Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №3
Численное решение нелинейных уравнений

Выполнил:
студент группы 953501
Кременевский В.С.

Руководитель:
доцент
Анисимов В.Я.

Минск 2021

Содержание

1. Цель работы

1) Изучить методы численного решения нелинейных уравнений (метод простой итерации, метод хорд, метод Ньютона)
2) Исследовать скорость сходимости итерационных процедур

3) Составить программу численного решения нелинейных уравнений методами бисекции, хорд, Ньютона
4) Проверить правильность работы программы на тестовых примерах
5) Численно решить нелинейное уравнение заданного варианта
6) Сравнить число итераций, необходимого для достижения заданной точности вычисления разными методами

2. Теоретический 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_t 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 $f(x)=0$.*

Таким образом, если при некотором k числа $f(x_k)$ и $f(x_{k+1})$ имеют разные знаки, то это означает, что на интервале (x_k,x_k+_1) уравнение имеет по крайней мере один действительный корень нечетной кратности (точнее – нечетное число корней). Выявить по таблице корень четной кратности очень сложно.

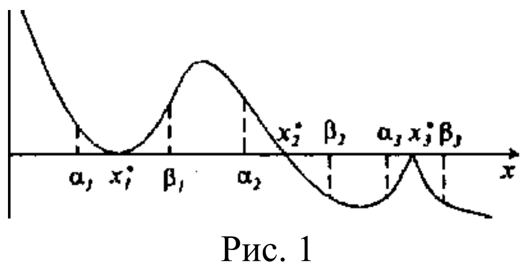


Рис. 1

На рис.1 представлены три наиболее часто встречающиеся ситуации:

а) кратный корень: $f'(x^*)=0, f(a_1)*f(b_1) > 0;$

б) простой корень: $f'(x^*)=0, f(a_2)*f(b_2) < 0;$

в) вырожденный корень: $f'(x^*)$ не существует, $f(a_3)*f(b_3)>0.$

Как видно из рис.1, в первых двух случаях значение корня совпадает с точкой экстремума функции и для нахождения таких корней рекомендуется использовать методы поиска минимума функции.

Для определения числа корней на заданном промежутке используется **Теорема Штурма**: Если $f(x)$ является многочленом и уравнение $f(x)=0$ не имеет кратных корней на промежутке [a, b], то число корней этого уравнения, лежащих на таком промежутке, совпадает с числом $N(a) - N(b)$, где функция N определяется следующим образом.

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

$f_0(x) = f(x),$

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

$f_i(x)$ = *остаток от деления* $f_{i-2}(x)$ *на* $f_{i-1}(x)$, *взятый с обратным знаком*

Функция $N(x)$ определяется как число перемен знака в ряде Штурма, если подставить в функции ряда значение x

Для отделения корней можно использовать график функции $y=f(x)$. Корнями уравнения являются те значения x, при которых график функции пересекает ось абсцисс. Построение графика функции даже с малой точностью обычно дает представление о расположении и характере корней уравнения (иногда позволяет выявить даже корни четной кратности). Если построение графика функции $y=f(x)$ вызывает затруднение, следует преобразовать исходное уравнение к виду $\varphi_1(x)=\varphi_2(x)$ таким образом, чтобы графики функций $y=\varphi_1(x)$ и $y=\varphi_2(x)$ были достаточно просты. Абсциссы точек пересечения этих графиков и будут корнями уравнения.

Допустим, что искомый корень уравнения отделен, т.е. найден отрезок *[a, b]*, на котором имеется только один корень уравнения. Для вычисления корня с требуемой точностью ε обычно применяют какую-либо итерационную процедуру ***уточнения корня***, строящую числовую последовательность значений x_n, сходящуюся к искомому корню уравнения. Начальное приближение x_0 выбирают на отрезке [a, b], продолжают вычисления, пока не выполнится неравенство $|\boldsymbol{x_{n-1} - x_n}| < \boldsymbol{\varepsilon}$, и считают, что x_n есть корень уравнения, найденный с заданной точностью. Имеется множество различных методов построения таких последовательностей и выбор алгоритма – весьма важный момент при практическом решении задачи. Немалую роль при этом играют такие свойства метода, как простота, надежность, экономичность, важнейшей характеристикой является его *скорость сходимости.* Последовательность x_n, сходящаяся к пределу x^*, имеет скорость сходимости порядка α, если при $n \to \infty$: $\left|x_{n+1} - x^*\right| = O(\left|x_n - x^*\right|^\alpha)$. При α=1 сходимость называется линейной, при 1<α<2 – сверхлинейной, при α=2 – квадратичной. С ростом α алгоритм, как правило, усложняется и условия сходимости становятся более жесткими. Рассмотрим наиболее распространенные итерационные методы уточнения корня.

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

$$x_k= \varphi(x_{k-1}),\ k=1,2,\ \ldots$$

Метод простых итераций не всегда обеспечивает сходимость к корню уравнения. Достаточным условием сходимости этого метода является выполнение неравенства $|\varphi'(\mathrm{x})| \le \mathrm{q} < 1$ на отрезке, содержащем корень и все приближения x_n. Метод имеет линейную скорость сходимости и справедливы следующие оценки:

$$\left|x_n - x^{**}\right| < \frac{q}{1-q}\left|x_n - x_{n-1}\right|,\ если\ \varphi'(x) > 0,$$

$$\left|x_n - x^{**}\right| < \left|x_n - x_{n-1}\right|,\ если\ \varphi'(x) < 0.$$

Метод имеет простую геометрическую интерпретацию: нахождение корня уравнения $f(x)=0$ равносильно обнаружению неподвижной точки функции $x= \varphi(x)$, т.е. точки пересечения графиков функций $y= \varphi(x)$ и $y=x$. Если производная $\varphi'(x)<0$, то последовательные приближения колеблются около корня, если же производная $\varphi'(x)>0$, то последовательные приближения сходятся к корню монотонно.

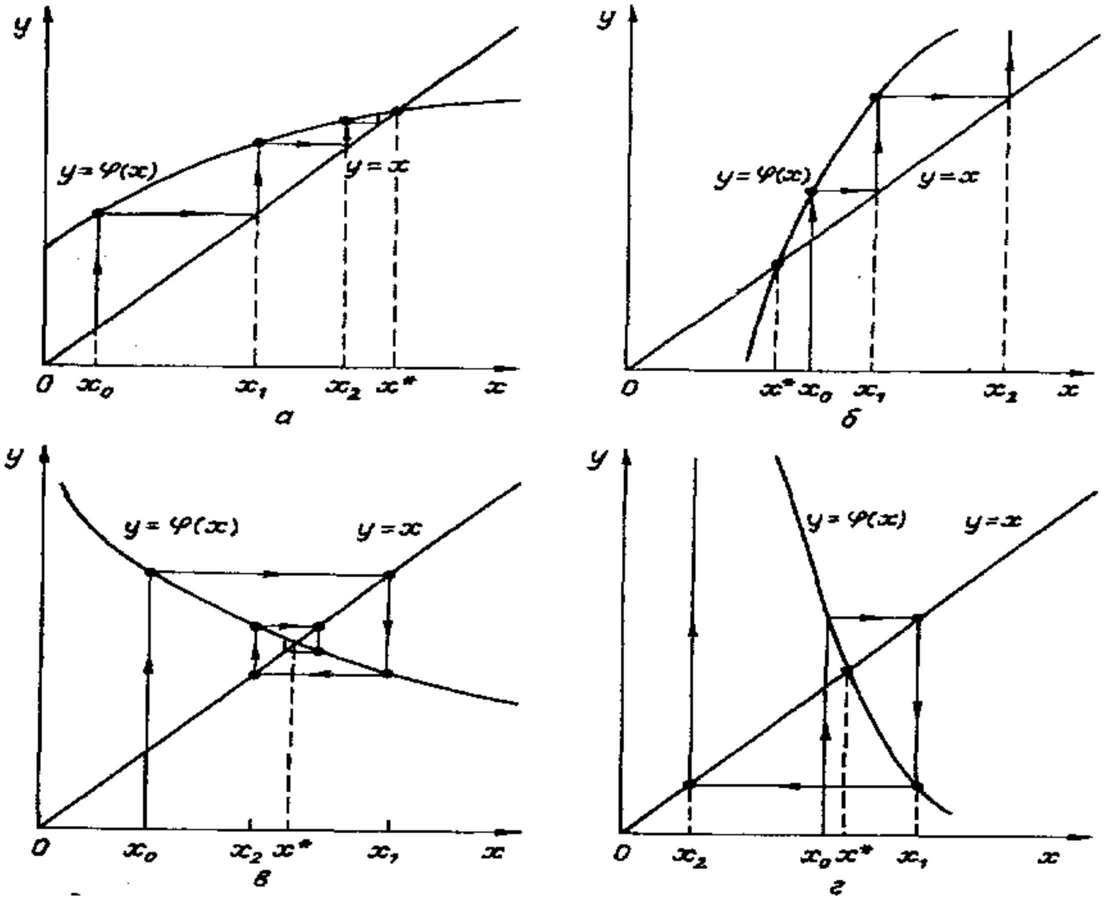


Рис. 2. Метод простых итераций: а - односторонний сходящийся процесс; б - односторонний расходящийся процесс; в - двухсторонний сходящийся процесс; г - двухсторонний расходящийся процесс

Рассмотрим процесс графически (рис. 2). Из графиков видно, что при $\varphi'(x)<0$ и при $\varphi'(x)>0$ возможны как сходящиеся, так и расходящиеся итерационные процессы. Скорость сходимости зависит от абсолютной величины производной $\varphi(x)$. Чем меньше $|\varphi'(x)|$ вблизи корня, тем быстрее сходится процесс.

Метод хорд. Пусть дано уравнение $f(x) = 0$, $a \le x \le b$, где $f(x)$ – дважды непрерывно дифференцируемая функция. Пусть выполняется условие $f(a)*f(b)<0$ и проведено отделение корней, то есть на данном интервале (a, b) находится один корень уравнения. При этом, не ограничивая общности, можно считать, что $f(b)>0$.

Пусть функция f выпукла на интервале (a, b) (см. рис. 3).

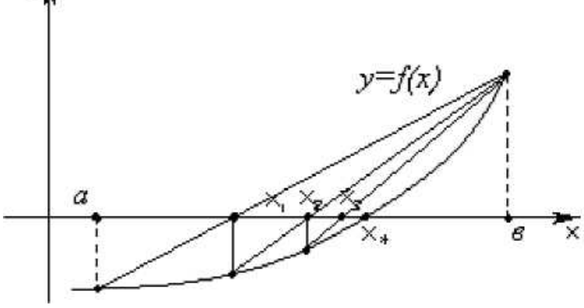


Рис. 3

Заменим график функции хордой (прямой), проходящей через точки *M₀(a, f(a))* и *M₁(b, f(b))*. Уравнение прямой, проходящей через две заданные точки, можно записать в виде $\dfrac{y-y_1}{y_2-y_1}=\dfrac{x-x_1}{x_2-x_1}$. В нашем случае получим: $\dfrac{y-f(a)}{f(b)-f(a)}=\dfrac{x-a}{b-a}$. Найдем точку пересечения хорды с осью Ox. Полагая $y = 0$, получаем из предыдущего уравнения: $x_1=a-\dfrac{f(a)}{f(b)-f(a)}*(b-a)$. Теперь возьмем интервал *(x_1,b)* в качестве исходного и повторим вышеописанную процедуру (см. рис. 3). Получим $x_2=x_1-\dfrac{f(x_1)}{f(b)-f(x_1)}*(b-x_1)$. Продолжим процесс. Каждое последующее приближение вычисляется по рекуррентной формуле

$$x_n=x_{n-1}-\frac{f(x_{n-1})}{f(b)-f(x_{n-1})}*(b-x_{n-1}),\ n=1,2,\ldots \tag{3.1}$$

Если же функция вогнута (см. рис. 4),

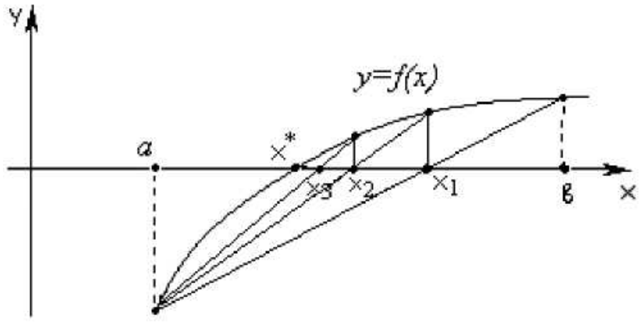


Рис. 4

уравнение прямой соединяющей точки *M_0(a, f(a))* и *M_1(b, f(b))* запишем в виде $\dfrac{y-f(b)}{f(a)-f(b)}=\dfrac{x-b}{a-b}$. Найдем точку пересечения хорды с осью Ox: $x_1=b-\dfrac{f(b)}{f(a)-f(b)}*(a-b)$. Теперь возьмем интервал *(a, x_1)* в качестве исходного и найдем точки пересечения хорды, соединяющей точки *(a, f(a))* и *(x_1, f(x_1))*, с осью абсцисс (см. рис. 4). Получим $x_2=x_1-\dfrac{f(x_1)}{f(a)-f(x_1)}*(a-x_1)$. Повторяя данную процедуру, получаем рекуррентную формулу:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f(a) - f(x_{n-1})} * (a - x_{n-1}),\ n = 1, 2, \ldots \tag{3.2}$$

Описанный выше метод построения рекуррентных последовательностей (3.1) и (3.2) называется методом хорд. Для использования метода хорд нужно было бы предварительно найти точки перегиба и выделить участки, на которых функция не меняет характер выпуклости. Однако на практике поступают проще: в случае $f(b)\,f''(b) > 0$ для построения рекуррентной последовательности применяются формулы (3.1), а в случае, когда $f(a)\,f''(a) > 0$, применяют формулы (3.2).

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

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

Метод имеет квадратичную скорость сходимости для простого корня, но очень чувствителен к выбору начального приближения. При произвольном начальном приближении итерации сходятся, если всюду $\left|f(x)\,f''(x)\right| < (f'(x))^2$, 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 $f(x)\,f''(x) > 0$. Если, кроме этого, для отрезка *[a,b]*, содержащего корень, выполняются условия $\left|\frac{f(a)}{f'(a)}\right| < b - a$, $\left|\frac{f(b)}{f'(b)}\right| < b - a$, то метод сходится для любых $a \leq x_0 \leq b$.

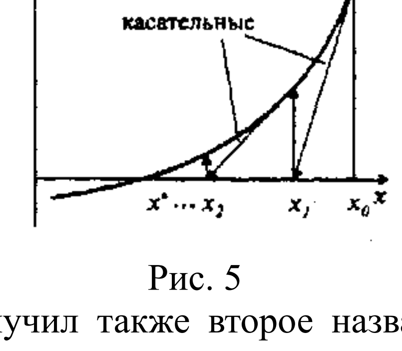


Рис. 5

Метод Ньютона получил также второе название *метод касательных* благодаря геометрической иллюстрации его сходимости, представленной на рис. 5.

Метод Ньютона позволяет находить как простые, так и кратные корни.

Основной его недостаток – малая область сходимости и необходимость вычисления производной.

Также хорошей практикой считается построение графиков, ведь имея перед глазами график мы сразу сможем найти приблизителье промежутки, где находится корень или даже возможность сразу найти корень. Ну и это просто очень сильный инсрумент для получения подробной информации о поведении функции на определенном промежутке.

3. Тестовые примеры

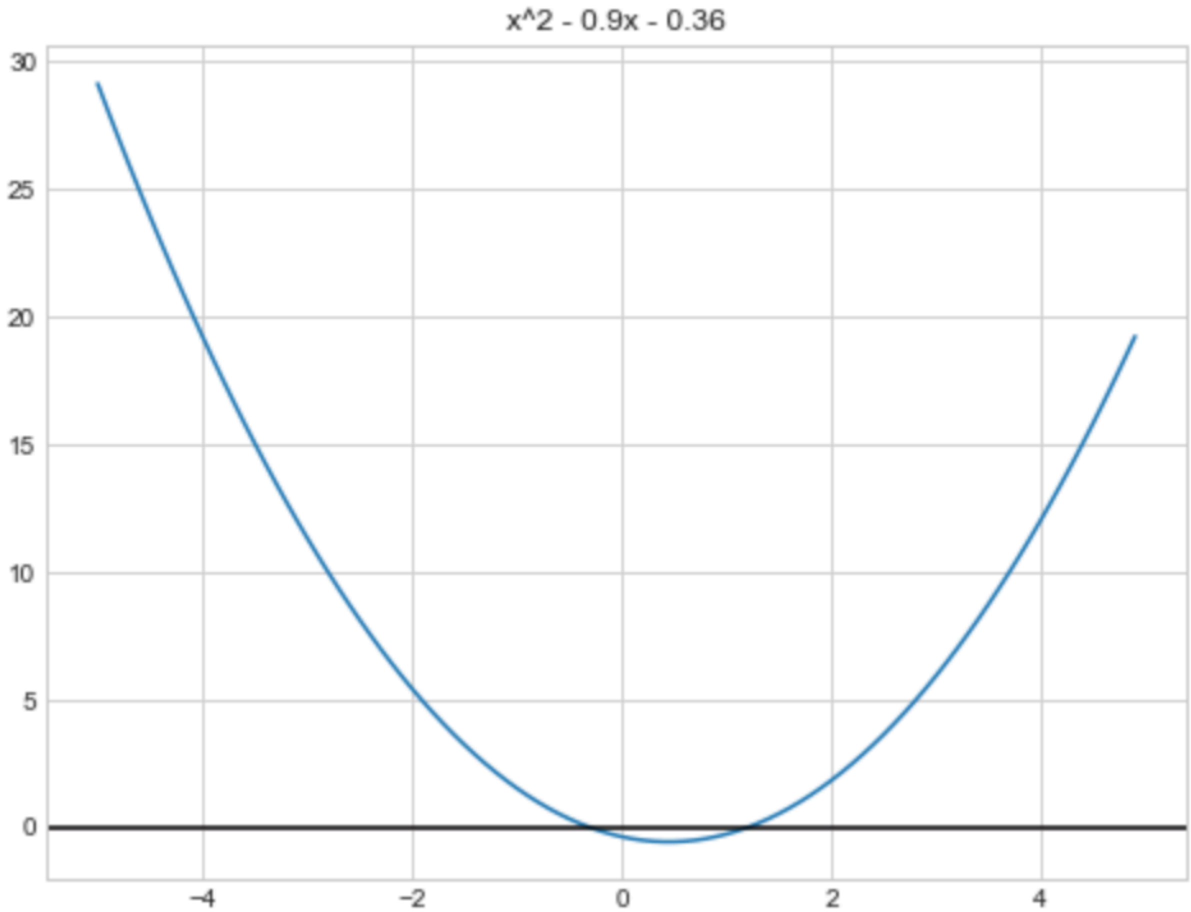
Тестовый пример 1

Используя теорему Штурма, определить число корней уравнения: на отрезке [-10, 10]. Вычислить корни с точность до 0.0001 методами половинного деления, хорд и Ньютона. Сравнить число итераций в использованных методах.

$$y = x^2 - 0.9x - 0.36$$

```
Количество корней на интервале (-10, 10): 2
Интервалы в которых находится ровно один корень: [(-1, 0), (1, 2)]

                Рассматриваемый интервал: (-1, 0)

*** Метод Биссекции (Половинного деления) ***
Корень: -0.29998779296875
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -0.2999606066662447
Количество Итераций: 8
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -0.30000000000000204
Количество Итераций: 4

                Рассматриваемый интервал: (1, 2)

*** Метод Биссекции (Половинного деления) ***
Корень: 1.20001220703125
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 1.1999505620110387
Количество Итераций: 8
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 1.2000000000005318
Количество Итераций: 4
```

$x^2 - 0.9x - 0.36$



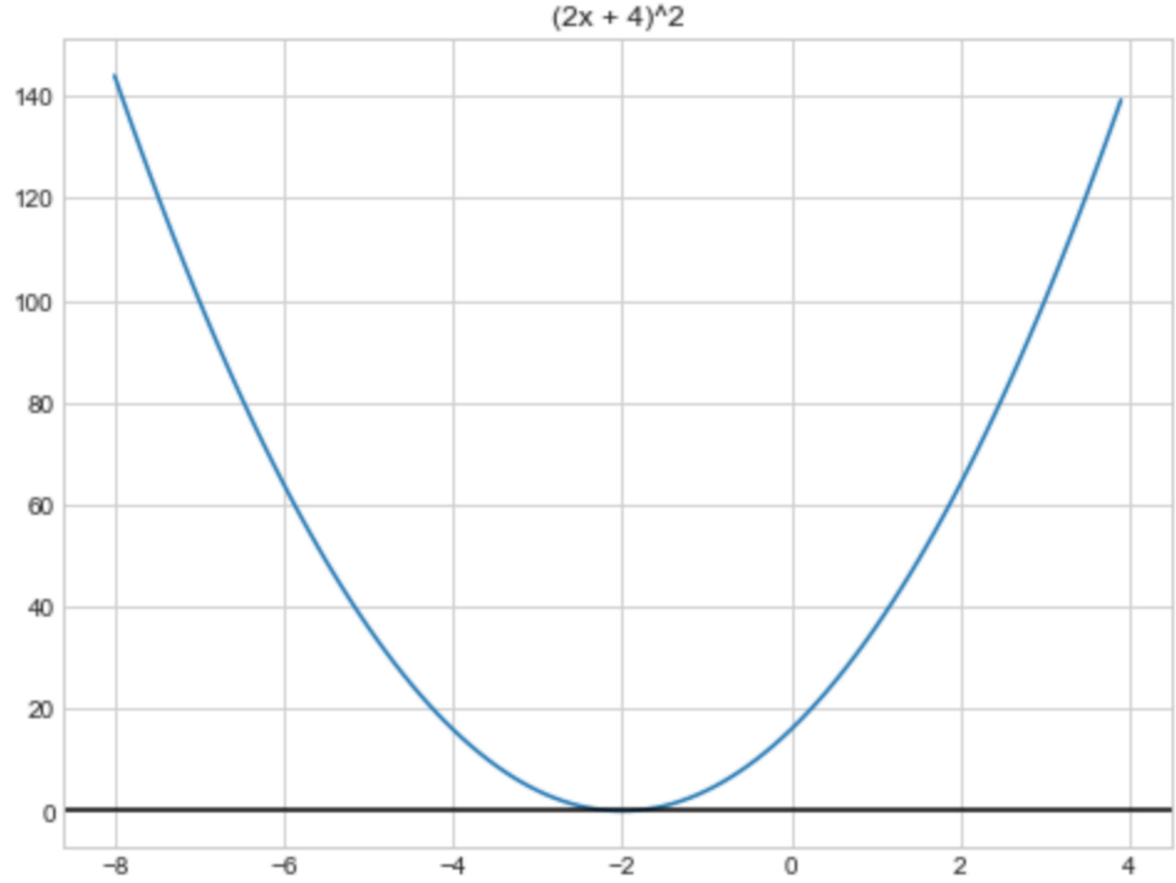
Тестовый пример 2

Используя теорему Штурма, определить число корней уравнения: на отрезке [-10, 10]. Вычислить корни с точность до 0.0001 методами половинного деления, хорд и Ньютона. Сравнить число итераций в использованных методах.

$$y = (2x + 4)^2$$

```
Количество корней на интервале (-10, 10): 1
Интервалы в которых находится ровно один корень: [(-3, -2)]

               Рассматриваемый интервал: (-3, -2)

*** Метод Биссекции (Половинного деления) ***
Корень: -2
Количество Итераций: 1

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -2.0
Количество Итераций: 1
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -2.00006103515625
Количество Итераций: 13
```

$(2x + 4)^2$



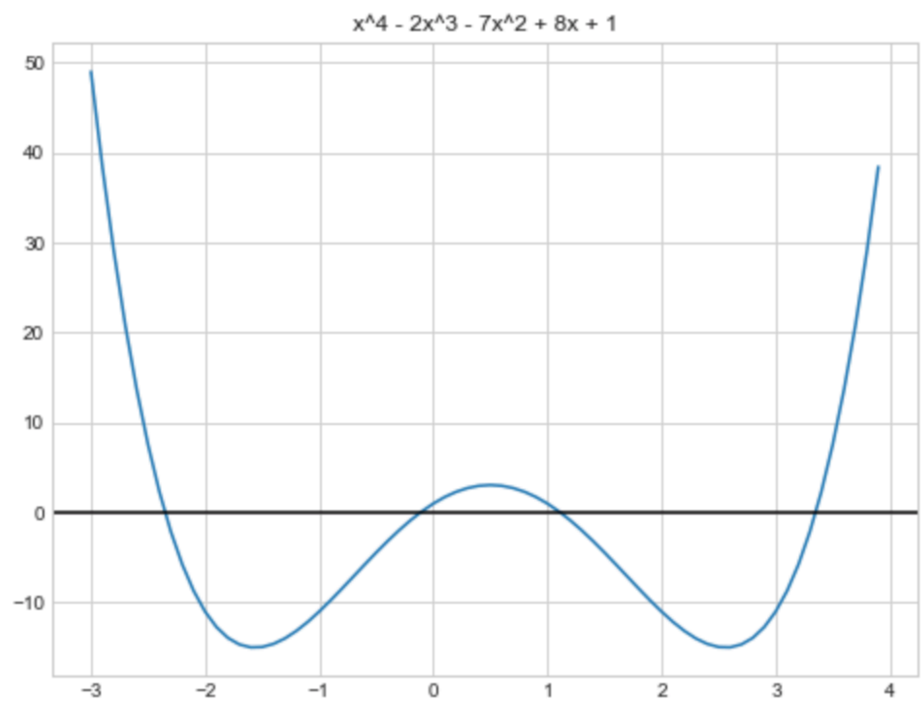
Тестовый пример 3

Используя теорему Штурма, определить число корней уравнения: на отрезке [-10, 10]. Вычислить корни с точность до 0.0001 методами половинного деления, хорд и Ньютона. Сравнить число итераций в использованных методах.

$$y = x^4 - 2x^3 - 7x^2 + 8x + 1$$

```
Количиство корней на интервале (-10, 10): 4
Интервалы в которых находится ровно один корень: [(-3, -2), (-1, 0), (1, 2), (3, 4)]

                Рассматриваемый интервал: (-3, -2)

*** Метод Биссекции (Половинного деления) ***
Корень: -2.35003662109375
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -2.350082809843179
Количество Итераций: 14
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -2.3500847963187756
Количество Итераций: 4

                Рассматриваемый интервал: (-1, 0)

*** Метод Биссекции (Половинного деления) ***
Корень: -0.11407470703125
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -0.11401153728200486
Количество Итераций: 7
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -0.11401681881954807
Количество Итераций: 4
```

```
                Рассматриваемый интервал: (1, 2)

*** Метод Биссекции (Половинного деления) ***
Корень: 1.11407470703125
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 1.114011537282005
Количество Итераций: 7
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 1.114016818819548
Количество Итераций: 4

                Рассматриваемый интервал: (3, 4)

*** Метод Биссекции (Половинного деления) ***
Корень: 3.35003662109375
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 3.3500828098431787
Количество Итераций: 14
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 3.3500847963187756
Количество Итераций: 4
```



Тестовый пример 4

Используя теорему Штурма, определить число корней уравнения: на отрезке [-10, 10]. Вычислить корни с точность до 0.0001 методами половинного деления, хорд и Ньютона. Сравнить число итераций в использованных методах.

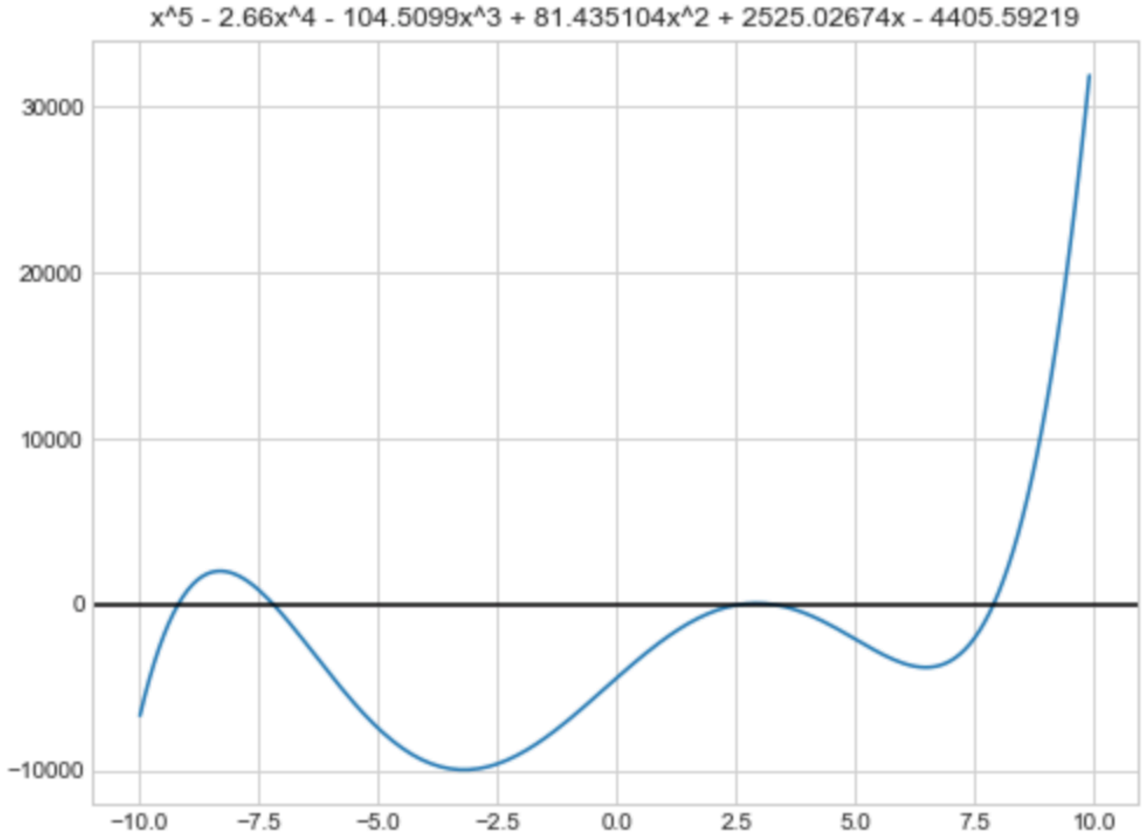
$$y = x^5 + 2.66x^4 - 104.5099x^3 - 81.435104x^2 + 2525.02674x - 4405.59219$$

```
Количество корней на интервале (-10, 10): 5
Интервалы в которых находится ровно один корень: [(-10, -9), (-8, -7), (2, 3), (3, 4), (7, 8)]

                Рассматриваемый интервал: (-10, -9)

*** Метод Биссекции (Половинного деления) ***
Корень: -9.20001220703125
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -9.19999999013451
Количество Итераций: 18
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -9.200000005308917
Количество Итераций: 4

                Рассматриваемый интервал: (-8, -7)

*** Метод Биссекции (Половинного деления) ***
Корень: -7.199981689453125
Количество Итераций: 15

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -7.200000000542465
Количество Итераций: 6
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -7.199999992932308
Количество Итераций: 4
```

```
                Рассматриваемый интервал: (2, 3)

*** Метод Биссекции (Половинного деления) ***
Корень: 2.569976806640625
Количество Итераций: 15

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 2.570000197805116
Количество Итераций: 19
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 2.5700000007375503
Количество Итераций: 3

                Рассматриваемый интервал: (3, 4)

*** Метод Биссекции (Половинного деления) ***
Корень: 3.2800140380859375
Количество Итераций: 16

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 3.279999798419035
Количество Итераций: 20
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 3.2799999923029866
Количество Итераций: 4
```

```
                Рассматриваемый интервал: (7, 8)

*** Метод Биссекции (Половинного деления) ***
Корень: 7.88995361328125
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 7.890000000061255
Количество Итераций: 7
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 7.8900000024698365
Количество Итераций: 4
```

$$x^5 + 2.66x^4 - 104.5099x^3 - 81.435104x^2 + 2525.02674x - 4405.59219$$



Тестовый пример 5

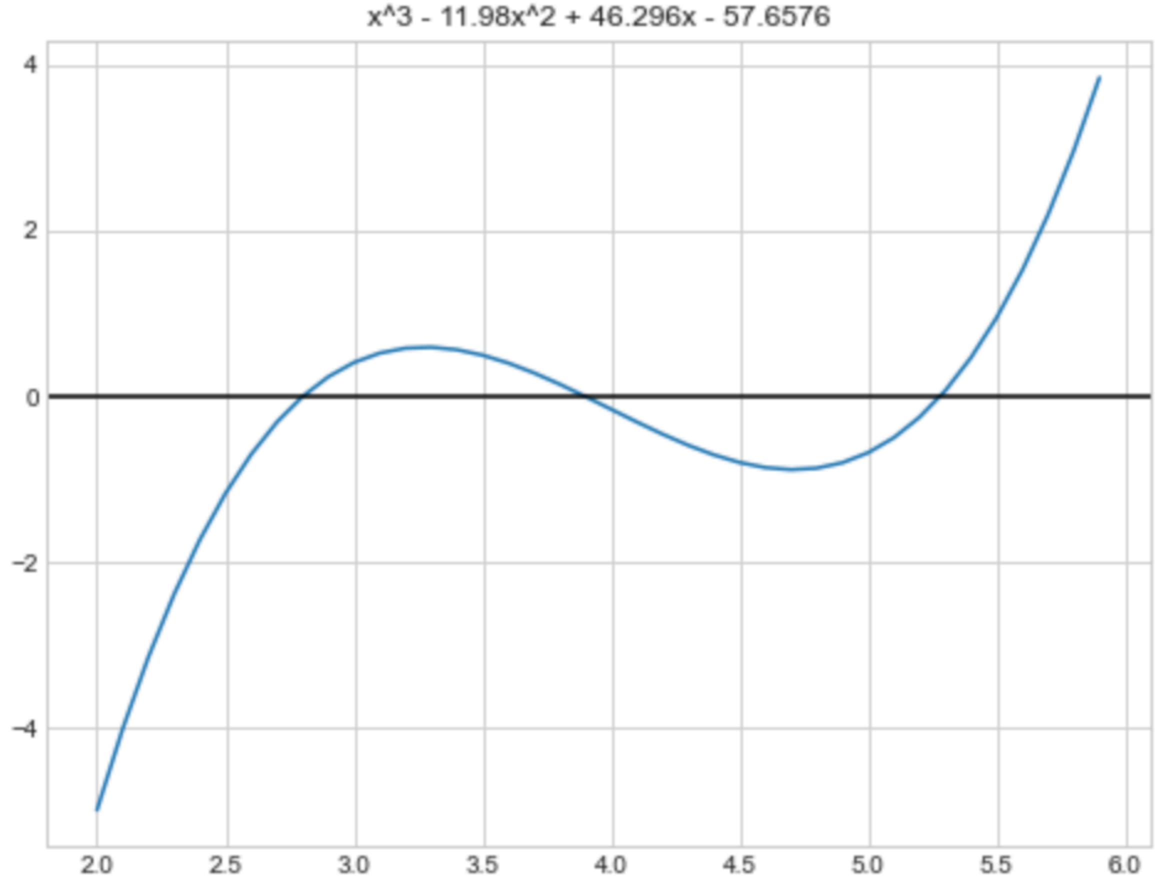
Используя теорему Штурма, определить число корней уравнения: на отрезке [-10, 10]. Вычислить корни с точность до 0.0001 методами половинного деления, хорд и Ньютона. Сравнить число итераций в использованных методах.

$$y = x^3 + 11.98^2 + 46.296x - 57.6576$$

```
Количество корней на интервале (-10, 10): 3
Интервалы в которых находится ровно один корень: [(2, 3), (3, 4), (5, 6)]

                Рассматриваемый интервал: (2, 3)

*** Метод Биссекции (Половинного деления) ***
Корень: 2.79998779296875
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 2.800025369209012
Количество Итераций: 16
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 2.799999996958965
Количество Итераций: 4

                Рассматриваемый интервал: (3, 4)

*** Метод Биссекции (Половинного деления) ***
Корень: 3.89996337890625
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 3.899999421379996
Количество Итераций: 4
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 3.9000000000023665
Количество Итераций: 4
```

```
                Рассматриваемый интервал: (5, 6)

*** Метод Биссекции (Половинного деления) ***
Корень: 5.27996826171875
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 5.279982490956562
Количество Итераций: 14
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 5.2800000000105385
Количество Итераций: 4
```

$$x^3 - 11.98x^2 + 46.296x - 57.6576$$



4. Решение задания

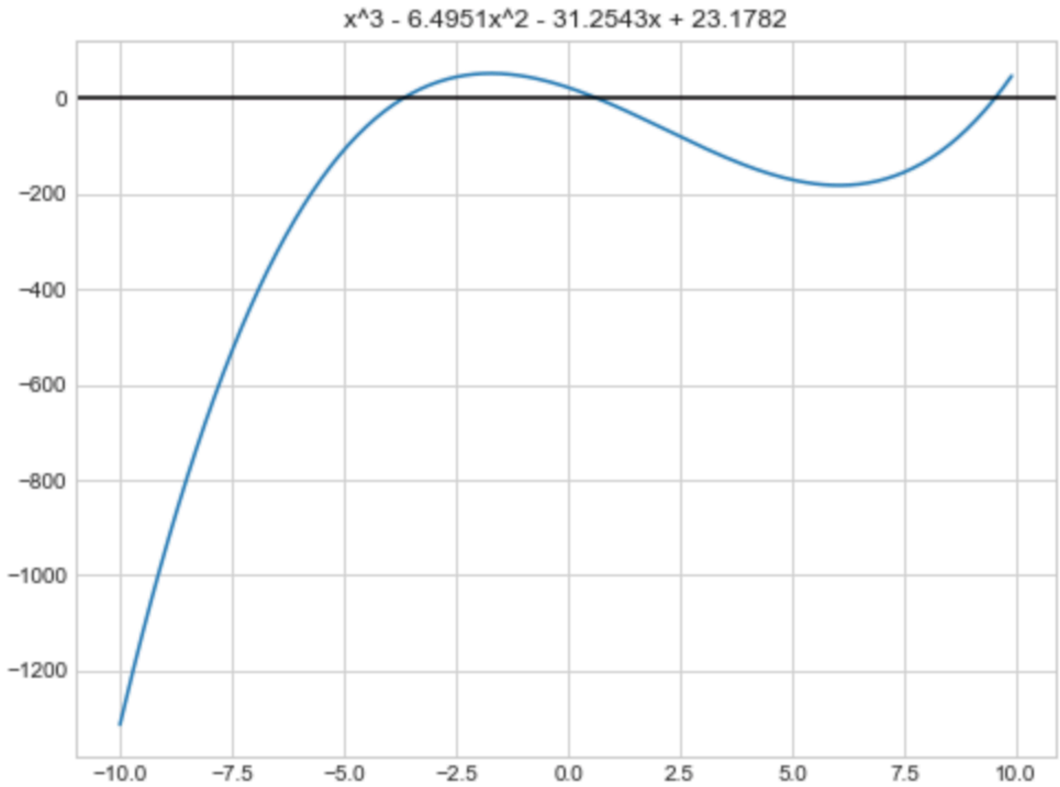
Используя теорему Штурма, определить число корней уравнения: на отрезке [-10, 10]. Вычислить наименьший корень, лежащий на отрезке, с точность до 0.0001 методами половинного деления, хорд и Ньютона. Сравнить число итераций в использованных методах.

$$y = x^3 - 6.4951^2 - 31.2543x + 23.1782$$

```
Количество корней на интервале (-10, 10): 3
Интервалы в которых находится ровно один корень: [(-4, -3), (0, 1), (9, 10)]

                Рассматриваемый интервал: (-4, -3)

*** Метод Биссекции (Половинного деления) ***
Корень: -3.686981201171875
Количество Итераций: 15

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: -3.6869587106105097
Количество Итераций: 6
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: -3.6869591161303483
Количество Итераций: 3

                Рассматриваемый интервал: (0, 1)

*** Метод Биссекции (Половинного деления) ***
Корень: 0.660247802734375
Количество Итераций: 15

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 0.6602217167039184
Количество Итераций: 4
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 0.6602230525750614
Количество Итераций: 3
```

```
                Рассматриваемый интервал: (9, 10)

*** Метод Биссекции (Половинного деления) ***
Корень: 9.52178955078125
Количество Итераций: 14

--------------------------------------------------
--------------------------------------------------
*** Метод Секущих (Хорд) ***
Корень: 9.521835859978257
Количество Итераций: 6
--------------------------------------------------
--------------------------------------------------
*** Метод Ньютона (Касательных) ***
Корень: 9.521836064553895
Количество Итераций: 2
```

$$x^3 - 6.4951x^2 - 31.2543x + 23.1782$$



5. Выводы

Таким образом, в ходе выполнения лабораторной работы были изучены методы численного решения нелинейных уравнений (метод простой итерации, метод хорд, метод Биссекции, метод Ньютона), исследована скорость сходимости итерационных процедур, составлена программа численного решения нелинейных уравнений методами бисекции, хорд, Ньютона, проверена правильность работы программы на тестовых примерах, численно решено нелинейное уравнение заданного варианта, сравнены количества итераций, необходимых для достижения заданной точности вычисления разными методами.

Оптимальным способом численного решения нелинейных уравнений является применение метода Ньютона, так скорость сходимости в этом методе почти всегда квадратичная, но иногда не квадратичная.

В ходе работы были полены 5 функций, имеющих несколько корней на заданном промежутке (-10, 10), и в ходе решения результат был проверен с помощью графика, что дает знать о том, что реализованные методы успешно справляются с решением нелинейных уравнений.

Для получения функций из ряда Штурма

```
def get_sturm_row(equation):
    x, y = symbols('x y')
    sturm_row = list()
    y = equation
    sturm_row.append(y)

    y_2 = diff(y)
    sturm_row.append(y_2)

    r = y_2
    while degree(r) > 0:
        r = div(y, y_2)[1] * (-1)
        sturm_row.append(r)
        y = y_2
        y_2 = r
    return sturm_row
```

6. Программная реализация

Разбиваем маленькие промежутки на интервалы в которых всего один корень

```
def get_1root_small_interval(sturm_row, interval: tuple):
    result_intervals = []
    digit_changes = [0, 0]

    for inter_num, inter in enumerate(interval):
        func_values = list()
        prev_val = 0
        current_val = 0
        for i, func in enumerate(sturm_row):
            func_values.append(func.subs('x', inter))
            if i == 0:
                prev_val = func_values[i]
                continue
            current_val = func_values[i]
            if prev_val == 0:
                prev_val = current_val
                continue
            if current_val == 0:
                continue

            if prev_val * current_val < 0:
                digit_changes[inter_num] += 1
            prev_val = current_val

    if digit_changes[0] == digit_changes[1]:
        return 0, 0

    if abs(digit_changes[1] - digit_changes[0]) == 1:
        return interval[0], interval[1]
    else:
        middle = float(interval[1] - interval[0]) / 2
        result_intervals.append(get_1root_small_interval(sturm_row, (interval[0], interval[0] + middle)))
        result_intervals.append(get_1root_small_interval(sturm_row, (interval[0] + middle, interval[1])))

    return result_intervals
```

Находим на большом интервале, маленьких интервальчиков, где находится всего один корень

```
def get_1root_intervals(sturm_row, interval: tuple):
    root_intervals = list()
    digit_change_times = dict()
    for i in range(interval[0], interval[1] + 1):
        func_values = list()
        digit_changes_count = 0
        for num, func in enumerate(sturm_row):
            # if num == len(sturm_row) - 1:
            #     func_values.append(func)
            #     continue
            func_values.append(func.subs('x', i))
            if num == 0:
                continue
            if func_values[num] * func_values[num-1] < 0:
                digit_changes_count += 1

        digit_change_times[i] = digit_changes_count

    for i in range(interval[0] + 1, interval[1] + 1):
        if digit_change_times[i-1] - digit_change_times[i] > 1:
            result_intervals = get_1root_small_interval(sturm_row, (i-1, i))
            clean_upped_intervals = []
            clean_upped_intervals = get_clean_intervals(result_intervals, clean_upped_intervals)

            for inter in clean_upped_intervals:
                root_intervals.append(inter)

        elif digit_change_times[i-1] != digit_change_times[i]:
            root_intervals.append((i-1, i))

    return root_intervals
```

Полный проход списка, состоящего из вложенных список

```
def traverse_list(ls, a):
    if type(ls) == tuple:
        if ls[0] != 0 or ls[1] != 0:
            a.append(ls)
        return
    for i in range(len(ls)):
        traverse_list(ls[i], a)
```

```
def get_clean_intervals(ls, a):
    traverse_list(ls, a)
    return a
```

Метод биссекции

```
def bisection(function, interval: tuple, tol=0.0001):
    x, y = symbols('x y')
    a, b = interval[0], interval[1]
    mid = (a + b) / 2

    f = lambdify(x, function, "numpy")

    if f(a) * f(b) > 0:
        raise Exception(f"Bisection method couldn't solve f(x) = 0, bcs f(a) * f(b) = {f(a) * f(b)} >= 0\n"
                        f"--> more than 1 root on ({(a, b)}) or no roots)")

    x_delta = tol * 2
    iteration = 1
    while f(mid) > tol or x_delta > tol:

        if f(a) == 0:
            return a, iteration
        elif f(b) == 0:
            return b, iteration
        elif f(mid) == 0:
            return mid, iteration

        if f(a) * f(mid) < 0:
            b = mid
        elif f(b) * f(mid) < 0:
            a = mid

        x_delta = mid
        mid = (a + b) / 2

        x_delta = abs(x_delta - mid)
        iteration += 1

    return mid, iteration
```

Метод секущих

```python
def secant(function, interval: tuple, tol=0.0001):
    x, y = symbols('x y')
    a, b = interval[0], interval[1]
    f = lambdify(x, function, "numpy")
    if f(a) * f(b) > 0:
        raise Exception(f"Secant method couldn't solve f(x) = 0, bcs f(a) * f(b) = {f(a) * f(b)} >= 0\n"
                        f"--> more than 1 root on ({(a, b)}) or no roots)")

    x_sec = a - (f(a) * (b - a)) / (f(b) - f(a))


    x_delta = 0
    iteration = 1
    while abs(f(x_sec)) > tol or x_delta > tol:

        if f(a) == 0:
            return a, iteration
        elif f(b) == 0:
            return b, iteration
        elif f(x_sec) == 0:
            return x_sec, iteration

        if f(x_sec) * f(a) < 0:
            b = x_sec
        elif f(x_sec) * f(b) < 0:
            a = x_sec

        x_delta = x_sec
        x_sec = a - (f(a) * (b - a)) / (f(b) - f(a))

        x_delta = abs(x_delta - x_sec)
        iteration += 1


    return x_sec, iteration
```

Метод Ньютона (касательных)

```python
def newton(function, interval, tol=0.0001):

    x, y = symbols('x y')
    a, b = interval[0], interval[1]


    x_start = (a + b) / 2
    f = lambdify(x, function, 'numpy')
    if f(a) * f(b) > 0:
        raise Exception(f"Newton method couldn't solve f(x) = 0, bcs f(a) * f(b) = {f(a) * f(b)} >= 0\n"
                        f"--> more than 1 root on ({(a, b)}) or no roots)")

    y_diff = function.diff()
    f_diff = lambdify(x, y_diff, 'numpy')

    x_newton = x_start - f(x_start) / f_diff(x_start)

    x_delta = tol * 2
    iteration = 1
    while abs(f(x_newton)) > tol or x_delta > tol:

        if f(x_newton) == 0:
            return x_newton, iteration

        x_delta = x_newton
        x_newton = x_newton - f(x_newton) / f_diff(x_newton)
        x_delta = abs(x_delta - x_newton)
        iteration += 1



    return x_newton, iteration
```