Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3
Задание №1 по курсу «Численные методы»
Вариант №12

Студент: С. М. Власова
Преподаватель: И. Э. Иванов
Группа: М8О-306Б
Дата:
Оценка:
Подпись:

Москва, 2020

Задание №3.1

Используя таблицу значений Y_i функции $y = f(x)$, вычисленных в точках X_i, $i = 0, \ldots, 3$, построить интерполяционные многочлены Лагранжа и Ньютона, проходящие через точки $[X_i, Y_i]$. Вычислить значение погрешности интерполяции в точке $X^\star$.

Вариант: 12

$$y = \sin(x) + x, \quad a) X_i = 0, \frac{\pi}{6}, \frac{2\pi}{6}, \frac{3\pi}{6}; \quad b) X_i = 0, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{2}; \quad X^\star = 1.0.$$

1 Описание метода решения

ИНТЕРПОЛЯЦИЯ

Пусть на отрезке $[a, b]$ задано множество не совпадающих точек x_i (интерполяционных узлов), в которых известны значения функции $f_i = f(x_i), \quad i = 0, \ldots, n$. Приближающая функция $\varphi(x, a)$ такая, что выполняются равенства

$$\varphi(x_i, a_0, \ldots, a_n) = f(x_i) = f_i, \quad i = 0, \ldots, n.$$

называется интерполяционной.

Наиболее часто в качестве приближающей функции используют многочлены степени n :

$$P_n(x) = \sum_{i=0}^{n} a_i \cdot x^i$$

Подставляя в многочлен значения узлов интерполяции и используя условие $P_n(x_i) = f_i$, получаем систему линейных алгебраических уравнений относительно коэффициентов a_i :

$$\sum_{i=0}^{n} a_i \cdot x^i = f_k, \quad k = 0, \ldots, n,$$

которая в случае несовпадения узлов интерполяции имеет единственное решение.

Для нахождения интерполяционного многочлена не обязательно решать вышеуказанную систему. Произвольный многочлен может быть записан в виде:

$$L_n(x) = \sum_{i=0}^{n} f_i \cdot l_i(x).$$

Здесь $l_i(x)$ – многочлены степени n, так называемые лагранжевые многочлены влияния, которые удовлетворяют условию

$$l_i(x_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

и, соответственно, $l_i(x) = \prod\limits_{j=0, j\neq i}^{n} \frac{(x-x_i)}{(x_i-x_j)}$, а интерполяционный многочлен запишется в виде

$$L_n(x) = \sum_{i=0}^{n} f_i \prod_{j=0, j\neq i}^{n} \frac{(x - x_i)}{(x_i - x_j)}.$$

Интерполяционный многочлен, записанный в такой форме, называется **интерполяционным многочленом Лагранжа.**

Если ввести функцию $\omega_{n+1}(x) = (x - x_0)(x - x_1)\dots(x - x_n) = \prod_{i=0}^{n}(x - x_i)$, то выражение для интерполяционного многочлена Лагранжа примет вид:

$$L_n(x) = \sum_{i=0}^{n} f_i \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)}.$$

Недостатком интерполяционного многочлена Лагранжа является необходимость полного пересчета всех интерполяционных коэффициентов в случае добавления дополнительных узлов. Чтобы избежать указанного недостатка, используют интерполяционный многочлен в форме Ньютона.

Введем понятие разделенной разности. Разделенные разности нулевого порядка совпадают со значениями функции в узлах. Разделенные разности первого порядка обозначаются $f(x_i, x_j)$ и определяются через разделенные разности нулевого порядка:

$$f(x_i, x_j) = \frac{f_i - f_j}{x_i - x_j},$$

разделенные разности второго порядка определяются через разделенные разности первого порядка:

$$f(x_i, x_j, x_k) = \frac{f(x_i, x_j) - f(x_j, x_k)}{x_i - x_k}.$$

Разделенная разность порядка $n - k + 2$ определяется соотношениями

$$f(x_i, x_j, x_k, \dots, x_{n-1}, x_n) = \frac{f(x_i, x_j, x_k, \dots, x_{n-1}) - f(x_j, x_k, \dots, x_n)}{x_i - x_n}.$$

Таким образом, для $(n + 1)-$й точки могут быть построены разделенные разности до $n-$го порядка; разделенные разности более высоких порядков равны нулю.

Пусть известны значения аппроксимируемой функции $f(x)$ в точках $x_0, x_1, \dots, x_n$. Интерполяционный многочлен, значения которого в узлах интерполяции совпадают со значениями функции $f(x)$, может быть записан в виде:

$$P_n(x) = f(x_0) + (x - x_0) \cdot f(x_0, x_1) + (x - x_0)(x - x_1) \cdot f(x_0, x_1, x_2) + \dots +$$

$$+(x - x_0)(x - x_1)\dots(x - x_n) \cdot f(x_0, x_1, \dots, x_n).$$

Запись многочлена выше есть так называемый **интерполяционный многочлен Ньютона.** Если функция $f(x)$ не есть многочлен $n-$й степени, то формула для

$P_n(x)$ приближает функцию $f(x)$ с некоторой погрешностью. Отметим, что при добавлении новых узлов первые члены многочлена Ньютона остаются неизменными. Если функция задана в точках $x_0, x_1, \ldots, x_n$, то при построении интерполяционного многочлена Ньютона удобно пользоваться таблицей, называемой таблицей разделенных разностей, пример которой для $n = 4$ приведен в таблице 1.

Таблица разделенных разностей

x_0	$f(x_0)$				
x_1	$f(x_1)$	$f(x_0, x_1)$			
x_2	$f(x_2)$	$f(x_1, x_2)$	$f(x_0, x_1, x_2)$		
x_3	$f(x_3)$	$f(x_2, x_3)$	$f(x_1, x_2, x_3)$	$f(x_0, x_1, x_2, x_3)$	
x_4	$f(x_4)$	$f(x_3, x_4)$	$f(x_2, x_3, x_4)$	$f(x_1, x_2, x_3, x_4)$	$f(x_0, x_1, x_2, x_3, x_4)$

Для повышения точности интерполяции в сумму P_n могут быть добавлены новые члены, что требует подключения дополнительных интерполяционных узлов. При этом безразлично, в каком порядке подключаются новые узлы. Этим формула Ньютона выгодно отличается от формулы Лагранжа.

Погрешность интерполяционных многочленов Лагранжа и Ньютона для случая аналитически заданной функции $f(x)$ априорно может быть оценена по формуле

$$|\varepsilon_n(x)| = |f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|,$$

где $M_{n+1} = max|f^{(n+1)}(\xi)|, \quad \xi \in [x_0, x_n]$.

Если величину производных аппроксимируемой функции оценить сложно (например, для таблично заданной функции), то используется апостериорная оценка по первому отброшенному члену интерполяционного многочлена Ньютона, в который входят разделенные разности, являющиеся аналогами производных соответствующих порядков.

2 Протокол

Входные данные я храню в файле *test3.1-1.t* и *test3.1-2.t*: первая строка — число интерполяционных узлов, вторая строка — их значения, третья строка — точка, в которой необходимо вычислить значение погрешности интерполяции.
Выходные данные я вывожу на экран.
Скриншот консоли:

$a) X_i = 0, \frac{\pi}{6}, \frac{2\pi}{6}, \frac{3\pi}{6}; \quad X^\star = 1.0.$

```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ g++ 3.1.cpp -o 3.1
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ cat test3.1-1.t
4
0 0.52359877559 1.0471975512 1.57079632679
1.0
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ ./3.1 < test3.1-1.t
Многочлен Лагранжа:
L_4(x) = 3.56536x(x - 1.0472)(x - 1.5708)-6.66407x(x - 0.523599)(x - 1.5708) + 2.98484x(x - 0.523599)(x - 1.0472)
Значение многочлена Лагранжа в точке X*:
L_4(1) = 1.84109
Точное значение функции в точке X*:
f(1) = 1.84147
Значение погрешности интерполяции м-м Лаг-жа: 0.000384968

Многочлен Ньютона:
P_4(x) = 1.95493x-0.24434x(x - 0.523599)-0.113872x(x - 0.523599)(x - 1.0472)
Значение многочлена Ньютона в точке X*:
P_4(1) = 1.84109
Точное значение функции в точке X*:
f(1) = 1.84147
Значение погрешности интерполяции м-м Ньютона: 0.000384968
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$
```

Многочлен Лагранжа

$$L_4(x) = 3.56536x(x - 1.0472)(x - 1.5708) - 6.66407x(x - 0.523599)(x - 1.5708) +$$

$$+2.98484x(x - 0.523599)(x - 1.0472)$$

Значение многочлена Лагранжа в точке $X^\star$

$$L_4(X^\star) = 1.84109$$

Значение погрешности интерполяции методом Лаг-жа: 0.000384968

Многочлен Ньютона

$$P_4(x) = 1.95493x - 0.24434x(x - 0.523599) - 0.113872x(x - 0.523599)(x - 1.0472)$$

Значение многочлена Ньютона в точке $X^\star$

$$P_4(X^\star) = 1.84109$$

Значение погрешности интерполяции методом Ньютона: 0.000384968

$b) X_i = 0, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{2}; \quad X^\star = 1.0.$

```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ cat test3.1-2.t
4
0 0.52359877559 0.78539816339 1.57079632679
1.0
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ ./3.1 < test3.1-2.t
Многочлен Лагранжа:
L_4(x) = 7.13073x(x - 0.785398)(x - 1.5708)-9.24203x(x - 0.523599)(x - 1.5708) + 1.98989x(x - 0.523599)(x - 0.785398)
Значение многочлена Лагранжа в точке X*:
L_4(1) = 1.84314
Точное значение функции в точке X*:
f(1) = 1.84147
Значение погрешности интерполяции м-м Лаг-жа: 0.00166512

Многочлен Ньютона:
P_4(x) = 1.95493x-0.208608x(x - 0.523599)-0.121411x(x - 0.523599)(x - 0.785398)
Значение многочлена Ньютона в точке X*:
P_4(1) = 1.84314
Точное значение функции в точке X*:
f(1) = 1.84147
Значение погрешности интерполяции м-м Ньютона: 0.00166512
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$
```

Многочлен Лагранжа

$$L_4(x) = 7.13073x(x - 0.785398)(x - 1.5708) - 9.24203x(x - 0.523599)(x - 1.5708) +$$

$$+1.98989x(x - 0.523599)(x - 0.785398)$$

Значение многочлена Лагранжа в точке $X^\star$

$$L_4(X^\star) = 1.84314$$

Значение погрешности интерполяции методом Лаг-жа: 0.00166512

Многочлен Ньютона

$$P_4(x) = 1.95493x - 0.208608x(x - 0.523599) - 0.121411x(x - 0.523599)(x - 0.785398)$$

Значение многочлена Ньютона в точке $X^\star$

$$P_4(X^\star) = 1.84314$$

Значение погрешности интерполяции методом Ньютона: 0.00166512

Можно заметить, что оба метода дают один и тот же результат как в тесте 1, так и в тесте 2.

3 Исходный код

Листинг 1: Метод Лагранжа и метод Ньютона

```
#include <iostream>
#include <math.h>
#include <vector>

double F(double x)
{
    double val = std::sin(x) + x;
    return val;
}

double OmegaVal(double index, std::vector<double>& x)
{
    double val = 1;
    int n = x.size();

    for(int i = 0; i < n; i++)
        if(i == index)
            continue;
        else
            val *= (x[index] - x[i]);
    return val;
}

double NewtonVal(std::vector<double>& P, std::vector<double>& x, double x_star)
{
    int n = P.size();
    double res;
    double value = 0;

    for(int i = 0; i < n; i++)
    {
        res = 1;
        for(int j = 0; j < i; j++)
            res *= (x_star - x[j]);
        value += P[i]*res;
    }
    return value;
}

```

```
40
41 double LagrangeVal(std::vector<double>& L, std::vector<double>& x, double x_star)
42 {
43     int n = L.size();
44     double res;
45     double value = 0;
46
47     for(int i = 0; i < n; i++)
48     {
49         res = 1;
50         for(int j = 0; j < n; j++)
51             if(j == i)
52                 continue;
53             else
54                 res *= (x_star - x[j]);
55         value += L[i]*res;
56     }
57     return value;
58 }
59
60 void PrintNewton(std::vector<double>& P, std::vector<double>& x, int n)
61 {
62     for(int i = 0; i < n; i++)
63     {
64         if(P[i] == 0)
65             continue;
66         std::cout << P[i];
67         for(int j = 0; j < i; j++)
68         {
69             if(x[j] == 0)
70                 std::cout << "x";
71             else if(x[j] > 0)
72                 std::cout << "(x - " << x[j] << ")";
73             else
74                 std::cout << "(x + " << -x[j] << ")";
75         }
76         if(P[i+1] > 0)
77             std::cout << " + ";
78     }
79     std::cout << std::endl;
80 }
81
82 void PrintLagrange(std::vector<double>& L, std::vector<double>& x, int n)
83 {
```

```
84      for(int i = 0; i < n; i++)
85      {
86          if(L[i] == 0)
87              continue;
88          std::cout << L[i];
89          for(int j = 0; j < n; j++)
90              if(i == j)
91                  continue;
92              else
93              {
94                  if(x[j] == 0)
95                      std::cout << "x";
96                  else if(x[j] > 0)
97                      std::cout << "(x - " << x[j] << ")";
98                  else
99                      std::cout << "(x + " << -x[j] << ")";
100             }
101             if(L[i+1] > 0)
102                 std::cout << " + ";
103     }
104     std::cout << std::endl;
105 }
106
107
108 double SplitDiff(int i, int j, std::vector<double>& x, std::vector<double>& f)
109 {
110     double val;
111     if(j - i == 1)
112         val = (f[i] - f[j])/(x[i]-x[j]);
113     else
114         val = (SplitDiff(i, j-1, x, f) - SplitDiff(i+1, j, x, f))/(x[i]-x[j]);
115     return val;
116 }
117
118 int main()
119 {
120     double x_star, val, fval;
121     int n;
122     std::cin >> n;
123
124     std::vector<double> x(n);
125     std::vector<double> f(n);
126     std::vector<double> L(n);
127     std::vector<double> P(n);
```

```
128
129     for(int i = 0; i < n; i++)
130     {
131         std::cin >> x[i];
132         f[i] = F(x[i]);
133     }
134
135     std::cin >> x_star;
136     for(int i = 0; i < n; i++)
137         L[i] = f[i]/OmegaVal(i, x);
138     for(int i = 0; i < n; i++)
139         if(i == 0)
140             P[i] = f[i];
141         else
142             P[i] = SplitDiff(0, i, x, f);
143
144     std::cout << "Lagrange polynomial:" << std::endl << "L_" << n << "(x) = ";
145     PrintLagrange(L, x, n);
146
147     val = LagrangeVal(L, x, x_star);
148     fval = F(x_star);
149     std::cout << "The value of the Lagrange polynomial at X*:" << std::endl << "L_" << n
            << "(" << x_star << ") = " << val << std::endl;
150     std::cout << "The exact value of the function at X*:" << std::endl << "f(" << x_star <<
            ") = " << fval << std::endl;
151     std::cout << "The value of the interpolation error of Lagrange method:" << std::fabs(fval -
            val) << std::endl << std::endl;
152
153     std::cout << "Newton's polynomial:" << std::endl << "P_" << n << "(x) = ";
154     PrintNewton(P, x, n);
155
156     val = NewtonVal(P, x, x_star);
157     std::cout << "The value of the Newton's polynomial at X*:" << std::endl << "P_" << n
            << "(" << x_star << ") = " << val << std::endl;
158     std::cout << "The exact value of the function at X*:" << std::endl << "f(" << x_star <<
            ") = " << fval << std::endl;
159     std::cout << ""The value of the interpolation error of Newton's method:" << std::fabs(fval -
             val) << std::endl;
160
161     return 0;
162 }
```

4 Выводы

Выполнив первое задание третье лабораторной работы, я изучила два метода интерполяции функции. Для решения этой задачи используется приближающая функция, которая в каждой точке некоторого множества интерполяционных узлов принимает значения интерполируемой функции. В данном задании рассматриваются две приближающие функции – интерполяционный многочлен Лагранжа и интерполяционный многочлен Ньютона. Недостатком многочлена Лагранжа является необходимость пересчета всех его коэффициентов в случае добавления новых интерполяционных узлов. В этом смысле метод Ньютона является более "гибкимоэффициенты многочлена Ньютона являются разделенными разностями какого-либо порядка, так что при добавлении новых узлов, уже вычисленные коэффициенты не изменяются. В рамках лабораторной работы, когда множество узлов фиксировано и не дополняется, оба метода дают одинаковые результаты.