БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ЛАБОРАТОРНАЯ РАБОТА № 3

Методы численного анализа

**Интерполирование с помощью многочлена Чебышева**

Манец Мария

2 курс 1 группа

Будник А. М.

I. Постановка задачи

Минимизировать остаток интерполирования, используя многочлен Чебышева. Для этого с его помощью выберем узлы интерполирования на нужном нам отрезке (в рамках этой лабораторной [1,2]) и проведём по ним интерполяцию многочленом Лагранжа. Вычислим его значения в контрольных точках х\*, х\*\* и х\*\*\*. Оценим результат и сравним его с предыдущими.

Входные данные:

Функция : , а=0.1

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

Алгоритм решения и формулы

Идея метода состоит в выборе таких узлов для построения интерполяционного полинома, чтобы в оценке

многочлен был наименее отклоняющимся от нуля, что достигается на чебышёвском наборе узлов

В таком случае оценка погрешности принимает вид

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

**static** **double** [] *fx*=**new** **double**[***n***];

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

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

**static** **double**[][] *konechRazn*;

**static** **double** *accuracy* = 1e-6;

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

**static** **double** *right*=2;

**static** **double** *left*=1;

**static** **double** *h*;

**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** **double** f3(**double** x ){

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

}

**static** **int** factorial(**int** x){

**if** (x==1 || x==0)

**return** 1;

**return** x\**factorial*(x-1);

}

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

**double** ans = 0;

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

**double** term = 1, xi = *nodes*[i];

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

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

**double** xj = *nodes*[j];

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** **void** createNodes (){

*h* = (*right* - *left*) / (***n***-1);

**double** add = (*right* + *left*) / 2., mul = (*right* - *left*) / 2.;

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

*nodes*[i] = add + mul \* Math.*cos*(Math.***PI*** \* (2 \* (***n*** - i) + 1) / 2 / (***n*** + 1));

*fx*[i] = *f* (*nodes*[i]);

}

}

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

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

}

**static** **double** findTheoreticalResidual ()

{

**double** mul = (*right* - *left*) / 2.;

**for** (**int** i = 1; i < ***n***; ++i)

mul \*= ((*right* - *left*) / (i + 1) / 4.);

**switch** (***n*** % 4)

{

**case** 0:

**return** *f1*(*right*) \* mul;

**case** 1:

**return** *f2*(*right*) \* mul;

**case** 2:

**return** *f3*(*right*) \* mul;

**case** 3:

**return** *f*(*right*) \* mul;

}

**return** 0;

}

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

*createNodes*();

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

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

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

}

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

System.***out***.println("expected: "+*findTheoreticalResidual*());

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

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

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

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

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

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

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

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

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

}

}

Результаты и вывод

Выходные данные:

nodes:

1.0042775693130948 1.0380602337443565 1.1033233298543825 1.1956192854956398 1.3086582838174552 1.4347369038899742 1.5652630961100258 1.6913417161825448 1.8043807145043602 1.8966766701456175 1.9619397662556435

expected: 1.8602838205312818E-14

x\*:

resalt: 1.0541510874107547

true: 6.661338147750939E-16

x\*\*:

resalt: 1.3627281527256858

true: 1.5543122344752192E-15

Вывод:

Данный метод является одним из самых точных методов из числа рассмотренных, в силу того, что в этом методе происходит выбор самих узлов. Выбранные узлы являются корнями многочлена Чебышева, а многочлен Чебышева есть наиболее близкая к нулю функция, то есть мы приближаем остаток интерполирования к нулю наилучшим образом.

Однако в нашем случае результат стал более точным только у первой контрольной точки, в то время как у третьей он не изменился, а у второй стал хуже, чем во всех предыдущих результатах. Это можно списать на то, что вначале рассматриваемого отрезка «концентрация узлов» вышла более «плотной», поэтому точность понижается при приближении к третьей точке.