr[t] - y_arr[t + 1])/(x_arr[t] - x_arr[t + 1])) /
        (x_arr[t-1] - x_arr[t+1]);
}

void draw_table(char *plot, char symbol, double *y, int size,
                int height, double min, double max)
{
    double stepY = (max - min) / height;

    for(int i = 1; i < size - 2; i++) {
        if(y[i] >= min && y[i] <= max)
            plot[i + (int)((max - y[i]) / stepY) * size] = symbol;
    }
}

void draw_axis(char *plot, char x, char y, double a, double b,
               int size, double min, double max, int height)
{
    double stepX = (b - a) / size;
    double stepY = (max - min) / height;

    for(int i = 1; i < size - 3; i++)
        plot[i + (int)trunc(max / stepY) * size] = x;
    plot[size - 3 + (int)trunc(max / stepY) * size] = '>';
    for(int i = 1; i < height - 1; i++)
        plot[size - (int)trunc(b / stepX) + i * size] = y;
    plot[size - (int)trunc(b / stepX) + size] = '^';
}

void draw_box(char *plot, int size, int height)
{
    for(int i = 0; i < height; i++)
        plot[i * size] = plot[size - 2 + i*size] = '|';

    for(int i = 0; i < size - 1; i++)
        plot[i] = plot[i + size*(height-1)] = '-';
}

```

```

    for(int i = 0; i < height - 1; i++)
        plot[size - 1 + i*size] = '\n';

    plot[size + (height-1)*size] = 0;
}

double fun(double x) {return pow(M_E, x);}
void plot_term(f_type f, double a, double b, int width, int height)
{
    int m = 6;
    double *y_arr, *x_arr;
    double yMax = f(a);
    double yMin = f(a);
    char *res = malloc((width+1) * height * sizeof(char));
    memset(res, ' ', (width+1)* height - 1);
    double *y1 = gen_val_table(f, a, b, &yMax, &yMin, width - 1);
    to_inter(f, m, a, b, &x_arr, &y_arr);
    double *y2 = gen_inter_val_table(func_kvad_inter, a, b,
                                     &yMax, &yMin, x_arr, y_arr, m, width - 1);

    yMin -= 0.05 *(yMax - yMin);
    yMax += 0.05 *(yMax - yMin);

    draw_table(res, '1', y1, width, height, yMin, yMax);
    draw_table(res, '2', y2, width, height, yMin, yMax);
    y2 = gen_inter_val_table(func_lin_inter, a, b, &yMax, &yMin, x_arr, y_arr, m, width - 1);
    draw_table(res, '3', y2, width, height, yMin, yMax);
    draw_axis(res, 'x', 'y', a, b, width, yMin, yMax, height);
    draw_box(res, width, height);
    puts(res);
    printf("Axis: xMin = %lf, xMax = %lf, yMin = %lf, yMax = %lf\n", a, b, yMin, yMax);
    free(y1);
    free(y2);
}

double metod_n(f_type f, double x, double eps)
{
    return x - f(x) / calc_derivative(f, x, eps);
}

void calc(f_type f, double a, double b, double h, double eps)
{
    double xn = a - 1, y;
    int roots_count = 0;
    printf("--- Roots of function ---\n");
    for(double x = a - h; x < b + h; x += h)
    {
        y = f(x);
        if(y*f(x+h) < 0) {
            roots_count++;
            printf(" %i. %lf\n", roots_count, metod_n(f, x, eps));
        }
    }
}

```

```

    }
    else if(y < eps && xn < a) xn = x;
    else if(y > eps && xn > a) {
        roots_count++;
        printf(" %i. [%lf, %lf]\n", roots_count, xn, x);
        xn = a - 1;
    }

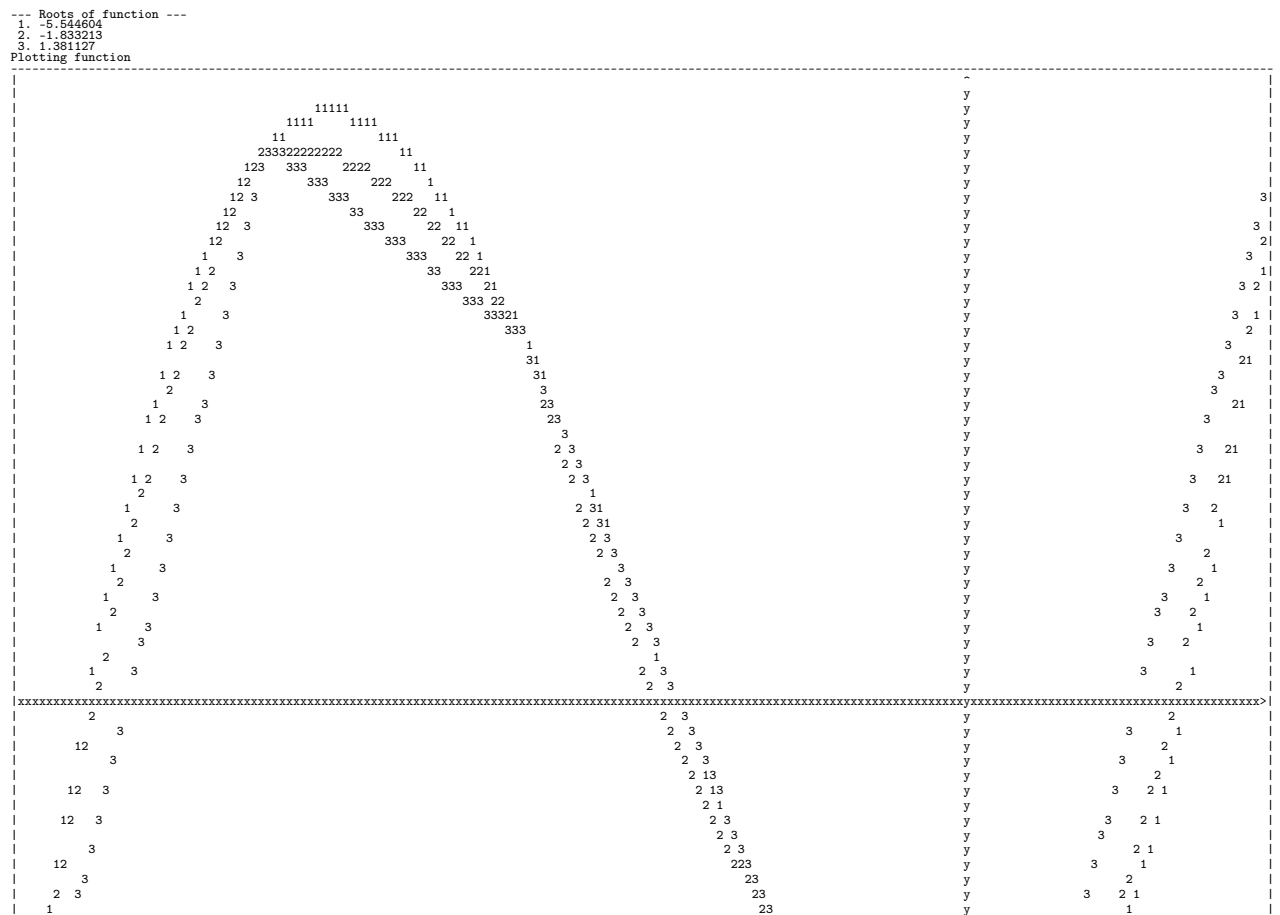
} if(!roots_count) printf("No roots.\n");
}

double function(double x) { return (x*x*x + 6*x*x - 0.02*pow(M_E, x) - 14); }

int main()
{
    double a = -6, b = 2, h = 1E-6, eps = 1E-5;
    calc(function, a, b, h, eps);
    printf("Plotting function\n");
    plot_term(function, a, b, 180, 80);
}

```

1.3 Пример



Axis: xMin = -6.000000, xMax = 2.000000, yMin = -16.776794, yMax = 19.738455