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) |

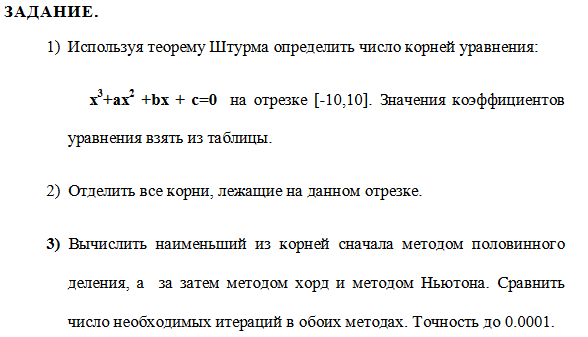
Минск 2023

**Содержание**

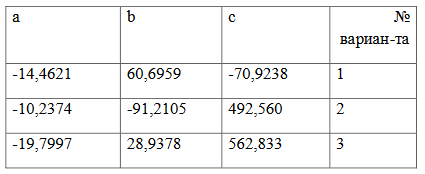
1. Цель работы
2. Задание
3. Программная реализация
4. Полученные результаты
5. Оценка полученных результатов
6. Вывод

**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

****

Исходные данные:



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:



*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

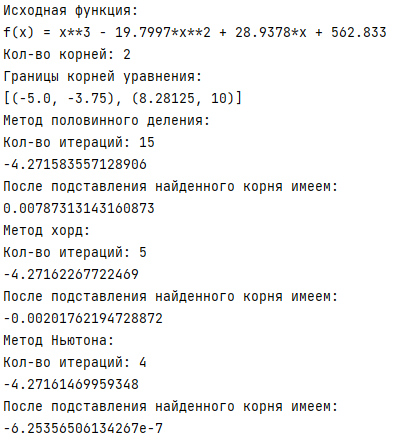
Программная реализация Теоремы Штурмы:

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**

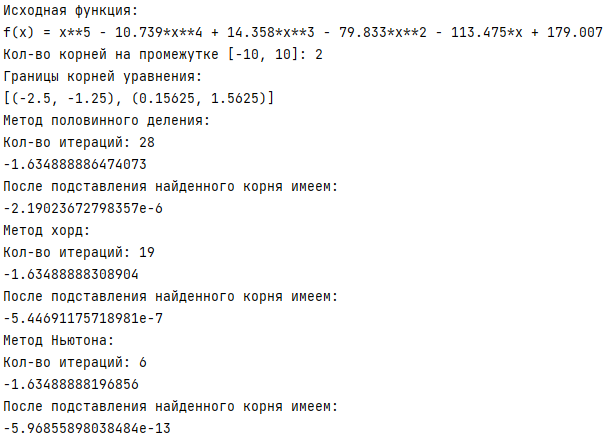


*Результаты вычислений:*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Метод половинного деления* | *Метод хорд* | *Метод Ньютона* |
| 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 |

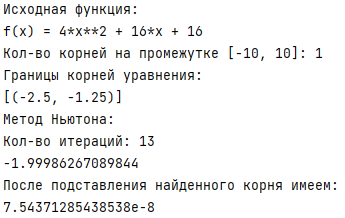
*Тестовый пример 1.*

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

**

*Тестовый пример 2.*

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



*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 (xо должен быть достаточно близок к корню уравнения).