Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики
Кафедра вычислительной математики

Ярмолкевич Даниил Андреевич

Лабораторная работа №1 по предмету

«Методы численного анализа»

Студента 2 курса 2 группы

Преподаватель
Полищук Максим Александрович

Минск, 2018

Оглавление

Задание 1

Постановка задачи:

В соответствии с вариантом задания (равным номеру в списке академической группы) произвести табулирование функции $f(x)$, используя чебышевскую сетку с m узлами, где $m \in \{3,4,5,6,8\}$. Для каждого m построить интерполяционный многочлен Ньютона в начале отрезка на чебышевской сетке.

В отчёте представить значения аргумента x_n, приближенные значения интерполяционного многочлена $P(x_n)$, точные значения функции $f(x_n)$. Вывод всех величин организовать в таблицу. Для каждого m оценить погрешность интерполирования на отрезке (в равномерной норме). Сравнить со значениями оценок погрешностей $|r_m(x_n)|$ в серединах отрезков между соседними узлами при равномерном интерполировании на m узлах (в точках $x_n = \frac{b-a}{2(m-1)}(2n+1), n = \overline{0, m-2}$ интервала [a;b]).

Для вычислений использовать тип *float*.

Входные данные:

Вариант 7 (24%17): $10x^3(5\pi + x)^{-1/4}$,[0;10]

Использованные формулы

1) Чебышевская сетка узлов: $x_i = \frac{b+a}{2} + \frac{b-a}{2}\cos(\frac{\pi(2i+1)}{2n}),\ i = \overline{0, n-1}$
2) Оценка погрешности интерполирования на отрезке (в равномерной норме):

$$\|f - P_{n-1}(x)\| \le \frac{\|f^{(n)}\|(b-a)^n 2^{1-2n}}{n!}$$

3) Разделенные разности: $f(x_i, ..., x_{i+k}) = \frac{f(x_{i+1}, ..., x_{i+k}) - f(x_i, ..., x_{i+k-1})}{x_{i+k} - x_i}$
4) Интерполяционный многочлен Ньютона в начале отрезка на чебышевской сетке:
$P_{(n)}(x) = x_0 + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + ... + f(x_0, x_1, ..., x_{n-1})(x - x_0)...(x - x_{n-1})$
5) Оценка погрешности в точке на равномерной сетке узлов при интерполировании с помощью формулы Ньютона для начала таблицы:
$\|f - P_{n-1}(x)\| \le \left|\frac{Mh^{n+1}t(t-1)...(t-n)}{(n+1)!}\right|$

Результат работы программы и анализ полученных результатов

```
m=4:
x          function      polynomial
0,3806023  0,2752866     0,2752866
3,0865829  141,2296448    141,2296448
6,9134173  1515,1251221   1515,1251221
9,6193981  3967,7675781   3967,7678223
Best mistake: 6.2456865
Mistakes: 9.252869 5.5517206 9.252868

m=5:
x         function       polynomial
0,2447174  0,0733308     0,0733308
2,0610738  42,6446533    42,6446533
5,0000000  585,9703979    585,9703979
7,9389262  2269,0380859   2269,0380859
9,7552824  4132,7690430   4132,7690430
Best mistake: 0.62126994
Mistakes: 1.019271 0.43683046 0.43683046 1.019271

m=6:
x  function   polynomial
0,1703709  0,0247733  0,0247733
1,4644661  15,4287252 15,4287252
3,7059047  242,4681549    242,4681396
6,2940950  1151,2882080   1151,2880859
8,5355339  2802,4848633   2802,4846191
9,8296289  4224,8994141   4224,8994141
Best mistake: 0.07415864
Mistakes: 0.14352366 0.04784122 0.0341723 0.04784122 0.14352366

m=8:
x         function       polynomial
0,0960736  0,0044475     0,0044475
0,8426520  2,9664783     2,9664783
2,2221489  53,3241196    53,3241196
4,0245485  309,2828674    309,2828674
5,9754515  988,7357788    988,7357788
7,7778511  2137,3591309   2137,3588867
9,1573477  3438,8295898   3438,8293457
9,9039268  4318,2905273   4318,2905273
Best mistake: 0.0012973116
Mistakes: 0.0038925828 8.982884E-4 4.083129E-4 3.175767E-4 4.0831283E-4 8.982884E-
4 0.0038925796

m=9:
x         function       polynomial
0,0759612  0,0021990     0,0021990
0,6698730  1,4942157     1,4942158
1,7860620  27,8591042    27,8591061
3,2898993  170,5578308    170,5578461
5,0000000  585,9703979    585,9703369
6,7101007  1388,4733887   1388,4735107
8,2139378  2505,8513184   2505,8510742
9,3301268  3630,8837891   3630,8837891
9,9240389  4343,7993164   4343,7988281
Best mistake: 1.8066436E-4
Mistakes: 6.984924E-4 1.3969847E-4 5.373019E-5 3.4191933E-5 3.4191937E-5
5.373019E-5 1.3969849E-4 6.984924E-4
```

Как видим, погрешность интерполирования уменьшается с возрастанием количества узлов. Самое точное приближение получено при m=9.

Листинг программы

Файл “util.CommonAnalysis.java”

```
package util;

import java.util.ArrayList;
import java.util.Collections;
import java.util.function.Function;

import static java.lang.Math.PI;
import static java.lang.Math.cos;
import static java.lang.Math.pow;

/**
 * Contains static common functions, connected with numerical analysis
 */
public class CommonAnalysis {

    /**
     * Table of divided differences of function <b><i>f</i></b>
defined on points <b><i>x</i></b>
     * @param x list of points
     * @param f function
     * @return table of divided differences, (i,j)=<i>f(x<sub>(j-
i)</sub> , ... ,x<sub>(j)</sub>)</i>
     */
    public static float[][] table(ArrayList<Float> x,
                                  Function<Float, Float> f) {
        float[][] result = new float[x.size()][x.size()];
        for (int i = 0; i < x.size(); i++) {
            result[0][i] = f.apply(x.get(i));
        }
        for (int i = 1; i < x.size(); i++) {
            for (int j = i; j < x.size(); j++) {
                result[i][j] = (result[i - 1][j] - result[i - 1][j -
1]) / (x.get(j) - x.get(j - i));
            }
        }
        return result;
    }

    /**
     * Mistake of interpolation in chebyshev nodes
     * @param a start of interval
     * @param b end of interval
     * @param n degree of polynom
     * @param nthDerivative nth derivative of function
     * @return numeric value of mistake
     */
    public static float bestMistake(float a,
                                    float b,
                                    int n,
```

```
                                    Function<Float, Float>
nthDerivative) {
        return (float) (maxAbsValue(a, b, nthDerivative, 0.01f) *
pow(b - a, n) * pow(2, 1 - 2 * n) / factorial(n));
    }

    /**
     * Factorial
     * @param n - argument, n>=0
     * @return numeric value of factorial
     */
    public static int factorial(int n) {
        int result = 1;
        int next = 2;
        while (next <= n) {
            result *= next;
            next += 1;
        }
        return result;
    }

    /**
     * Maximum absolute value of function on step with defined step
     * @param a start of step
     * @param b end of step
     * @param f function
     * @param step step of finding values of function
     * @return numeric value of maximum absolute value
     */
    public static float maxAbsValue(float a,
                                    float b,
                                    Function<Float, Float> f,
                                    float step) {
        float result = 0;
        for (float i = a; i <= b; i += step) {
            float newResult = Math.abs(f.apply(i));
            if (newResult > result) {
                result = newResult;
            }
        }
        return result;
    }

    /**
     * Mistakes of interpolating on uniform nodes in the middles of
intervals
     * @param a start of interval
     * @param b end of interval
     * @param m number of points
```

```
     * @param nthDerivative nth derivative of function
     * @return list of numeric values of mistakes
     */
    public static ArrayList<Float> mistake(float a,
                                           float b,
                                           int m,
                                           Function<Float,Float>
nthDerivative) {
        float h = (b - a) / (m - 1);
        ArrayList<Float> mistakes=new ArrayList<>();
        for (int i = 0; i < m - 1; i++) {
            float point = (float) ((b - a) * 0.5 * (2f * i + 1) / (m
- 1f));
            float t = (point - a) / h;
            float mistake = maxAbsValue(a,b,nthDerivative,0.01f);
            for (int j = 0; j < m; j++) {
                mistake *= (t - j);
            }
            mistake = (float) (Math.abs(mistake) * pow(h, m) /
factorial(m));
            mistakes.add(mistake);
        }
        return mistakes;

    }

    /**
     * Chebyshev nodes in interval
     * @param a start of interval
     * @param b end of interval
     * @param m number of points
     * @return list of chebyshev nodes
     */
    public static ArrayList<Float> chebyshevNodes(float a,
                                                  float b,
                                                  int m) {
        ArrayList<Float> list = new ArrayList<>(m);
        for (int i = 0; i < m; i++) {
            float result = (float) ((b + a) / 2 + (b - a) / 2 *
cos(PI * (2 * i + 1) / (2 * m)));
            list.add(result);
        }
        Collections.reverse(list);
        return list;
    }

    /**
     * Uniform nodes in interval
     * @param a start of interval
```

```
     * @param b end of interval
     * @param n number of nodes
     * @return list of nodes
     */
    public static ArrayList<Float> uniformNodes(float a, float b,
int n) {
        ArrayList<Float> list=new ArrayList<>(n);
        for (int i = 0; i <= n; i++) {
            list.add(a+(b-a)*i/n);
        }
        return list;
    }

    /**
     * Function values in nodes
     * @param nodes - nodes
     * @param f - function
     * @return list of values
     */
    public static ArrayList<Float> functionValues(ArrayList<Float>
nodes, Function<Float,Float> f) {
        ArrayList<Float> list=new ArrayList<>(nodes.size());
        nodes.forEach(x -> list.add(f.apply(x)));
        return list;
    }
}
```

Файл "task1.NewtonPolynom.java"

```
package task1;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.function.Function;

/**
 * Class of Newton polynomial function, that is defined by list of nodes and list
of divided differences
 */
public class NewtonPolynom implements Function<Float, Float> {

    /**
     * List of divided differences
     */
    private ArrayList<Float> a;
    /**
     * List of nodes
     */
    private ArrayList<Float> nodes;

    /**
     * Constructs Newton polynomial function
     * @param a list of divided differences
     * @param nodes list of nodes
```

```
     */
    public NewtonPolynom(ArrayList<Float> a, ArrayList<Float> nodes) {
        this.a = a;
        this.nodes = nodes;
    }

    /**
     * Calculates the value of polynomial in point
     * @param x point
     * @return numerical value of <i>P(x)</i>
     */
    @Override
    public Float apply(Float x) {
        LinkedList<Float> list=new LinkedList<>();
        for (int i = 0; i < a.size(); i++) {
            float start=a.get(i);
            for (int j = 0; j < i; j++) {
                start=start*(x-nodes.get(j));
            }
            list.add(start);
        }
        return (float)list.stream().mapToDouble(hey->hey).sum();
    }
}
```

Файл “task1.NewtonChebyshevInterpolator.java”

```
package task1;

import util.CommonAnalysis;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.function.Function;

import static java.lang.Math.PI;
import static java.lang.Math.pow;

/**
 * Class, that interpolates input function by Newton polynom
 */
public class NewtonChebyshevInterpolator {

    /**
     * number of points
     */
    private int m;
    /**
     * function
     */
    private Function<Float, Float> f;
    /**
     * nth derivative of function <i><b>f</b></i>
     */
    private Function<Float, Float> nthDerivative;
    /**
```

```
     * start of interval
     */
    private float a;
    /**
     * end of interval
     */
    private float b;
    /**
     * list of chebyshev nodes
     */
    private ArrayList<Float> nodeList;
    /**
     * Newton polynomial function
     */
    private NewtonPolynom polynom;
    /**
     * mistake of interpolating in chebyshev nodes
     */
    private float bestMistake;
    /**
     * mistakes of interpolating function in uniform nodes in the middles of
intervals
     */
    private ArrayList<Float> mistakes;

    public static void main(String[] args) {
        ArrayList<Integer> m = new ArrayList<>(Arrays.asList(4, 5, 6, 8, 9));
        HashMap<Integer, Function<Float, Float>> derivatives = new HashMap<>();
        derivatives.put(4, x -> (float) (-15 * (64000 * PI * PI * PI + 14400 * PI
* PI * x + 1680 * PI * x * x + 77 * x * x * x) / (128 * pow(5 * PI + x, 17. /
4))));
        derivatives.put(5, x -> (float) (75 * (160000 * PI * PI * PI + 24000 * PI
* PI * x + 2100 * PI * x * x + 77 * x * x * x) / (512 * pow(5 * PI + x, 21. /
4))));
        derivatives.put(6, x -> (float) (-675 * (320000 * PI * PI * PI + 36000 *
PI * PI * x + 2520 * PI * x * x + 77 * x * x * x) / (2048 * pow(5 * PI + x, 25. /
4))));
        derivatives.put(8, x -> (float) (-1044225 * (128000 * PI * PI * PI + 9600
* PI * PI * x + 480 * PI * x * x + 11 * x * x * x) / (32768 * pow(5 * PI + x, 33.
/ 4))));
        derivatives.put(9, x -> (float) (3132675 * (1344000 * PI * PI * PI + 86400
* PI * PI * x + 3780 * PI * x * x + 77 * x * x * x) / (131072 * pow(5 * PI + x,
37. / 4))));
        m.forEach(M -> {
            try {
                NewtonChebyshevInterpolator interpolator = new
NewtonChebyshevInterpolator(x -> (float) (10 * x * x * x * Math.pow(Math.PI * 5 +
x, -0.25)),
                        0, 10, M,
                        derivatives.get(M));
                interpolator.createOutput();
            } catch (IOException exc) {
                exc.printStackTrace();
            }
        });
    }

    /**
     * Constructs the interpolator class, processes data
```

```
     * @param f function
     * @param a start of interval
     * @param b end of interval
     * @param m number of points
     * @param nthDerivative nth derivative of function
     */
    public NewtonChebyshevInterpolator(Function<Float, Float> f,
                                       float a,
                                       float b,
                                       int m,
                                       Function<Float, Float> nthDerivative) {
        this.m = m;
        this.f = f;
        this.a = a;
        this.b = b;
        this.nthDerivative = nthDerivative;
        processData();
    }

    /**
     * Processes input data
     */
    private void processData() {
        nodeList = CommonAnalysis.chebyshevNodes(a, b, m);
        float[][] table = CommonAnalysis.table(nodeList, f);
        ArrayList<Float> coefficients = new ArrayList<>();
        for (int i = 0; i < m; i++) {
            coefficients.add(table[i][i]);
        }
        polynom = new NewtonPolynom(coefficients, nodeList);
        bestMistake = CommonAnalysis.bestMistake(a, b, m, nthDerivative);
        mistakes = CommonAnalysis.mistake(a, b, m, nthDerivative);
    }

    /**
     * Creates output
     * @throws IOException
     */
    public void createOutput() throws IOException {
        PrintWriter writer = new PrintWriter("task1out/m" + m + ".txt");
        writer.println("m=" + m + ":");
        writer.println("x\tfunction\tpolynomial");
        nodeList.forEach(X -> {
            writer.printf("%.7f", X);
            writer.print("\t");
            writer.printf("%.7f", f.apply(X));
            writer.print("\t");
            writer.printf("%.7f", polynom.apply(X));
            writer.println();
        });
        writer.println("Best mistake: " + bestMistake);
        writer.print("Mistakes: ");
        mistakes.forEach(x -> writer.print(x + " "));
        writer.close();
    }

}
```