Санкт-Петербургский Государственный Университет
Прикладная математика и информатика

Отчёт по вычислительному практикуму 2
Задача алгебраического интерполирования Интерполяционный многочлен в форме Лагранжа и в форме Ньютона

Выполнил:
Яковлев Денис Михайлович
Группа 21.Б06-мм
st095998@student.spbu.ru

Под руководством Алцыбеева Глеба Олеговича
Преподавателя и ассистента по дисциплине "Вычислительный практикум"

17 Марта 2023 г.

Содержание

1 Введение

Численные методы - это основной инструмент для решения задач, не разрешимых аналитическими методами (когда решение задачи выражается при помощи формул).
Стоит отметить, что численные методы являются аппроксимационными, то есть, позволяют получить лишь приближённое решение задачи. Для того, чтобы узнать отличие от точного значения (если это возможно), прибегают к использованию абсолютной и относительной погрешностей.
Поставленная задача алгебраического интерполирования ставит собою цель обучить студента интерполировать произвольно заданную алгебраическую или трансцендентную функцию вида

$$f(x) = 0 \tag{1}$$

Пользуясь построением интерполяционного многочлена в форме Лагранжа и в форме Ньютона.

2 Постановка задачи

2.1 Подготовительный этап

Составить и вывести на печать таблицу из $(m+1)$ значения функции f в попарно-различных точках (узлах) z_j, 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 $h = \frac{b-a}{m}$, узлы $z_j = a + jh, j = 0, 1, ..., m$).

2.2 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 $n \leq m$).
Решением задачи будет значение $P_n(x) \approx f(x)$ (здесь P_n — алгебраический интерполяционный многочлен функции f, степени не выше n (при этом $n \leq m$), построенный по набору из $(n+1)$ узла z_j , 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 P_n будут располагаться в первых $(n+1)$ строках отсортированной таблицы (в наших обозначениях это $x_0, x_1, ..., x_n$.

Найти значение $P_n^L(x)$, используя представление в форме Лагранжа. Вычислить фактическую погрешность $ef_n(x) = |f(x) - P_n^L(x)|$.
Найти значение $P_n^N(x)$, используя представление в форме Ньютона. Для этого построить таблицу разделенных разностей по первым $(n+1)$ значениям таблицы до порядка n включительно. Вычислить фактическую погрешность $ef_n(x) = |f(x) - P_n^N(x)|$.

3 Ход работы

3.1 Основные положения

3.1.1 Параметры задачи

ПАРАМЕТРЫ ЗАДАЧИ (запрашивать у пользователя; вводятся с клавиатуры):

1) число значений в таблице (в наших обозначениях это $m+1$);

2) концы отрезка $[a, b]$, из которого выбираются узлы интерполяции;

3) x - точка интерполирования, значение в которой хотим найти;

4) n - степень интерполяционного многочлена, который будет построен для того, чтобы найти значение в точке x.

3.1.2 Важно!

ВАЖНО: При задании аргументов исходной таблицы выбирать/определять их попарно-различными!
Никаких ограничений на x нет, введенное x может совпадать с табличным или лежать вне $[a, b]$, из которого выбираются узлы интерполяции.
Запрашивая у пользователя значение n, сразу ограничивать его значением m, то есть просить ввести $n \leq m$. Если введенное пользователем $n > m$, сообщать: «Введено недопустимое значение n» и просить ввести n заново.

3.1.3 Выходные данные задачи

НА ЭКРАНЕ должна быть отражена следующая информация:

1) название задачи (Задача алгебраического интерполирования);

2) номер Вашего варианта;

3) запрос на число значений в таблице, границы интервала;

4) исходная таблица значений функции (результат подготовительного этапа);

5) запрос на точку интерполирования x; $x = ...$

6) запрос на степень интерполяционного многочлена $n \leq m$; $n = ...$

7) отсортированная таблица (или набор узлов, ближайших к точке x, по которым будет строиться интерполяционный многочлен);

8) значение интерполяционного многочлена $P_n^L(x)$, найденное при помо-щи представления в форме Лагранжа;

9) значение абсолютной фактической погрешности для формы Лагранжа $|f(x) - P_n^L(x)|$;

10) значение $P_n^N(x)$, найденное при помощи представления в форме Ньютона;

11) значение абсолютной фактической погрешности для формы Ньютона $|f(x) - P_n^N(x)|$;

12) предложение ввести новые значения x и n или выйти из программы.

3.2 Пример: Входные данные

В качестве входных параметров:

- функция $f(x) = e^{-x} - \frac{x^2}{2}$;
- $a = 0$ - начало отрезка;
- $b = 5$ - конец отрезка;
- $(m + 1) = 26$ - число аргументов;
- $n = 10$ - степень интерполяционного многочлена;
- $x = 2;\ x = 8$ - две точки интерполяции;

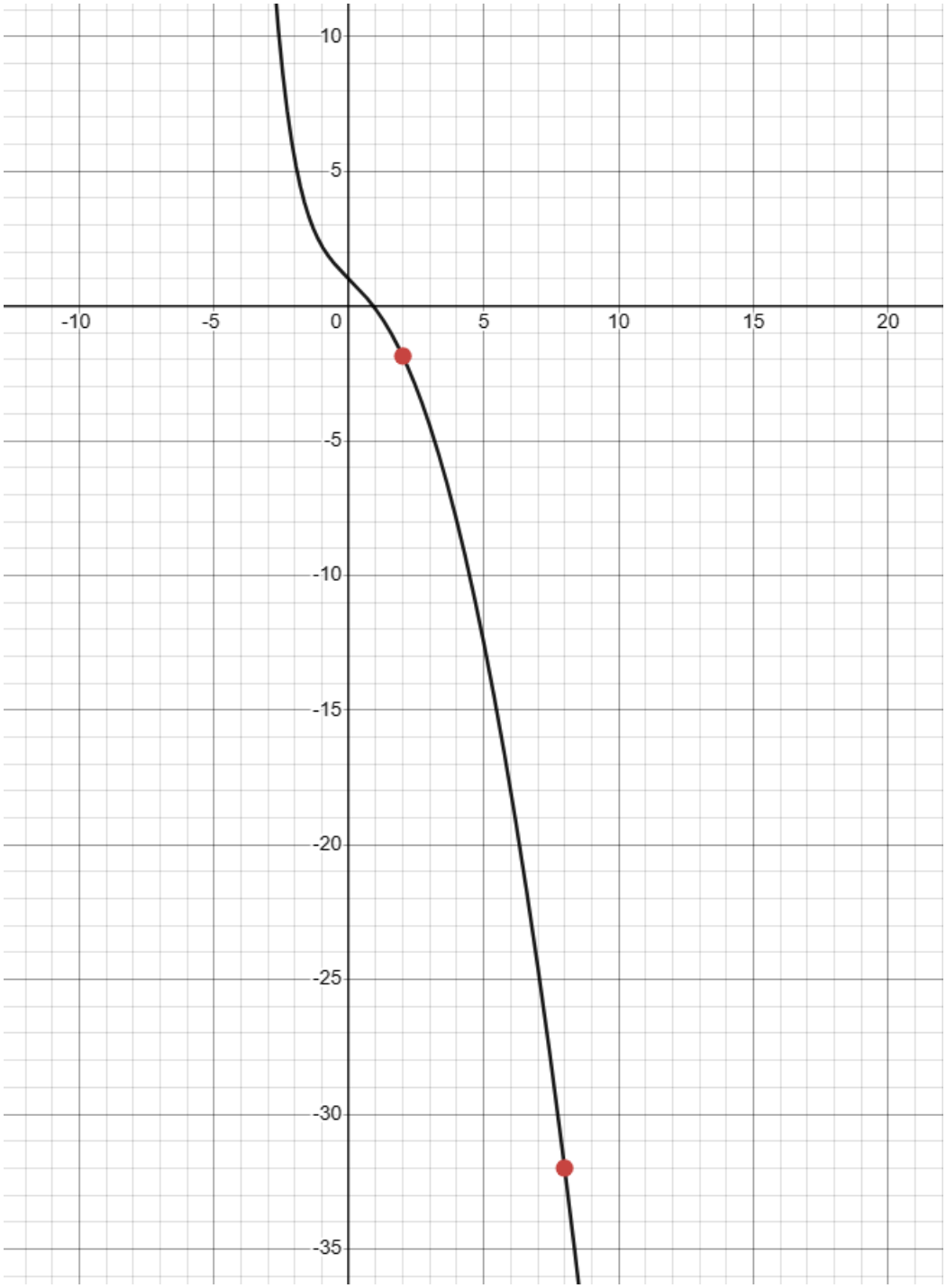

Рис. 1: График функции $f(x) = e^{-x} - \frac{x^2}{2}$. Значения в точках интерполирования выделены красным цветом

Ссылка на репозиторий: `https://github.com/DeMiYak/NumericalMethods`

3.3 Вывод

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

Вариант 7. Функция f(x) = eps(-x) - x^2/2

Начало отрезка: 0
Конец отрезка: 5
Число аргументов: 26
x  f(x)
0  1
0.2  0.798731
0.4  0.59032
0.6  0.368812
0.8  0.129329
1  -0.132121
1.2  -0.418806
1.4  -0.733403
1.6  -1.0781
1.8  -1.4547
2  -1.86466
2.2  -2.3092
2.4  -2.78928
2.6  -3.30573
2.8  -3.85919
3  -4.45021
3.2  -5.07924
3.4  -5.74663
3.6  -6.45268
3.8  -7.19763
4  -7.98168
4.2  -8.805
4.4  -9.66772
4.6  -10.5699
4.8  -11.5118
5  -12.4933

Введите точку интерполяции x: 2

Степень интерполяционного многочлена (меньше числа аргументов): 10

#   x   f(x)
2.22045e-16   2   -1.86466
0.2   2.2   -2.3092
```

```
0.2   1.8   -1.4547
0.4   2.4   -2.78928
0.4   1.6   -1.0781
0.6   1.4   -0.733403
0.6   2.6   -3.30573
0.8   1.2   -0.418806
0.8   2.8   -3.85919
1   1   -0.132121
1   3   -4.45021
1.8   0.2   0.798731
1.4   0.6   0.368812
1.6   0.4   0.59032
1.2   0.8   0.129329
2   0   1
1.2   3.2   -5.07924
1.4   3.4   -5.74663
1.6   3.6   -6.45268
1.8   3.8   -7.19763
2   4   -7.98168
2.2   4.2   -8.805
2.4   4.4   -9.66772
2.6   4.6   -10.5699
2.8   4.8   -11.5118
3   5   -12.4933
Нажмите 1, чтобы начать процесс интерполяции
Нажмите 0, чтобы выйти

1

Интерполяция Лагранжа:
Значение в точке интерполяции x = 2: -1.864664716763387
Абсолютная погрешность: 2.22045e-16

Интерполяция Ньютона:
Значение в точке интерполяции x = 2: -1.864664716763387
Абсолютная погрешность: 0


Нажмите 1, чтобы начать процесс интерполяции
Нажмите 0, чтобы выйти

1
Введите точку интерполяции x: 8

Степень интерполяционного многочлена (меньше числа аргументов): 10
```

```
#   x   f(x)
3   5   -12.4933
3.2   4.8   -11.5118
3.4   4.6   -10.5699
3.6   4.4   -9.66772
3.8   4.2   -8.805
4   4   -7.98168
4.2   3.8   -7.19763
4.4   3.6   -6.45268
4.6   3.4   -5.74663
4.8   3.2   -5.07924
5   3   -4.45021
7.8   0.2   0.798731
7.4   0.6   0.368812
7.6   0.4   0.59032
7.2   0.8   0.129329
8   0   1
7   1   -0.132121
5.2   2.8   -3.85919
6.8   1.2   -0.418806
5.4   2.6   -3.30573
6.6   1.4   -0.733403
6.4   1.6   -1.0781
5.6   2.4   -2.78928
6.2   1.8   -1.4547
5.8   2.2   -2.3092
6   2   -1.86466

Интерполяция Лагранжа:
Значение в точке интерполяции x = 8: -31.99839789606631
Абсолютная погрешность: 0.00126664

Интерполяция Ньютона:
Значение в точке интерполяции x = 8: -31.99839789606649
Абсолютная погрешность: 0.00126631


Нажмите 1, чтобы начать процесс интерполяции
Нажмите 0, чтобы выйти

0
```

4 Код программы

4.1 Интерполяция по Лагранжу

```
1  void LagrangeInterpolation(double *arg, size_t size, size_t deg, double x, double
       (*fnc)(double*, size_t, double, double), double (*f)(double))
2  {
3      cout << endl << "Интерполяция Лагранжа:" << endl;
4      double denominator;
5      double numerator;
6      double total = 0;
7      double *ptr = arg;
8      for(;ptr < (arg + deg); ptr++)
9      {
10         denominator = fnc(arg, deg, *ptr, *ptr);
11         numerator = fnc(arg, deg, x, *ptr);
12         // cout « "denominator: «< denominator « endl;
13         total += numerator/denominator*(*(ptr+size));
14         // cout « "total: «< total « endl;
15     }
16     cout << "Значение в точке интерполяции x = " << x << ": " << total << endl;
17     cout << "Абсолютная погрешность: " << fabs(total - f(x)) << endl << endl;
18 }
```

4.2 Интерполяция по Ньютону

```
void NewtonInterpolation(double *arg, size_t size, size_t deg, double x, double
    (*fnc)(double*, size_t, double, double), double (*f)(double))
{
    cout << "Интерполяция Ньютона:" << endl;
    size_t degp = deg + 1;
    double space[degp][deg], total = 0, product = 1;
    for(size_t i = 0; i < deg; i++)
    {
        space[0][i] = arg[i];
        space[1][i] = arg[size+i];
    }
    for(size_t i = 2; i < degp; i++)
    {
        for(size_t j = 0; j < degp - i; j++)
        {
            space[i][j] = (space[i-1][j+1]-space[i-1][j])/(space[0][j+i-1]-space[0][j]);
        }
    }


    for(size_t i = 0; i < 2; i++)
    {
        cout << "f" << i << ": ";
        for(size_t j = 0; j < degp; j++)
            cout << space[i][j] << ' ';
        cout << endl;
    }

    for(size_t i = 2; i < degp; i++)
    {
        cout << "f" << i << ": ";
        for(size_t j = 0; j < degp - i; j++)
        cout << space[i][j] << ' ';
        cout << endl;
    }

    for(size_t i = 1; i < degp; i++)
    {
        total += space[i][0]*product;
        product *= (x - arg[i-1]);
    }

    cout << "Значение в точке интерполяции x = " << x << ": " << total << endl;
    cout << "Абсолютная погрешность: " << fabs(total - f(x)) << endl << endl;
}
```

4.3 Дополнительные функции

```
1  double f(double x)
2  {
3      return exp(-x) - x*x/2;
4  };
5
6  double productValue(double *arg, size_t size, double x, double c)
7  {
8      double value = 1; double* ptr = arg;
9      for(;ptr < (arg + size); ptr++) {if(c!=*ptr) value *= (x-*ptr);}
10     // cout « "value: «< value « endl;
11     return value;
12 }
13
14 double* buildup(double(*f)(double), size_t size, double a, double b)
15 {
16     double *temp = new double[2*size];
17     double *ptr = temp;
18     double h;
19     if(size==1) h = (a+b)/2;
20     else h = (b-a)/(size-1);
21     double x = a;
22     cout << "x   " << "f(x)" << endl;
23     for(;ptr < (temp + size); ptr++, x+=h)
24     {
25         *ptr = x;
26         *(ptr + size) = f(x);
27         cout << *ptr << "   " << *(ptr+size) << endl;
28     }
29     return temp;
30 };
31
32 void sortTable(double *arg, size_t size, size_t deg, double x, void (*fnc1)(double*,
       double*, size_t, size_t), void (*fnc2)(double*, double*, size_t))
33 {
34
35     double temp[size];
36     for(size_t i = 0; i < size; i++) temp[i]=fabs(x-arg[i]);
37     fnc1(temp, arg, size, deg);
38     fnc2(temp, arg, size);
39 };
40
41 void sortArray(double *temp, double *arg, size_t size, size_t deg)
42 {
43     for(size_t i = 0; i < deg; i++)
44     {
```

```
45          size_t min = i;
46          for(size_t j = i + 1; j < size; j++)
47          {
48              if(temp[j] < temp[min]) min = j;
49          }
50          if(min!=i)
51          {
52              double temp1 = temp[min], temp2 = arg[min], temp3 = arg[min+size];
53              temp[min] = temp[i];
54              temp[i] = temp1;
55              arg[min] = arg[i];
56              arg[i] = temp2;
57              arg[min+size] = arg[i+size];
58              arg[i+size] = temp3;
59          }
60      }
61  };
62
63  void printArray(double *tmp, double *temp, size_t size)
64  {
65      double *ptr = temp;
66      double *val = tmp;
67      cout << "#    " << "x    " << "f(x)" << endl;
68      for(;ptr < (temp + size); ptr++, val++)
69      {
70          cout << *val << "    " << *ptr << "    " << *(ptr + size) << endl;
71
72      }
73  }
```