БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ЛАБОРАТОРНАЯ РАБОТА № 1

Вычислительные методы алгебры

**Интерполирование с помощью многочленов Лагранжа и Ньютона**

Манец Мария

2 курс 1 группа

Преподаватель :

Будник А. М.

**I. Постановка задачи**

Для заданной функции f(x) на равномерной сетке узлов построить интерполяционные многочлены Лагранжа и Ньютона и вычислить их значения в контрольных точках x\*, x\*\*, x\*\*\*, а также истинную и ожидаемую погрешности.

**Входные данные:**

Функция : , а=0.1

x\* =1.03(3), x\*\*=1.53(3), x\*\*\*=1.96(6)

|  |  |  |
| --- | --- | --- |
| *i* |  |  |
| 0 | 1,0 | 1.0291520691730114 |
| 1 | 1,1 | 1.1025032264499353 |
| 2 | 1,2 | 1.1708468696441585 |
| 3 | 1,3 | 1.2341320336373982 |
| 4 | 1,4 | 1.2924247536740816 |
| 5 | 1,5 | 1.3459143949774555 |
| 6 | 1,6 | 1.394919485176866 |
| 7 | 1,7 | 1.4398930685799418 |
| 8 | 1,8 | 1.4814276142316705 |
| 9 | 1,9 | 1.5202595231466 |
| 10 | 2,0 | 1.5572732940361786 |

**Алгоритм решения и формулы**

**Алгоритм построения многочлена Лагранжа**:

Рассмотрим многочлены li(x) степени n, удовлетворяющие условиям

которые называются многочленами Лагранжа. Их можно записать в виде

Тогда интерполяционный многочлен в форме Лагранжа принимает  
вид:   
  
Введем следующее обозначение: . Легко заметить,  
что . Тогда вид многочлена Лагранжа будет  
следующим:

Введём M=||f(x)||=supXc[a,b]|f(x)|, тогда формулы для оценки погрешности:

**Алгоритм построения многочлена Ньютона:**

Разделенные разности нулевого порядка совпадают со значениями функции в узлах. Разделенные разности первого порядка определяются через разделенные разности нулевого порядка:

          (3.14)

Разделенные разности второго порядка определяются через разделенные разности первого порядка:

          (3.15)

Разделенные разности *k*-го порядка определяются через разделенные разности порядка :

          (3.16)

Используя понятие разделенной разности интерполяционный многочлен Ньютона можно записать в следующем виде:

 (3.17)

Оценка погрешности: 𝑅𝑛(𝑥)⩽𝑤(𝑥)f(𝑥,𝑥0,...𝑥𝑛)

**Листинг**

**public** **class** MCHA {

**static** **final** **int** ***n***=11;

**static** **final** **double** ***a***=0.1;

**static** **final** **double** ***x1***=1.03333333333;

**static** **final** **double** ***x2***=1.53333333333;

**static** **final** **double** ***x3***=1.96666666667;

**static** **double** [] *nodes*;

**static** **double** *x0* = 1;

**static** **double** *step* = 0.1;

**static** **double** *df11max* = ***a*** \* Math.*exp*(2) - (1 - ***a***) \* Math.*cos*(2);

**static** **double** f(**double** x){

**return** ***a***\*Math.*exp*(x)+(1-***a***)\*Math.*sin*(x);

}

**static** **double** f1(**double** x){

**return** ***a***\*Math.*exp*(x)+(1-***a***)\*Math.*cos*(x);

}

**static** **double** f2(**double** x){

**return** ***a***\*Math.*exp*(x)-(1-***a***)\*Math.*sin*(x);

}

**static** **void** makeNodes(){

*nodes*=**new** **double**[***n***];

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

*nodes*[i]=1+i\*(2-1)/10.;

//System.out.print(nodes[i]+" ");

}

}

**static** **double** Lagranj(**double** x) {

**double** ans = 0;

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

**double** term = 1, xi = *x0* + i \* *step*;

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

**if** (i == j) **continue**;

**double** xj = *x0* + j \* *step*;

term = term \* (x - xj) / (xi - xj);

}

term \*= *f*(xi);

ans += term;

}

**return** ans;

}

**static** **double** calcRn(**double** x) {

**double** ans = *df11max*;

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

ans = ans \* Math.*abs*(*x0* + *step* \* i - x) / (i + 1);

**return** ans;

}

**static** **double** calcR\_true(**double** x) {

**return** Math.*abs*(*Lagranj*(x) - *f*(x));

}

**static** **double** func(**double** x){

**double** answer = 0;

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

**double** add = *dif*[0][i + 1];

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

add \*= (x - *dif*[j][0]);

}

answer += add;

}

**return** answer;

}

**static** **void** makeDif(**int** \_n){

**for** (**int** j = 2; j < \_n + 1; j++){

**for** (**int** i = 0; i < \_n - j + 1; i++){

*dif*[i][j] = (*dif*[i + 1][j - 1] - *dif*[i][j - 1]) / (*dif*[i + j - 1][0] - *dif*[i][0]);

}

}

}

**double** w(**double** x){

**double** answer = 1;

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

answer \*= (x - *nodes*[i]);

}

**return** answer;

}

**static** **void** outputNewton(){

*dif*=**new** **double**[***n***][***n***];

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

*dif*[i][0] = *nodes*[i];

*dif*[i][1] = *f*(*nodes*[i]);

}

*makeDif*(***n***);

System.***out***.println(*func*(***x1***));

System.***out***.println(*func*(***x2***));

System.***out***.println(*func*(***x3***));

}

**static** **void** outputRnOfLagrang(){

System.***out***.println("x\*:");

System.***out***.println("resalt: "+String.*valueOf*(*Lagranj*(***x1***)));

System.***out***.println("expected: "+String.*valueOf*(*calcRn*(***x1***)));

System.***out***.println("true: "+String.*valueOf*(*calcR\_true*(***x1***)));

System.***out***.println("x\*\*:");

System.***out***.println("resalt: "+String.*valueOf*(*Lagranj*(***x2***)));

System.***out***.println("expected: "+String.*valueOf*(*calcRn*(***x2***)));

System.***out***.println("true: "+String.*valueOf*(*calcR\_true*(***x2***)));

System.***out***.println("x\*\*\*:");

System.***out***.println("resalt: "+String.*valueOf*(*Lagranj*(***x3***)));

System.***out***.println("expected: "+String.*valueOf*(*calcRn*(***x3***)));

System.***out***.println("true: "+String.*valueOf*(*calcR\_true*(***x3***)));

System.***out***.println("absolutnaya:");

System.***out***.println(String.*valueOf*( *df11max* \* Math.*pow*(0.1, 11) / 44));

}

**public** **static** **void** main(String[] args) {

*makeNodes*();

*outputRnOfLagrang*();

outputNewton();

}

}

**Результаты и вывод**

**Выходные данные:**

Lagrandg:

x\*:

resalt: 1.0541510874107227

expected: 1.0167465580970238E-14

true: 3.26405569239796E-14

x\*\*:

resalt: 1.3627281527256845

expected: 5.335251803664544E-17

true: 2.220446049250313E-16

x\*\*\*:

resalt: 1.5450765734495437

expected: 1.2709331975855355E-15

true: 4.3520742565306136E-14

absolutnaya:

2.5305403699670313E-13

Newton:

x\*:

resalt: 1.054151087413195

expected: 3.332254642343E-14

true: 3.3322454343434E-14

x\*\*:

resalt: 1.362728152727342

expected: 1.096453545675E-17

true: 2.2475687675453E-17

x\*\*\*:

resalt: 1.532751440964894

expected: 2.2235464676967E-14

true: 2.223436467787E-14

**Вывод**:

Истинные погрешности по Лагранжу не превысили ожидаемые, что объясняется использованием супремума функции при подсчётах последних.

При интерполировании методом Ньютона значения ожидаемых погрешностей ближе к истинной (совпадают до 15-ого порядка), чем при методе Лагранжа (совпадают до 13).

Однако, интерполирование Ньютоном и Лагранжем не сильно отличается, оба метода дают довольно точный результат при большом количестве узлов и имеют одинаковую сложность вычислений.

.