Санкт-Петербургский государственный университет

Прикладная математика и информатика

Отчет по лабораторной работе №2 по вычислительному практикуму

**Приближенное решение нелинейных уравнений**

Выполнил:

Чернов Павел Олегович

группа 221

Санкт-Петербург

2022

1. **Введение**

Задача алгебраического интерполирования. Интерполяционный многочлен в форме Ньютона и в форме Лагранжа.

Подготовительный этап:

Составить и вывести на печать таблицу из (m+1) значения функции f в попарно-различных точках (узлах)zj, где j=0,1,..,m. Здесь число значений в таблице —параметр задачи, формула для непрерывной функции f представлена в варианте задания.

При создании таблицы возможно как случайное задание узлов из некоторого промежутка [a; b] (важным ограничением здесь является попарная различность узлов), так и задание с помощью формулы (например, равноотстоящие с шагом h=(b-a)/m узлы zj = a+j·h, j=0,1, ..., m.

Решение задачи алгебраического интерполирования:

Дана таблично-заданная функция (m+1) аргумента. Требуется найти значение в точке x, (здесь x—параметр задачи; пользователю предлагается ввести произвольное значение x).

Для этого требуется построить интерполяционный алгебраический многочлен, степени не выше n, (n‒ параметр задачи; пользователю предлагается ввести произвольное значение n ≤ m).

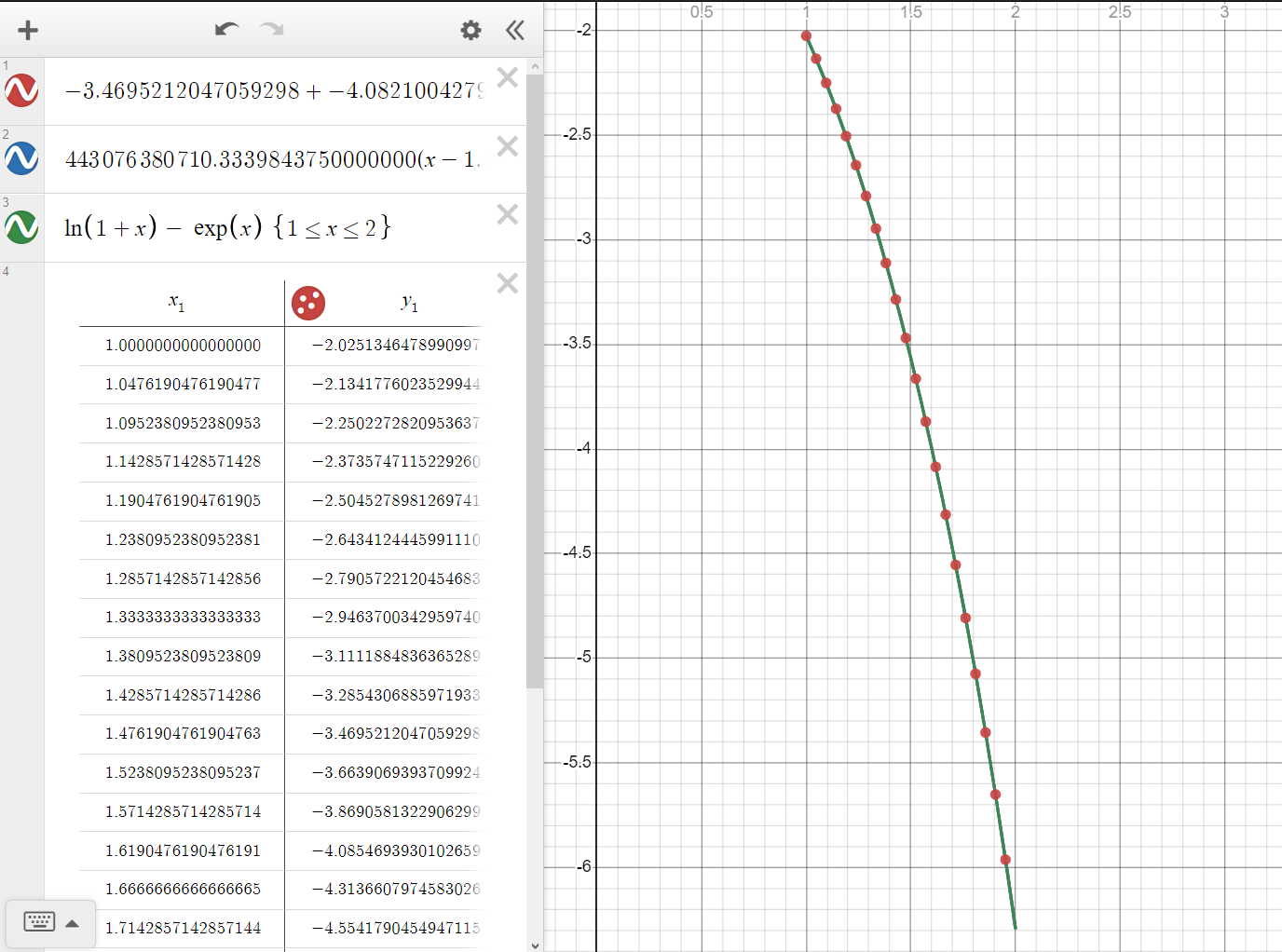
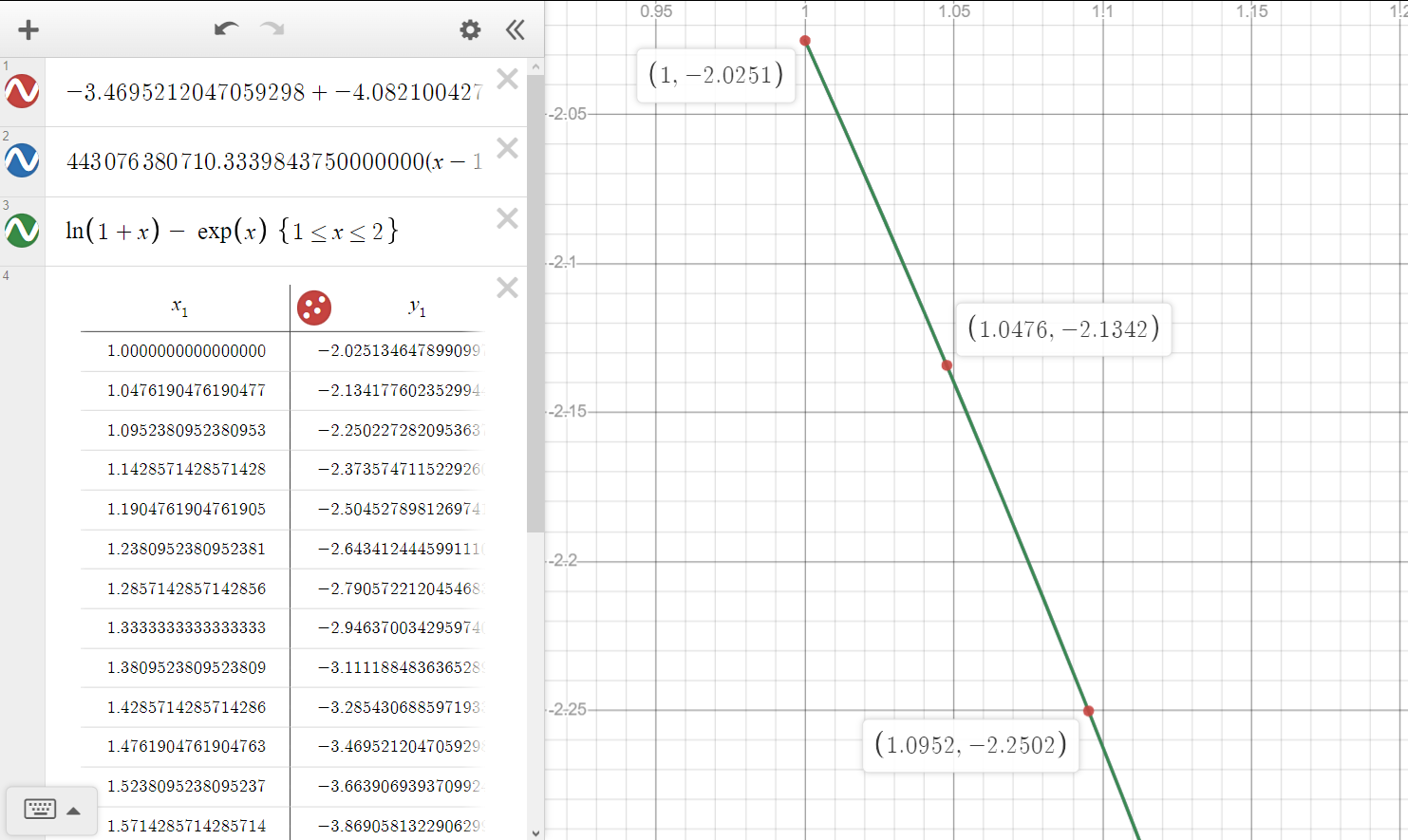
Решением задачи будет значение Pn(x)≈f(x) (здесь Pn—алгебраический интерполяционный многочлен функции f, степени не выше n (при этом n ≤ m), построенный по набору из (n+1) узла zj, решающему задачу минимизации погрешности интерполирования в заданной точке x.

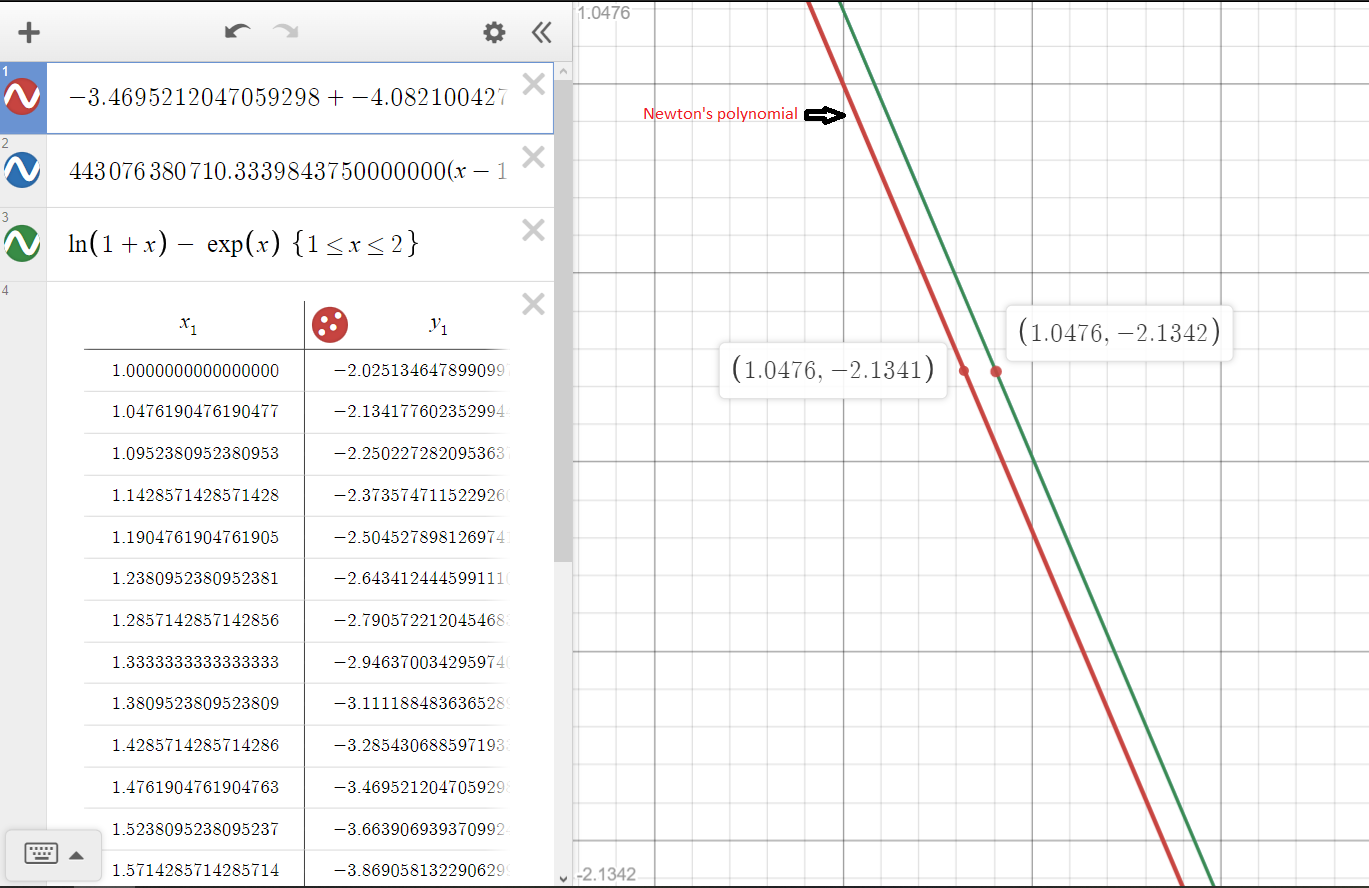
Для выбора «оптимальных» для точки x узлов необходимо упорядочить узлы исходной таблицы по мере удаления их от точки интерполирования x (провести любую любимую сортировку). Далее работать уже с отсортированной таблицей. Узлы для построения Pn будут располагаться в первых (n+1) строках отсортированной таблицы (в наших обозначениях это x0, x1, ... xn).

Найти значение PnL(x), используя представление в форме Лагранжа. Вычислить фактическую погрешность efn(x) = |f(x)-PnL(x)|.

Найти значение PnN(x), используя представление в форме Ньютона. Для этого построить таблицу разделенных разностей по первым (n+1) значениям таблицы до порядка n включительно. Вычислить фактическую погрешность efn(x) = |f(x)-PnN(x)|.

1. **Ход работы**
2. Представленная ниже программа позволяет аппроксимировать функцию на заданном отрезке, а также вычислить ее приблизительное значение на всей области определения.
3. Для аппроксимации функции используется нахождение интерполяционного многочлена как в форме Лагранжа, так и в форме Ньютона.
4. Исходным параметром задачи является функция . В программе можно задать а) число табличных значений, по которым будет проводиться приближение, б) концы отрезка [a; b] из которого выбираются узлы аппроксимации, в) x –точка интерполирование, значение в которой хотим найти, г) n – степень интерполяционного многочлена, который будет построен для того, чтобы найти значение x. Ввод осуществляется при помощи клавиатуры.
5. Программа написана на языке С++, в расчетах используются переменные типа double.
6. **Приложение**

**а) Графики**



**б) Программа**

#include <windows.h>

#include <vector>

#include <iomanip>

#include "Header.h"

double f(double x) { //Интерполируемая функция

return log(1 + x) - exp(x); //вариант 13

}

void sort(std::vector<std::pair<double, double>> &table, double x) { //Сортировка таблица значений относительно расстояния до точки интерполяции

for (int i = 0; i < table.size(); ++i) {

for (int j = i + 1; j < table.size(); ++j) {

if (abs(table[j].first - x) < abs(table[i].first - x)) {

std::pair<double, double> tmp = table[j];

table[j] = table[i];

table[i] = tmp;

}

}

}

}

double Lagrange\_poly(unsigned int n, double x, std::vector <std::pair<double, double>> table) {

double res = 0;

for (int k = 0; k <= n; ++k) {

double mult = 1;

for (int i = 0; i <= n; ++i) {

if (i != k) {

mult \*= table[k].first - table[i].first;

}

}

double w = 1; for (int j = 0; j <= n; ++j) {

if (j != k) {

w \*= x - table[j].first;

}

} res += w \* table[k].second / mult;

}

return res;

}

double Newton\_poly\_correct(unsigned int n, double x, std::vector <std::pair<double, double>> table) {

double res = table[0].second, mult = 1;

std::vector <std::vector<double>> A(n+1, std::vector<double>(n+1));

for (int i = 0; i < n; ++i) {

A[i][0] = table[i].second;

}

for (int j = 1; j < n; ++j) {

for (int i = 0; i < n - j; ++i) {

A[i][j] = (A[i + 1][j - 1] - A[i][j - 1]) / (table[j+i].first - table[i].first);

}

}

for (int i = 0; i < n; ++i) {

mult \*= (x - table[i].first);

res += A[0][i+1] \* mult; }

return res;

}

int main() {

SetConsoleTitleA("Algebraic interpolation problem");

std::cout << std::fixed << std::setprecision(16);

unsigned int m; //Кол-во точек в таблице (m = 21)

std::cout << "Input number of table values: ";

std::cin >> m;

double a, b; //Концы отрезка интерполирования (a = 1, b = 2)

std::cout << "Input segment boundaries: ";

std::cin >> a >> b;

std::cout << "Table of values: \n" << std::setw(15) << "X\_i" << std::setw(17) << "f(X\_i)\n";

std::vector <std::pair<double, double>> table\_values(m); //Если задавать рандомно, надо будет компаратор написать, чтобы по иксам сортить

for (int i = 0; i < m; ++i) {

table\_values[i].first = a + (b - a) \* i / m;

table\_values[i].second = f(table\_values[i].first);

std::cout << table\_values[i].first << "|" << table\_values[i].second << "\n";

}

int flag;

do {

double x; //Точка интерполирования

std::cout << "Input interpolation point: ";

std::cin >> x;

while (x <= -1) {

std::cout << "Have you ever seen the initial function? You are inputting values out of its range, stupid. Fine, I'm giving you one another try: ";

std::cin >> x;

}

unsigned int n; //Степень интерполяционного М.Ч. (n = 14)

std::cout << "Input degree of interpolation polinom (Warning, degree must be less than number of table values, i.e. n < m): ";

std::cin >> n;

while (n >= m) {

std::cout << "Incorrect input, consider condition n < m, try again: ";

std::cin >> n;

}

if (n < m - 1) {

sort(table\_values, x);

std::cout << "Sorted table of values: \n" << std::setw(15) << "X\_i" << std::setw(17) << "f(X\_i)\n";

for (int i = 0; i < m; ++i) {

std::cout << table\_values[i].first << "|" << table\_values[i].second << "\n";

}

}

double Lagr\_res = Lagrange\_poly(n, x, table\_values), Newt\_res = Newton\_poly\_correct(n, x, table\_values), f\_res = f(x);

std::cout << "Value of the Lagrange polynomial at the interpol. point: " << Lagr\_res << "\n" <<

"Value of absolute error rate: " << abs(Lagr\_res - f\_res) << "\n";

std::cout << "Value of the Newton polynomial at the interpol. point: " << Newt\_res << "\n" <<

"Value of absolute error rate: " << abs(Newt\_res - f\_res) << "\n";

std::cout << "If you want to try another interpolation point, enter 1, otherwise 0: ";

std::cin >> flag;

} while (flag);

return 0;

}

https://github.com/Chernovuk/Chislaki/tree/master/Second\_task