Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ

к лабораторной работе

на тему

Численное решение нелинейных уравнений

Выполнил: студент группы 153501
Тимофеев Кирилл Андреевич

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Вариант 8

Цель выполнения задания:

- изучить методы численного решения нелинейных уравнений: методов бисекции, хорд, простой итерации, релаксации, метода Ньютона и его модификаций;
- Изучение метода Эйткена ускорения сходимости;
- Сравнение числа итераций, необходимого для достижения заданной точности вычисления разными методами;

Краткие теоретические сведения:

Численное решение нелинейного уравнения $f(x)=0$ заключается в вычислении с заданной точностью значения всех или некоторых корней уравнения и распадается на несколько задач: *во-первых*, надо исследовать количество и характер корней (вещественные или комплексные, простые или кратные), *во-вторых*, определить их приближенное расположение, т.е. значения начала и конца отрезка, на котором лежит только один корень, *в-третьих*, выбрать интересующие нас корни и вычислить их с требуемой точностью. Вторая задача называется ***отделением корней***. Решив ее, по сути дела, находят приближенные значения корней с погрешностью, не превосходящей длины отрезка, содержащего корень. Отметим два простых приема отделения действительных корней уравнения - *табличный* и *графический*. Первый прием состоит в вычислении таблицы значений

функции $f(x)$ в заданных точках x_i и использовании следующих теорем математического анализа:

1. *Если функция $y=f(x)$ непрерывна на отрезке $[a,b]$ и $f(a)f(b)<0$, то внутри отрезка $[a,b]$ существует по крайней мере один корень уравнения $f(x)=0$.*

2. *Если функция $y=f(x)$ непрерывна на отрезке $[a,b]$, $f(a)f(b) < 0$ и $f'(x)$ на интервале (a,b) сохраняет знак, то внутри отрезка $[a,b]$ существует единственный корень уравнения $f(x)=0$.*

Для определения числа корней на заданном промежутке используется Теорема Штурма: Если f (x) многочлен и уравнение не имеет кратных корней на промежутке[a, b], то число корней уравнения f (x) = 0, лежащих на промежутке [a, b], совпадает с числом N(a) – N(b), которое определяется из следующей процедуры.

Строим ряд Штурма $f_0(x), f_1(x), f_2(x), \ldots, f_m(x)$, *где*

$f_0(x) = f(x)$;

$f_1(x) = f'(x)$;

$f_0(x)$ *делим на* $f_1(x)$ *и в качестве* $f_2(x)$ *берем остаток от деления, взятый с обратным знаком;*

$f_1(x)$ *делим на* $f_2(x)$ *и в качестве* $f_3(x)$ *берем остаток от деления, взятый с обратным знаком;*

и т.д.

Полагаем N(a) – число перемен знака в ряде Штурма, если вместо x подставлена точка a, N(b) – число перемен знака в ряде Штурма, если вместо x подставлена точка b.

Метод простых итераций. Вначале уравнение $f(x)=0$ преобразуется к эквивалентному уравнению вида $x=\varphi(x)$. Это можно сделать многими способами, например, положив $\varphi(x)=x+\blacklozenge(x)f(x)$, где $\blacklozenge(x)$ - произвольная непрерывная знакопостоянная функция. Выбираем некоторое начальное приближение x_0 и вычисляем дальнейшие приближения по формуле

$$x_k=\varphi(x_{k-1}), \quad k=0,1,\ldots$$

Метод хорд. Пусть дано уравнение $f(x)=0$, $a \le x \le b$, где $f(x)$ - дважды непрерывно дифференцируемая функция.

Берется некоторое начальное приближение. Тогда если функция вогнута, то используется рекуррентная формула

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f(a)-f(x_{n-1})} \cdot (a - x_{n-1}) \quad n = 1,2,..$$

$$x_0 = b \ .$$

Если функция выпукла

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f(b)-f(x_{n-1})} \cdot (b - x_{n-1}) \quad n = 1,2,... \ ,$$

$$x_0 = a \ .$$

Метод Ньютона (касательных). Для начала вычислений требуется задание одного начального приближения x_0, последующие приближения вычисляются по формуле

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad f'(x_n) \neq 0.$$

Метод имеет квадратичную скорость сходимости для простого корня, но очень чувствителен к выбору начального приближения. При произвольном начальном приближении итерации сходятся, если всюду $|f(x)f''(x)| < (f'(x))^2$, в противном случае сходимость будет только при x_0, достаточно близком к корню. Существует несколько достаточных условий сходимости. Если производные $f'(x)$ и $f''(x)$ сохраняют знак в окрестности корня, рекомендуется выбирать x_0 так, чтобы $f(x_0)f''(x_0) > 0$. Если, кроме этого, для отрезка $[a,b]$, содержащего корень, выполняются условия

$$\left|\frac{f(a)}{f'(a)}\right| < b-a, \quad \left|\frac{f(b)}{f'(b)}\right| < b-a,$$

то метод сходится для любых $a \le x_0 \le b$.

Метод бисекции:

На [a,b] берется c=(a+b)/2

1.Если |f(c)| < ε, то *c* – корень. Здесь ε - заданная точность.
2. Если f(c)f(a)<0, то корень лежит в интервале [a,c].
3. Если f(c)f(b)<0, то корень лежит на отрезке[c,b].

Процесс повторяется до нахождения корня с заданной точностью.

Програмная реализация:

```
using System;
```

```
using System.Collections.Generic;

namespace lab3
{
                        class Program
                        {
                                                static  decimal epsilon = 0.0001M;
                                                static void Main(string[] args)
                                                {
            epsilon /= 2;
            Polynom _pol;
            Polynom p1 = new Polynom(new decimal[] { -10, 1 });
            Polynom p2 = new Polynom(new decimal[] { 10, 1 });
            Polynom p3 = new Polynom(new decimal[] {0, 1});

            List<(Polynom pol, string description)> list = new List<(Polynom pol, string
description)>();
                                                                                _pol = new
Polynom(new decimal[] { 6, -5, -2, 1 });
            list.Add((_pol, "(x-1)(x-2)(x-3)-- простой пример с известными корнями"));
            _pol = new Polynom(new decimal[] { 0, 0, 0, 1 });
            list.Add((_pol,"x3 -- один корень в точке перегиба"));
            _pol = p1 * p2 * p3;
            list.Add((_pol, "(x-10)(x+10)x -- корни на краях интервала и в точке перегиба"));
            _pol = new Polynom(new decimal[] { 16, -12, 0, 1 });
            list.Add((_pol, "x3-12x+16-- корень в точке 2, где находится минимум"));
            _pol = new Polynom(new decimal[] { -16, -12, 0, 1 });
            list.Add((_pol, "x3-12x-16 -- корень в точке -2, где находится максимум"));
            _pol = new Polynom(new decimal[] { -20.6282M, 39.8645M, -13.3667M, 1 });
            list.Add((_pol, "пример из задания"));

            foreach ((Polynom pol, string description) pair in list)
            {
                Console.WriteLine(pair.pol + "    " + pair.description);
                Polynom pol = pair.pol;
                Polynom[] row = ShturmRow(pol);
                foreach (Polynom __pol in row)
```

```
            {
                Console.WriteLine(__pol);
            }

            Console.WriteLine(N(row, -10) - N(row, 10));


            decimal[] points = new decimal[] { -10 };
            var roots = RootsOfDiff(pol.Diff());
            Array.Resize(ref points, 2 + roots.Length);
            roots.CopyTo(points, 1);
            points[^1] = 10;

            for (int i = 1; i < points.Length; ++i)
            {
                if (pol.Count(points[i]) * pol.Count(points[i - 1]) <= 0)
                {
                    Console.WriteLine("[" + points[i - 1] + " , " + points[i] + "]");
                    Console.WriteLine("Метод хорд:");
                    Horde(points[i - 1], points[i], pol);
                    Console.WriteLine("Метод бисекции:");
                    Bisection(points[i - 1], points[i], pol);
                    Console.WriteLine("Метод Ньютона:");
                    Newton(points[i - 1], points[i], pol);
                    Console.WriteLine("\n");
                }
            }
            Console.WriteLine("\n\n\n");
        }
                                                }

                                                static void Horde(decimal a, decimal b,
Polynom f)
                                                {

        if (f.Count(a) == 0)
```

```
            {
                Console.WriteLine(a + "  f(x)=" + f.Count(a));
                Console.WriteLine("Количество итераций: " + 0);
                return;
            }

            if (f.Count(b) == 0)
            {
                Console.WriteLine(b + "  f(x)=" + f.Count(b));
                Console.WriteLine("Количество итераций: " + 0);
                return;
            }

            if (f.Count(b) * f.Diff().Diff().Count(b) > 0)
                                                                                {

              int count = 0;

              decimal x = a;

              decimal prev = 0;

              decimal fb = f.Count(b);

              while(Math.Abs(x - prev) >= epsilon)

              {

                                        var fx = f.Count(x);

                                        prev = x;

                                        x = x - fx / (fb - fx) * (b - x);

                                        ++count;

              }

                                                                Console.WriteLine(x + "  f(x)="
+ f.Count(x));

              Console.WriteLine("Количество итераций: " + count);
```

```
                                                                    }
                                                                    else
                                                                    {
            int count = 0;
            decimal x = b;
            decimal prev = 0;
            decimal fa = f.Count(a);
            while (Math.Abs(prev - x) >= epsilon)
            {
                                    var fx = f.Count(x);
                                    prev = x;
                                    x = x - fx / (fa - fx) * (a - x);
                                    ++count;
            }

            Console.WriteLine(x + "  f(x)=" + f.Count(x));
            Console.WriteLine("Количество итераций: " + count);
                                                                    }
                                                }

                                                static void Bisection(decimal a, decimal b,
Polynom f)
                                                {
        if (f.Count(a) == 0)
        {
            Console.WriteLine(a + "  f(x)=" + f.Count(a));
            Console.WriteLine("Количество итераций: " + 0);
            return;
        }
```

```
            if (f.Count(b) == 0)
            {
                Console.WriteLine(b + "  f(x)=" + f.Count(b));
                Console.WriteLine("Количество итераций: " + 0);
                return;
            }

            decimal a1 = a;
                                                                    decimal b1 = b;
                                                                    int count = 0;
                                                                    while(b-a > epsilon)
                                                                    {
              decimal mid = (a + b) / 2;
                 decimal fmid = f.Count(mid);
                 if (fmid == 0) break;
                 if (fmid * f.Count(b) < 0) a = mid;
                 else b = mid;

              ++count;
                                                                    }

       Console.WriteLine((a+b)/2 + "  f(x)=" + f.Count((a+b)/2));
       Console.WriteLine("Количество итераций: " + count);
                                                    }

                                                    static void Newton(decimal a, decimal b, Polynom f)
                                                    {
                                                                    //сходимость
                                                                    Polynom df = f.Diff();
            decimal[] roots = RootsOfDiff(f.Diff());
            decimal sp = -df[1] / (2 * df[2]);
```

```
            if (a < sp && b > sp)
            {
                if (f.Count(a) * f.Count(sp) < 0)
                {
                    b = sp;
                }
                else
                {
                    a = sp;
                }
            }

            decimal midPoint = (a + b) / 2;
            decimal x;
            if (df.Count(midPoint) * df.Diff().Count(midPoint) < 0) x = a;
            else x = b;

            int count = 0;
            decimal prev = 0;

            do
            {
                ++count;
                prev = x;
                x = x - f.Count(x) / df.Count(x);
            }
            while (Math.Abs(x - prev) >= epsilon);

            Console.WriteLine(x + "  f(x)=" + f.Count(x));
            Console.WriteLine("Количество итераций: " + count);
        }

                                                    static Polynom[] ShturmRow(Polynom func)
                                                    {
                                                                                Polynom[] row =
new Polynom[2];
```

```
                                                                row[0] = new
Polynom(func);
                                                                row[1] = new
Polynom(func.Diff());

        while(row[^1].pow != 0)
                                                                {
            int i = row.Length;
            var tmp = new Polynom[i + 1];
            row.CopyTo(tmp, 0);
            row = tmp;
            row[i] = -row[i - 2] % row[i - 1];
                                                                }
                                                                return row;
                                                        }
                                                        static int N(Polynom[] row, decimal a)
                                                        {
                                                                int count = 0;
                                                                bool skip = true;
                                                                bool isPos = true;
                                                                foreach(Polynom f
in row)
                                                                {
            if (skip)
            {
                isPos = f.Count(a) > 0;
                                        skip = false;
                                        continue;
```

```
            }

            bool tmp = f.Count(a) > 0;
            if (tmp != isPos) ++count;

            isPos = tmp;

* pol[1] - 4 * pol[0] * pol[2];
            if (D < 0) return new decimal[0];
new decimal[1];
Sqrt(D)-pol[1])/(2 * pol[2]);
(Sqrt(D) - pol[1]) / (2 * pol[2]);
roots[0])

            Array.Resize(ref roots, 2);
            roots[1] = tmp;

epsilon = 0.0M)
                                    }

                                    return count;
}

static decimal[] RootsOfDiff(Polynom pol)
{
                                    decimal D = pol[1]

                                    decimal[] roots =
                                    roots[0] = (-
                                    decimal tmp =
                                    if(tmp !=
                                    {

                                    }

                                    return roots;
}

static decimal Sqrt(decimal x, decimal
```

```
                                                {
                                                                if (x < 0) throw
new OverflowException("Cannot calculate square root from a negative number");

                                                                decimal current =
(decimal)Math.Sqrt((double)x), previous;
                                                                do
                                                                {

            previous = current;

            if (previous == 0.0M) return 0;

            current = (previous + x / previous) / 2;
                                                                }
                                                                while
(Math.Abs(previous - current) > epsilon);
                                                                return current;
                                                }

                        }
}
```

Результаты расчетов программы:

```
 x3-2x2-5x+6   (x-1)(x-2)(x-3)-- простой пример с известными корнями
 x3-2x2-5x+6
 3x2-4x-5
 +4,22222222222222222222222227x-4,88888888888888888888888885
 +5,609418282548476454293628809 4
3
[-10 , -0,78629964784689118407899399 47]
Метод хорд:
-1,999589086242484448882324105 2  f(x)=0,006162355631187178041974670
Количество итераций: 79
Метод бисекции:
-1,999994768507196501297683794 5  f(x)=0,000078472173104488088472868
Количество итераций: 18
Метод Ньютона:
-2,000000000000082729770476111 4  f(x)=-0,000000000001240946557141726
Количество итераций: 8
```

```
 x3   x3 -- один корень в точке перегиба
 x3
 3x2

1
[-10 , 0]
Метод хорд:
0  f(x)=0
Количество итераций: 0
Метод бисекции:
0  f(x)=0
Количество итераций: 0
Метод Ньютона:
-0,000078226425762698449436798 3  f(x)=-0,000000000000478696731487797 0
Количество итераций: 29
```

```
 x3-100x   (x-10)(x+10)x -- корни на краях интервала и в точке перегиба
 x3-100x
 3x2-100
 +66,666666666666666666666666670x
 +100
2
[-10 , -5,773502691896257645091487805]
Метод хорд:
-10  f(x)=0
Количество итераций: 0
Метод бисекции:
-10  f(x)=0
Количество итераций: 0
Метод Ньютона:
-10  f(x)=0
Количество итераций: 1


[-5,773502691896257645091487805 , 5,773502691896257645091487805]
Метод хорд:
0,000000000000000000000000000  f(x)=0,000000000000000000000000000
Количество итераций: 2
Метод бисекции:
0,000000000000000000000000000  f(x)=0,000000000000000000000000000
Количество итераций: 0
Метод Ньютона:
0  f(x)=0
Количество итераций: 1


[5,773502691896257645091487805 , 10]
Метод хорд:
10  f(x)=0
Количество итераций: 0
Метод бисекции:
10  f(x)=0
Количество итераций: 0
Метод Ньютона:
10  f(x)=0
Количество итераций: 1
```

```
 x3-12x+16   x3-12x+16-- корень в точке 2, где находится минимум, и в точке -4
 x3-12x+16
 3x2-12
 +8,000000000000000000000000000x-16

1
[-10 , -2]
Метод хорд:
-3,999867775514265576946404960 2  f(x)=0,004759871688975419423259217
Количество итераций: 36
Метод бисекции:
-4  f(x)=0
Количество итераций: 1
Метод Ньютона:
-4,000000000000000930955550100  f(x)=-0,000000000000003351439980360
Количество итераций: 7


[-2 , 2]
Метод хорд:
2  f(x)=0
Количество итераций: 0
Метод бисекции:
2  f(x)=0
Количество итераций: 0
Метод Ньютона:
1,999964028087612021740271997 1  f(x)=0,000000007763824338209757698
Количество итераций: 15


[2 , 10]
Метод хорд:
2  f(x)=0
Количество итераций: 0
Метод бисекции:
2  f(x)=0
Количество итераций: 0
Метод Ньютона:
2,000031587377864258098061868 3  f(x)=0,000000005986606158735756672
```

```
 x3-12x-16   x3-12x-16 -- корень в точке -2, где находится максимум, и в точке 4
 x3-12x-16
 3x2-12
 +8,00000000000000000000000000x+16

1
[-10 , -2]
Метод хорд:
-2  f(x)=0
Количество итераций: 0
Метод бисекции:
-2  f(x)=0
Количество итераций: 0
Метод Ньютона:
-2,00003158737786425809806186 83  f(x)=-0,000000005986606158735756672
Количество итераций: 19


[-2 , 2]
Метод хорд:
-2  f(x)=0
Количество итераций: 0
Метод бисекции:
-2  f(x)=0
Количество итераций: 0
Метод Ньютона:
-1,99996402808761202174027199 71  f(x)=-0,000000007763824338209757698
Количество итераций: 15


[2 , 10]
Метод хорд:
3,99986777551426557694640496 02  f(x)=-0,00475987168897541942325921 7
Количество итераций: 36
Метод бисекции:
4  f(x)=0
Количество итераций: 1
Метод Ньютона:
4,00000000000000009309555501 00  f(x)=0,00000000000000335143998036 0
Количество итераций: 7
```

```
 x3-13,3667x2+39,8645x-20,6282   пример из задания
 x3-13,3667x2+39,8645x-20,6282
 3x2-26,7334x+39,8645
 +13,12781530888888888888888898x-38,57811246111111111111111127
 +12,78865465630493839506015208 1
3
[-10 , 1,893554243893419369771528 2983]
Метод хорд:
0,654213494006105799574697156 0  f(x)=0,010809410086894381354704451 0
Количество итераций: 90
Метод бисекции:
0,653742499028942119069388756 4  f(x)=-0,000336453537784836552653992 5
Количество итераций: 18
Метод Ньютона:
0,653756713537697316362401060 0  f(x)=-0,000000000153627108600269791 2
Количество итераций: 8


[1,893554243893419369771528 2983 , 7,017579089439913963561805035]
Метод хорд:
3,381361173759303564351030882 8  f(x)=-0,000331316783459858977148827
Количество итераций: 38
Метод бисекции:
3,381343974418517926314294540 4  f(x)=-0,000052172940786340454536019
Количество итераций: 17
Метод Ньютона:
3,381340759789757106257762954 7  f(x)=-0,000000000000037681113329111
Количество итераций: 4


[7,017579089439913963561805035 , 10]
Метод хорд:
9,331600066507018920111413866  f(x)=-0,000127031340724728401151 1
Количество итераций: 8
Метод бисекции:
9,331599317568187505185150322  f(x)=-0,000165703075313543516195 0
Количество итераций: 16
Метод Ньютона:
9,331602526667140104625715930  f(x)=0,000000000055905707367216 3
Количество итераций: 4
```

Оценка:

Абсолютные погрешности приближенных решений для тестовых примеров:

Уравнение	Первый корень	Абс. Погрешность м. хорд	Абс. Погрешность м. бисекции	Абс. Погрешность метода Ньютона
x3-2x2-5x+6	-2			
x3	0	0	0	0
(x-10)(x+10)x	-10	0	0	0
x3-12x+16	-4		0	0,0000000000000000930955550100
x3-12x-16	-2	0	0	0

Относительная погрешность приближенного числа:

Учитывая формулу

$$\left|\frac{a-a^*}{a}\right| = \frac{\beta_{l+1}r^{-(l+1)} + \beta_{l+1}r^{-(l+2)} + \dots}{\beta_1 r^{-1} + \beta_2 r^{-2} + \dots} \le \frac{r^{-1}}{\beta_1 r^{-1}} \le r^{1-l}$$

где r-основание системы исчисления

И параметры используемого типа decimal

Ниже в таблице даны параметры стандартных форматов чисел с плавающей запятой. Здесь: **w** — ширина битового поля для представления порядка, **t** — ширина битового поля для представления мантиссы, **k** — полная ширина битовой строки.

Параметр	*Binary32*	*Binary64*	*Binary128*	*Decimal64*	*Decimal128*
Параметры формата					
b	2	2	2	10	10
p, цифры	24	53	113	16	34
emax	127	1023	16383	384	6144
Параметры кодирования					
BIAS	127	1023	16383	398	6176
w, биты	8	11	15	13	17
t, биты	23	52	112	50	110
k, биты	32	64	128	64	128

То относительная погрешность приближенного числа не превышает 2^(1 – 110) = 1,50463*10^(-36)

Вывод:

В ходе лабораторной работы были изучены методы хорд, бисекции и метод Ньютона поиска корней в нелинейном уравнении. Точность используемого типа данных сделал погрешности вычислений пренебрежимо малыми. Наилучшую абсолютную погрешность во всех случаях демонстрировал метод Ньютона, имея наименьшее количество итераций (для неграничных случаев, где корень не находиться на крае исследуемого интервала). Методы бисекции и хорд продемонстрировали свою зависимость от начального интервала.

СПИСОК ЛИТЕРАТУРЫ

1. ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ АЛГЕБРЫ: БАЗОВЫЕ ПОНЯТИЯ И АЛГОРИТМЫ / Б.В, Фалейчик, 2010
2. КОМПЕНСАЦИЯ ПОГРЕШНОСТЕЙ ПРИ ОПЕРАЦИЯХ С ЧИСЛАМИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ[Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/266023/