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 №3
на тему:

«Численное решение систем нелинейных уравнений»

Выполнил: студент группы 353502
Згирская Дарья Денисовна
Проверил: Анисимов Владимир Яковлевич

Минск 2024

ЦЕЛИ ВЫПОЛНЕНИЯ ЗАДАНИЯ

- изучение методов численного решения нелинейных уравнений – методов хорд, Ньютона и половинного деления;
- исследование скорости сходимости итерационных процедур;
- сравнение числа итераций, необходимого для достижения заданной точности вычисления разными методами.

ЗАДАНИЕ

Вариант 5.

1) Используя теорему Штурма определить число корней уравнения: x3 +ax2 +bx + c=0 на отрезке [-10,10]. Значения коэффициентов уравнения взять из таблицы (рис. 1).

2) Отделить все корни, лежащие на данном отрезке.

3) Вычислить наименьший из корней сначала методом половинного деления, а за затем методом хорд и методом Ньютона. Сравнить число необходимых итераций в обоих методах. Точность до 0.0001.

a	b	c	№ вариан-та
-14,4621	60,6959	-70,9238	1
-10,2374	-91,2105	492,560	2
-19,7997	28,9378	562,833	3
-5,5796	-193,022	-633,105	4
9,57496	-243,672	773,65	5
20,2374	-131,210	-843,923	6
38,4621	364,594	914,196	7
-13,3667	39,8645	-20,6282	8
2,65804	-28,0640	21,9032	9
-6,4951	-31,2543	23,1782	10
9,9296	17,8390	-24,4532	11
6,0951	-35,3942	-25,7283	12

Рисунок 1

ХОД РАБОТЫ

Перед началом выполнения работы были изучены теоретические материалы на тему «Численное решение систем нелинейных уравнений».

Далее, чтобы при помощи теоремы Штурма определить количество корней данного уравнения, была создана процедура, строящая ряд Штурмана (рис. 2).

```
> sturm_series := proc (expr, deriv)
  local L, remainder, remainder2, quotient, i, p1, p2, t;

  L := [ ];
  L := [op(L), expr];
  L := [op(L), deriv];
  quo(expr, deriv, x, remainder);
  remainder := -remainder;
  L := [op(L), remainder];

  if (subs(x = 1, remainder) = remainder) then return L end if;

  for i from 3 by 1 to infinity do
    p2 := L[i];
    p1 := L[i - 1];
   remainder2 := -(rem(p1, p2, x));
   L := [op(L), remainder2];
   if (subs(x = 1, remainder2) = remainder2) then return L end if;
  end do;
  end proc:
```

Рисунок 2

Также была создана процедура, продсчитывающая количество смен знака в полученном ряде штурмана (рис. 3).

```
> sign_changes_count := proc (L, a)
  local i, a1, a2, count;
  count := 0;
  a1 := 0;

  for i from 1 by 1 to nops(L) do
   a2 := a1 :
   a1 := subs(x = a, L[i]) :
    if ((a2 < 0 and a1 > 0) or (a2 > 0 and a1 < 0)) then count := count + 1
    end if;
  end do;

  count;
  end proc:
```

Рисунок 3

И процедура, которая находит интервал, в котором находится корень, с помощью предыдущей процедуры. Она возвращает список со значениями x, каждый из которых является концом интервала для некоторого корня.

```
> sturm_isolation := proc (L,Lret, a, b)
    local i, c, count, a1, b1, endCount, test1, test2, Lret1;

    Lret1 := Lret;
    a1 := a;
    b1 := b;
    count := sign_changes_count(L, a1) - sign_changes_count(L, b1);

    for i from 1 by 1 to infinity do
      if count = 1 then
        Lret1 := [op(Lret1), a1];
        Lret1 := [op(Lret1), b1];
        return Lret1;
      end if;

      if count = 0 then
        return Lret1;
      end if;
      c := (a + b)/2;
      test1 := sign_changes_count(L, a1) - sign_changes_count(L, c);
      test2 := sign_changes_count(L, c) - sign_changes_count(L, b1);

      return concat(concat(Lret1, sturm_isolation(L, Lret1, a, c)), concat(Lret1,
        sturm_isolation(L, Lret1, c, b)));
    end do;
    end proc:
```

Рисунок 4

Таким образом, было выполнено отделение всех корней, лежащих на данном отрезке.

Теперь необходимо вычислить наименьший из корней методом половинного деления, методом хорд и методом Ньютона.

Для этого были соответствующие процедуры (рис. 5-7).

```
> bin_search := proc(expr, a, b, epsilon)
    local i, res1, res2, c, a1, b1;
    a1 := a;
    b1 := b;

    while abs(b1 - a1) ≥ epsilon do
      c := (a1 + b1)/2;
      res1 := subs(x = a1, expr);
      res2 := subs(x = c, expr);

      if res1 * res2 < 0 then
        b1 := c;
      else
        a1 := c;
      end if;
    end do;

    return (a1 + b1)/2;
  end proc:
```

Рисунок 5

```
> newthon_method := proc(expr, deriv, x0, epsilon)
    local x1, x2, i;

    x1 := x0;
    for i from 1 by 1 to infinity do

      if i = 10000 then
        print("Не сошлось за 10000 итераций");
        return;
      end if;
      x2 := x1 - subs(x = x1, expr)/subs(x = x1, deriv);

      if |x2 - x1| < epsilon then
        print(i);
        return x2;
      end if;

      x1 := x2;
    end do;
    end proc:
```

Рисунок 7

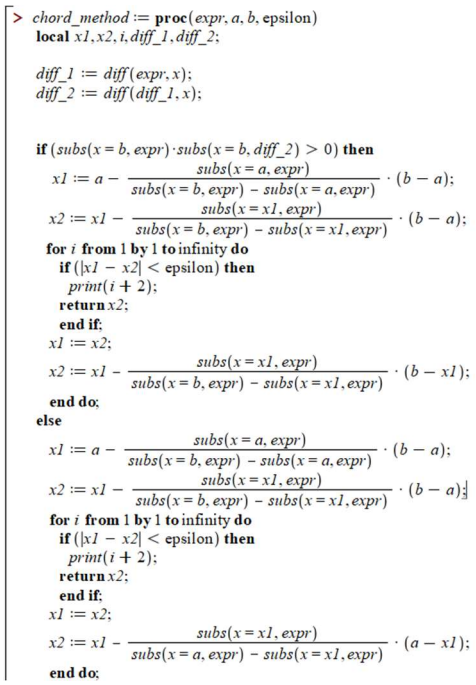
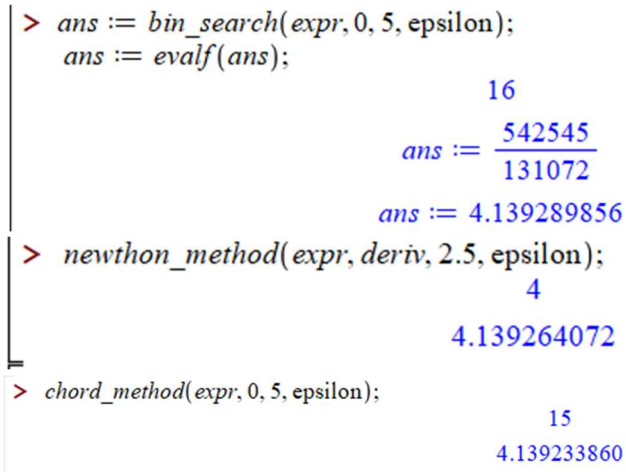


Рисунок 6

Реализованные процедуры ищут корень с заданной точностью epsilon.

Теперь сравним число необходимых итераций для использованных методов:



В данном случае самый быстрый метод – метод Ньютона, а самый медленный – метод половинного деления.

ТЕСТОВЫЕ ПРИМЕРЫ

```
#Тестовый пример 1. Метод Ньютона не сходится
>
> f := x^(1/3);
```

$$f := x^{1/3}$$

```
> deriv := diff(f, x);
```

$$deriv := \frac{1}{3 x^{2/3}}$$

```
> deriv_2 := diff(deriv, x);
```

$$deriv\_2 := -\frac{2}{9 x^{5/3}}$$

```
> f1 := abs(f·deriv_2);
```

$$f1 := \frac{2}{9 |x|^{4/3}}$$

```
> f2 := abs(deriv·deriv);
```

$$f2 := \frac{1}{9 |x|^{4/3}}$$

```
> #Т. к. f2 < f1 для любого x, то итерация сходится только при x0, достаточно близким к корню.
> newthon_method(f, deriv, 0.1, epsilon);
```

"Не сошлось за 10000 итераций"

```
> newthon_method(f, deriv, 2, epsilon);
```

"Не сошлось за 10000 итераций"

```
#Тестовый пример 2
>
> f := x^3 - 2·x + 2;
```

$$f := x^3 - 2 x + 2$$

```
> deriv := diff(f, x);
```

$$deriv := 3 x^2 - 2$$

```
> deriv_2 := diff(deriv, x);
```

$$deriv\_2 := 6 x$$

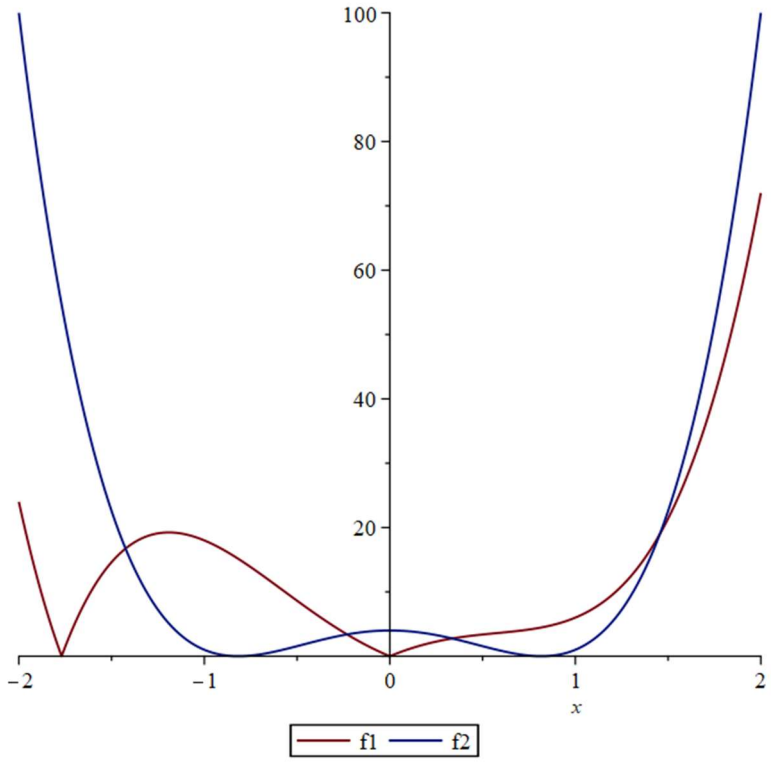
```
> f1 := abs(f·deriv_2);
```

$$f1 := 6 \left|\left(x^3 - 2 x + 2\right) x\right|$$

```
> f2 := abs(deriv·deriv);
```

$$f2 := \left|3 x^2 - 2\right|^2$$

```
> plot([f1, f2], x = -2 ..2, legend = ["f1", "f2"]);
```



```
> #Т. к. f2 < f1 для любого x, то итерация сходится только при x0, достаточно близким к корню.
> newthon_method(f, deriv, -0.5, epsilon);
```

"Не сошлось за 10000 итераций"

```
> ans := newthon_method(f, deriv, 2, epsilon) :
  ans := evalf(ans);
  eq := subs(x = ans, f);
```

$$8$$

$$ans := -1.769292354$$

$$eq := 2. \times 10^{-9}$$

ВЫВОДЫ

В ходе выполнения данной работы были изученыметоды решения нелинейных уравнений.

Было выполнено задание и решены 2 тестовых примера, показывающих функции, для которых не сходится метод Ньютона частично/полностью.

Также было произведено сравнение количества итераций для трех изученных методов.