

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №3

на тему:

«ЧИСЛЕННОЕ РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ»

БГУИР 1-40 04 01

Выполнил студент группы 253505
БЕКАРЕВ Станислав Сергеевич

(дата, подпись студента)

Проверил доцент кафедры
информатики
АНИСИМОВ Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2023

Содержание

1. Цель работы
2. Задание
3. Программная реализация
4. Полученные результаты
5. Оценка полученных результатов
6. Вывод

Цель работы

- изучить метод половинного деления, метод хорд и метод Ньютона численного решения нелинейных уравнений;
- составить программу решения нелинейных уравнений указанными методами, применимую для организации вычислений на ЭВМ;
- выполнить тестовые примеры и проверить правильность работы программы

ЗАДАНИЕ.

1) Используя теорему Штурма определить число корней уравнения:

$x^3 + ax^2 + bx + c = 0$ на отрезке $[-10, 10]$. Значения коэффициентов уравнения взять из таблицы.

2) Отделить все корни, лежащие на данном отрезке.

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

Исходные данные:

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

Вариант 3

Программная реализация

Для проверки решения подставим найденный корень в функцию и найдем ее значение.

Исходные данные

Функция $f(x)$, полученная в результате подстановки в $x^3 + a*x^2 + b*x + c$:

$$f(x) = x^3 - 19.7997x^2 + 28.9378x + 562.833$$

Примечание: $x**n$ – возведение x в степень n .

Код половинного деления:

```
def binary_method(left, right):
    count = 1
    while True:
        middle = (left + right) / 2
        if right - left < 10 ** (-4):
            print(f"Кол-во итераций: {count}")
            return middle
        elif f.subs(x, left) * f.subs(x, middle) <= 0:
            right = middle
        elif f.subs(x, middle) * f.subs(x, right) <= 0:
            left = middle
        count += 1
```

Код метода Ньютона:

```
def newton_method(left, right):
    if f.subs(x, left)/diff(f).subs(x, left) - left < right -
f.subs(x, right)/diff(f).subs(x, right):
        curr = left
    else:
        curr = right
    count = 1
    while True:
        next = curr - (f.subs(x, curr) / diff(f).subs(x, curr))
        if abs(curr - next) < 10 ** (-4):
            print(f"Кол-во итераций: {count}")
            return (next + curr) / 2
        curr = next
        count += 1
```

Код метода хорд:

```
def chord_method(left, right):
    count = 1
    if f.subs(x, left) * diff(diff(f)).subs(x, left) < 0:
        curr = right
        while True:
            next = curr - (f.subs(x, curr) * (left - curr)) /
(f.subs(x, left) - f.subs(x, curr))
            if(abs(curr - next) < 10 ** (-4)):
                print(f"Кол-во итераций: {count}")
                return (next + curr) / 2
            curr = next
            count += 1
    elif f.subs(x, right) * diff(diff(f)).subs(x, right) < 0:
        curr = left
        while True:
            next = curr - (f.subs(x, curr) * (right - curr)) /
(f.subs(x, right) - f.subs(x, curr))
            if (abs(next - curr) < 10 ** (-4)):
                print(f"Кол-во итераций: {count}")
                return (next + curr) / 2
            curr = next
            count += 1
```

Программная реализация Теоремы Штурмы:

```
def get_sign_changes(left, right):
    Nleft = 0
    Nright = 0
    for i in range(0, len(f_arr) - 1):
        if f_arr[i].subs(x, left)*f_arr[i+1].subs(x, left) < 0:
            Nleft += 1
    for i in range(0, len(f_arr) - 1):
        if f_arr[i].subs(x, right)*f_arr[i+1].subs(x, right) < 0:
            Nright += 1
    return Nleft - Nright

def search_root_range(left, right):
    while True:
        amount = get_sign_changes(left, right)
        if amount >= 2:
            right = right - (right - left) / 2
        elif amount == 0:
            left = right
            right = right + (right - left) / 2
        elif amount == 1:
            while right - left > 2:
                middle = (right + left) / 2
                if get_sign_changes(left, middle) >
get_sign_changes(middle, right):
                    right = middle
                else:
                    left = middle
            roots_bound.append((left, right))
            break

def search_allroots_range(left, right):
    curr_left = left
    for i in range(0, roots_amount):
        search_root_range(curr_left, right)
        curr_left = roots_bound[i][1]

def shturm_theory():
    global roots_amount
    for i in range(0, degree(f) - 1):
        f_arr.append(-1 * div(f_arr[i], f_arr[i + 1])[1])
    roots_amount = get_sign_changes(beg_left, beg_right)
    print(f"Кол-во корней: {roots_amount}")
    search_allroots_range(beg_left, beg_right)
    print("Границы корней уравнения:")
    print(roots_bound)
```

Полученные результаты

Исходная функция:

$$f(x) = x^3 - 19.7997x^2 + 28.9378x + 562.833$$

Кол-во корней: 2

Границы корней уравнения:

$[(-5.0, -3.75), (8.28125, 10)]$

Метод половинного деления:

Кол-во итераций: 15

-4.271583557128906

После подставления найденного корня имеем:

0.00787313143160873

Метод хорд:

Кол-во итераций: 5

-4.27162267722469

После подставления найденного корня имеем:

-0.00201762194728872

Метод Ньютона:

Кол-во итераций: 4

-4.27161469959348

После подставления найденного корня имеем:

-6.25356506134267e-7

Результаты вычислений:

	<i>Метод половинного деления</i>	<i>Метод хорд</i>	<i>Метод Ньютона</i>
Корень найденный с точностью 10^{-4}	-4.27158355	-4.27162268	-4.27161470
Значение функции	0.00787313	-0.00201762	-0.00000063
Кол-во итераций	15	5	4

Тестовый пример 1.

С помощью метода random создадим многочлен со случайными коэффициентами (повысим точность вычисления корней каждого метода до 10^{-8}):

Исходная функция:

$$f(x) = x^5 - 10.739x^4 + 14.358x^3 - 79.833x^2 - 113.475x + 179.007$$

Кол-во корней на промежутке $[-10, 10]$: 2

Границы корней уравнения:

$[(-2.5, -1.25), (0.15625, 1.5625)]$

Метод половинного деления:

Кол-во итераций: 28

-1.634888886474073

После подставления найденного корня имеем:

-2.19023672798357e-6

Метод хорд:

Кол-во итераций: 19

-1.63488888308904

После подставления найденного корня имеем:

-5.44691175718981e-7

Метод Ньютона:

Кол-во итераций: 6

-1.63488888196856

После подставления найденного корня имеем:

-5.96855898038484e-13

Тестовый пример 2.

В данном примере мы видим многочлен, который имеет кратный корень.

```
Исходная функция:  
f(x) = 4*x**2 + 16*x + 16  
Кол-во корней на промежутке [-10, 10]: 1  
Границы корней уравнения:  
[(-2.5, -1.25)]  
Метод Ньютона:  
Кол-во итераций: 13  
-1.99986267089844  
После подставления найденного корня имеем:  
7.54371285438538e-8
```

В данном примере сработал только метод Ньютона, так как в оставшихся двух не были соблюдены условия сходимости.

Вывод

В ходе выполнения лабораторной работы я изучил метод половинного деления, метод хорд и метод Ньютона решения нелинейных уравнений, написал программу их реализации на языке Python, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

- Программа позволяет получить решения системы с заданной точностью (заданная точность в условиях лабораторной работы 10^{-4});
- Метод Ньютона эффективнее по сравнению с методами хорд и половинного деления, так как затрачивает меньшее число итераций;
- Оптимальным способом численного решения нелинейных уравнений является применение метода Ньютона, так скорость сходимости в этом методе почти всегда квадратичная.
- Метод Ньютона позволяет находить как простые, так и кратные корни. Основной его недостаток – малая область сходимости (x_0 должен быть достаточно близок к корню уравнения).