Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа по вычислительной математике №2

Численное решение нелинейных уравнений и систем

Преподаватель: Малышева Татьяна Алексеевна

Выполнила: Голованова Дарья Владимировна

Группа: P3222

Санкт-Петербург,
2022г

Цель работы

Реализовать метод секущих и метод простой итераций для решения нелинейных уравнений и реализовать решение систем линейных уравнений методом простой итерации

Описание использованного метода

Метод секущих:

Суть метода заключается в том, что функция $y = f(x)$ на отрезке $[a, b]$ заменяется касательной, а в качестве приближенного значения корня $x^* = x_n$ принимается точка пересечения касательной с осью абсцисс.

$$x_1 = x_0 - h_0$$

$$h_0 = \frac{f(x_0)}{\tan\alpha} = \frac{f(x_0)}{f'(x_0)x_1} = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Однако, вышесказанное относится к методу Ньютона. Для того, чтобы получить метод секущих, нужно:

Упростить метод Ньютона, заменив f'(x) разностным приближением: $f'(x_i) \approx (f'(x_i) - f'(x_{i-1}))/(x_i - x_{i-1})$

Рабочая формула метода: $x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i),\ i = 1, 2 \ldots$

Метод секущих является двухшаговым, т.е. новое приближение x_{+1} определяется двумя предыдущими итерациями x_i и x_{i-1}. Выбор x_0 определяется, как и в методе Ньютона, x_1- выбирается рядом с начальным самостоятельно.

Критерий окончания итерационного процесса: $|x_n - x_{n-1}| \leq \varepsilon$ или $|f(x_n)f'(x_n)| \leq \varepsilon$ или $|f(x_n)| \leq \varepsilon$

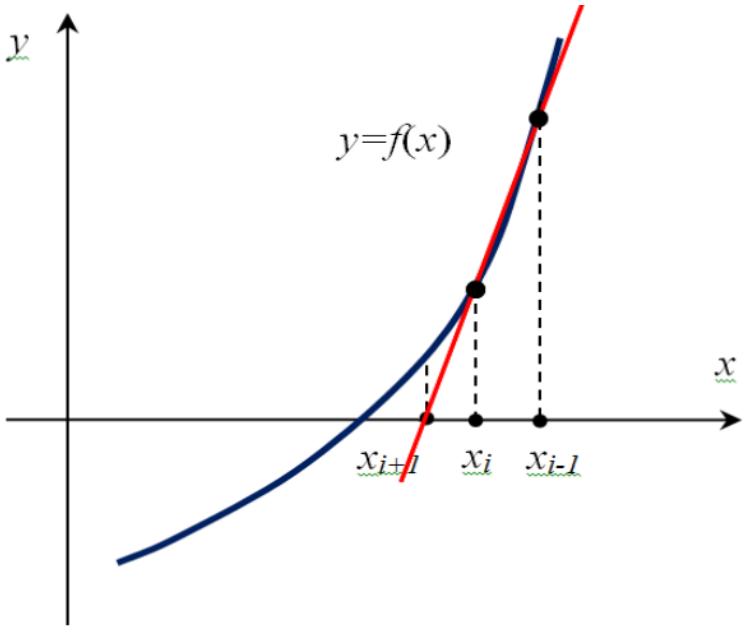


Рисунок 1

Метод простой итерации:

Суть метода заключается в том, что уравнение $f(x) = 0$ с помощью некоторых преобразований необходимо переписать в виде $x = \varphi(x)$ (как показано на Рисунке 2).

Уравнение $f(x) = 0$ эквивалентно уравнению $x = x + \lambda(x)f(x)$ для любой функции $\lambda(x) \neq 0$. Возьмем $\varphi(x) = x - \lambda(x)f(x)$ и выберем функцию (или переменную) $\lambda(x) \neq 0$ так, чтобы функция $\varphi(x)$ удовлетворяла необходимым условиям.

Для нахождения корня уравнения $x = \varphi(x)$ выберем некоторое начальное значение x_0, которое должно находиться как можно ближе к корню уравнения. Дальше с помощью итерационной формулы $x_{n+1} = \varphi(x_n)$ будем находить каждое следующее приближение корня уравнения.

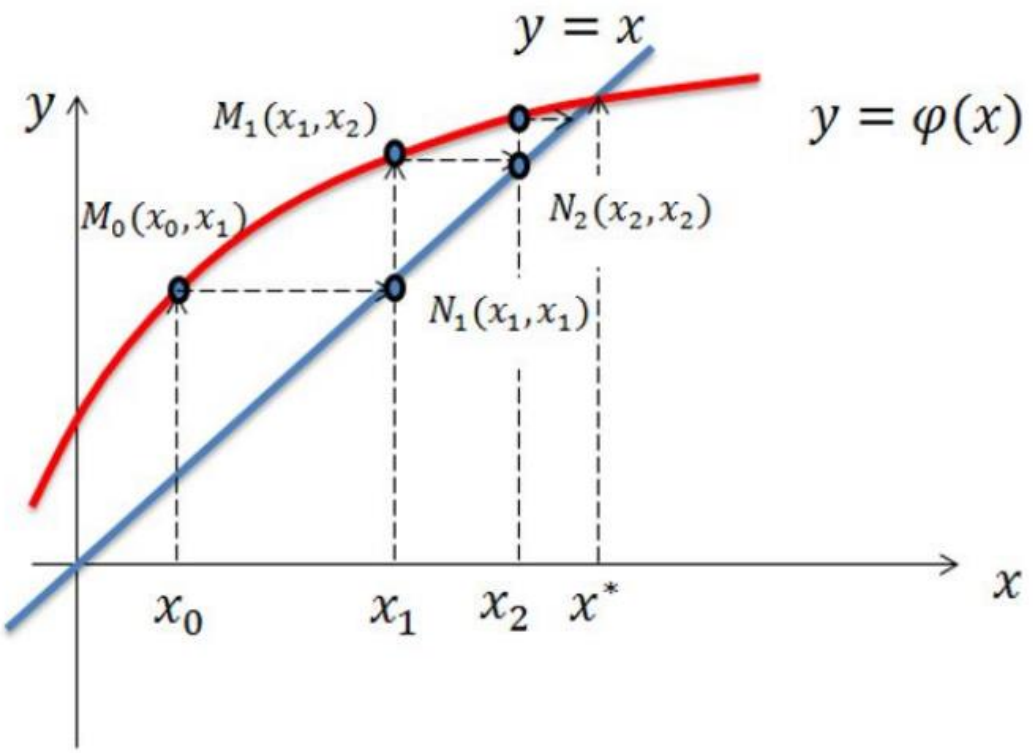


Рисунок 2

Рабочая формула метода:

$x_{i+1} = \varphi(x_i)$

Условия сходимости метода простой итерации определяются теоремой:

Если в некоторой σ - окрестности корня x* уравнения $f(x) = 0$ функция $x = \varphi(x)$ дифференцируема и удовлетворяет неравенству $|\varphi'(x)| < q$, где $0 \leq q < 1$ постоянная, то независимо от выбора начального приближения $x0$ из указанной σ окрестности итерационная последовательность xn не выходит из этой окрестности, метод сходится со скоростью геометрической прогрессии.

Достаточное условие сходимости метода:

$|\varphi'(x)| \leq q < 1$, где q – некоторая константа

Критерий окончания итерационного процесса:

$|x_n - x_{n-1}| \leq \varepsilon$

Метод простой итерации для решения систем нелинейных уравнений:

Суть метода заключается в том, чтобы привести первоначальную систему уравнений к следующему виду:

$$\begin{cases} x_1 = \varphi_1(x_1, \ldots, x_n), \\ x_2 = \varphi_1(x_2, \ldots, x_n), \\ \quad \vdots \\ x_n = \varphi_n(x_1, \ldots, x_n), \end{cases}$$

Для этого нам необходимо задать начальное приближение $x^{(0)} = (x_{10}, x_{20}, \ldots, x_{n0})^T$ и малое положительное число ε (точность).

Затем вычислить $x^{(k+1)}$ по формуле $x^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$, так продолжать увеличивая k на единицу пока не будет достигнут критерий окончания итерационного процесса. Критерий завершения итерационного процесса: $|x_i^{(k+1)} - x_i^{(k)}| \leq \varepsilon$, значит процесс завершен, $x^* = x^{(k+1)}$.

Выводы

В результате выполнения я изучила 5 методов решения нелинейных уравнений и пришла к следующим выводам относительно их преимуществ и недостатков:

1. Метод касательных:
   Достоинства: Метод обладает квадратичной сходимостью.
   Недостатки: Необходимость вычисления производной на каждой итерации.

2. Метод простой итерации:
   Достоинства: Простота реализации
   Недостатки: Сходимость метода в малой окрестности корня и вытекающая отсюда необходимость выбора начального приближения к корню из этой малой окрестности. В противном случае итерационный процесс расходится или сходится к другому корню этого уравнения. Также при $|\varphi'(x)| \approx 1$, то сходимость может быть очень медленной.

3. Метод секущих:
   Достоинства: Меньший объем вычислений по сравнению с методом Ньютона, т.к. не требуется вычислять производную.
   Недостатки: Порядок сходимости метода секущих ниже, чем у метода касательных и равен золотому сечению ≈1,618 (сверхлинейная).

4. Метод половинного деления:
   Достоинства: Обладает абсолютной сходимостью (близость получаемого численного решения задачи к истинному решению.) Устойчив к ошибкам округления.
   Недостатки: Если интервал содержит несколько корней, то неизвестно к какому относится вычислительный процесс. Медленный метод: имеет линейную сходимость. Имеет смысл применять в случаях, когда требуется высокая надежность счета, а скорость несущественна.

5. Метод хорд:
   Достоинства: Простота реализации

Недостатки: Скорость сходимости – линейная. Порядок сходимости метода хорд выше, чем у метода половинного деления.

Что касается Методов решения систем нелинейных уравнений, то мною были изучены два метода решения и сделаны следующие выводы:

Метод Ньютона для решения СНАУ представляет собой обобщение метода Ньютона для решения НУ его сутью является попытка свести решение системы нелинейных уравнений к решению системы линейных уравнений. Основная сложность метода Ньютона заключается в обращении матрицы Якоби. Вводя обозначение $\Delta x\,(k) = x\,(k+1) - x\,(k)\Delta x\,(k) = x\,(k+1) - x\,(k)$ получаем СЛАУ для вычисления $\Delta x\,(k)$. Решение этого СЛАУ создает основную вычислительную нагрузку алгоритма.

Метод простой итерации же в свою очередь позволяет, грубо говоря подобрать вектор решений системы уравнений путем выражения значения одной неизвестной через все остальные и постепенной подстановки значений неизвестных, вычисляемых на каждом шаге.

Блок-схемы

Метод секущих

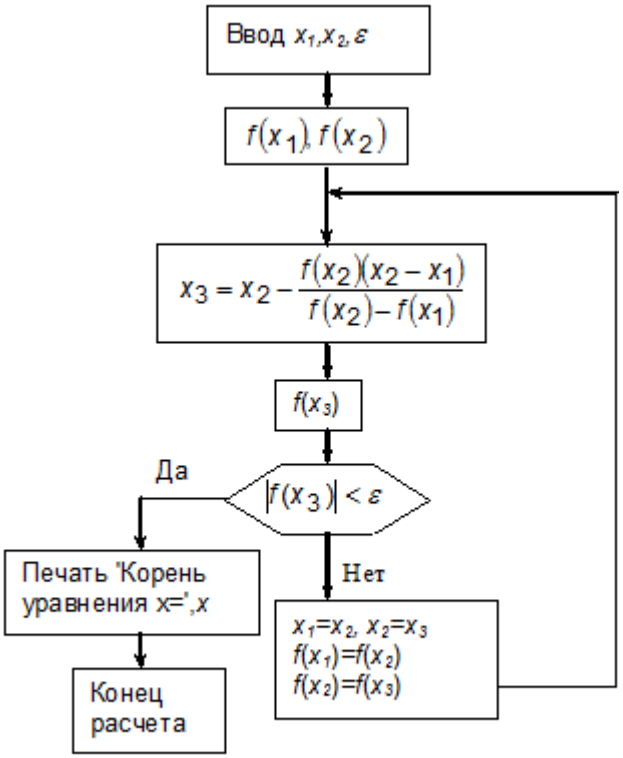
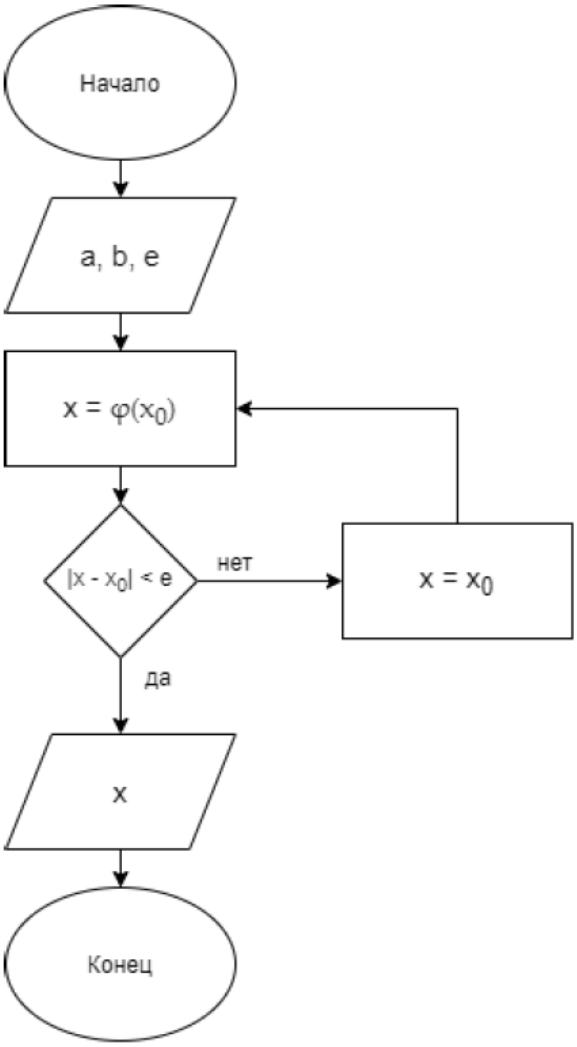
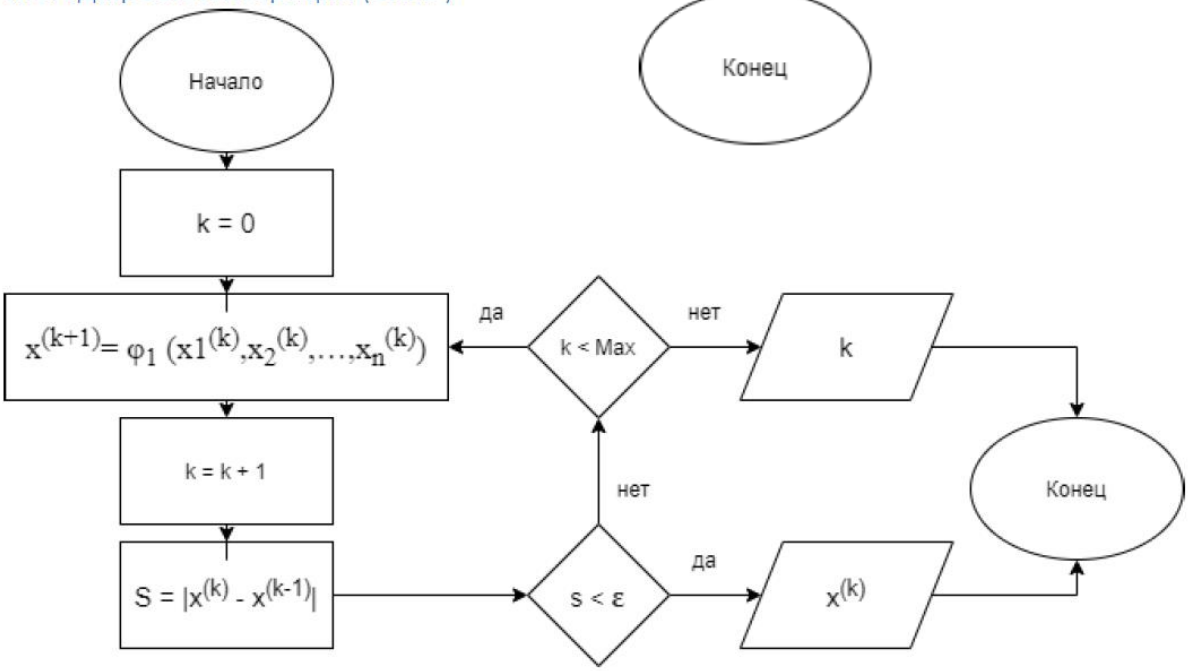


Рис.22.5. Блок-схема метода секущих

Метод простой итерации (НУ)



Метод простой итерации (СНАУ)



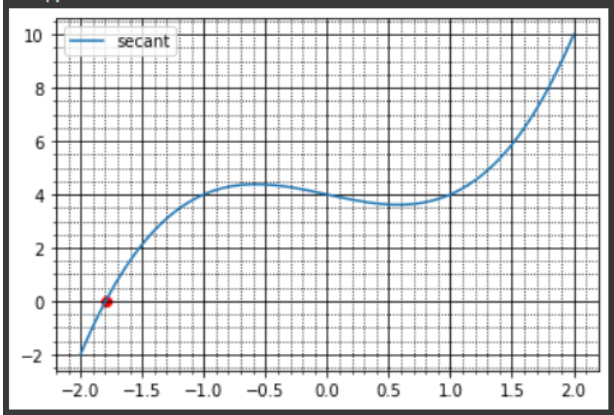
Листинг численных методов:

```
class NonLinearSystem:
    def iteration(fu1, fu2,  eps: float = 1e-7, kmax: int = 1e3) -> float:
        fig = plt.subplots()
        fuplot = lambda x: (-1) * np.arcsin(x + 15) + 1.15
        xplt = np.linspace(-20, 10, 100)
        plt.plot(xplt, fuplot(xplt), label='y = arcsin(x + 15) - 1.5')
        plt.plot(xplt, fu1(xplt), label='y = - 0.5 - cos(x-2)')
        plt.minorticks_on()
        plt.grid(which='major',
                 color='k')
        plt.grid(which='minor',
                 color='k',
                 linestyle=':')
        plt.legend()
        x0, y0, d1, d2, i = 0, 0, 1, 1, 0
        while 1:
            x, y = fu1(y0), fu2(x0)
            d1, d2 = fu1(x) - x, y - fu2(y)
            x0, y0, i = x, y, i + 1
            plt.scatter(y, fuplot(y), color='black')
            if not (abs(d1) > eps and abs(d2) > eps and i < kmax):
                break
        plt.scatter(y, fuplot(y), color='red')
        plt.show()
        return x, y, i

class NonLinearEquation:
    def secant(f: Callable[[float], float], x0: float, eps: float = 1e-
7, kmax: int = 1e3, left: float = -5,
               right: float = 5) -> float:
        x, x_prev, i = x0, x0 + 2 * eps, 0
        while abs(x - x_prev) >= eps and i < kmax:
            x, x_prev, i = x - f(x) / (f(x) - f(x_prev)) * (x - x_prev), x, i
+ 1

        fig = plt.subplots()
        xplt = np.linspace(left, right, 100)
        plt.plot(xplt, f(xplt), label="secant")
        plt.minorticks_on()
        plt.grid(which='major',
                 color='k')
        plt.grid(which='minor',
                 color='k',
                 linestyle=':')
        plt.scatter(x, f(x), color='red')
        plt.legend()
        plt.show()
        return x, i
```

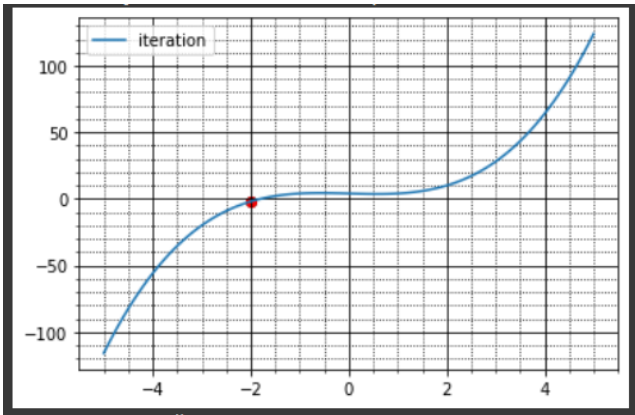
```
    def iteration(f: Callable[[float], float], x0: float, eps: float = 1e-
7, kmax: int = 1e3) -> float:
        x, x_prev, i = x0, 0, 0
        fig = plt.subplots()
        xplt = np.linspace(-5, 5, 100)
        plt.plot(xplt, f(xplt), label="iteration")
        plt.minorticks_on()
        plt.grid(which='major',
                 color='k')
        plt.grid(which='minor',
                 color='k',
                 linestyle=':')
        while abs(x - x_prev) >= eps and i < kmax:
            x_prev, x, i = x, x - 0.003 * f(x), i + 1
            i = i + 1
            plt.scatter(x, f(x), color='black')

        plt.scatter(x, f(x), color='red')
        plt.legend()
        plt.show()
        return x, i
```

Примеры

```
Введите номер уравнения:
1)x^3 + 2.28*x^2 - 1.934*x - 3.907
2)x^3 - x + 4
3)x^2 - 20 * sin(x)
q)Выйти 2
Введите левую границу: -2
Введите правую границу: 2
Введите точность: 0.01
```

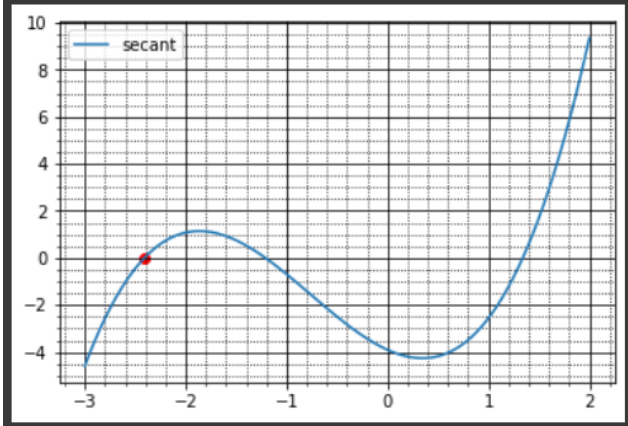



```
Метод секущих: Ответ -1.7963498240517297
Метод секущих: Количество итераций 3
```

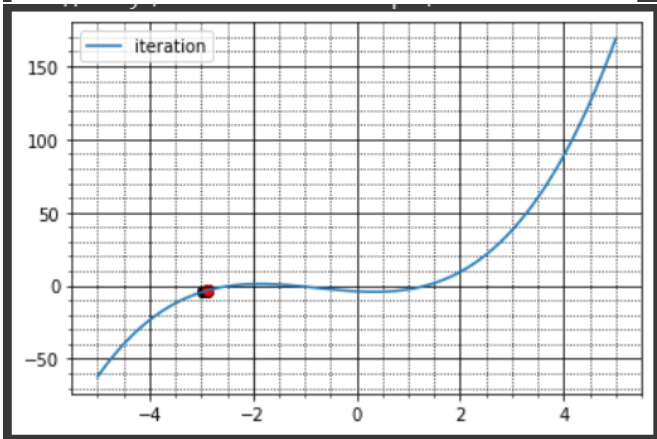



```
Метод итераций: Ответ -1.994
Метод итераций: Количество итераций 2
```

```
Введите номер уравнения:
1)x^3 + 2.28*x^2 - 1.934*x - 3.907
2)x^3 - x + 4
3)x^2 - 20 * sin(x)
q)Выйти 1
Введите левую границу: -3
Введите правую границу: 2
Введите точность: 0.01
```

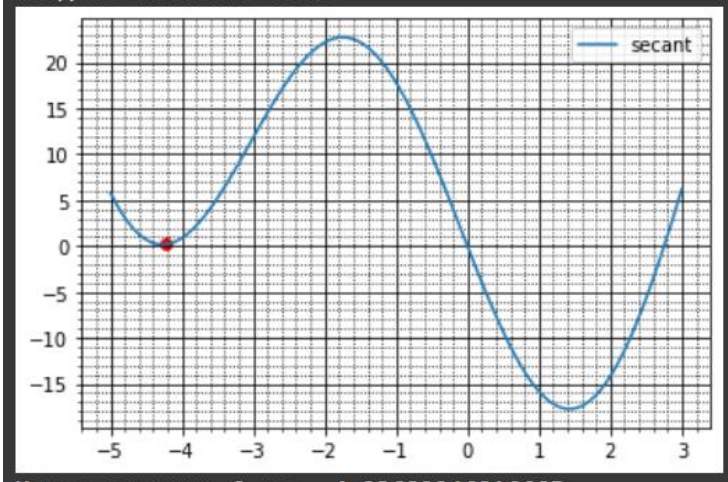



```
Метод секущих: Ответ -2.409762934252397
Метод секущих: Количество итераций 5
```

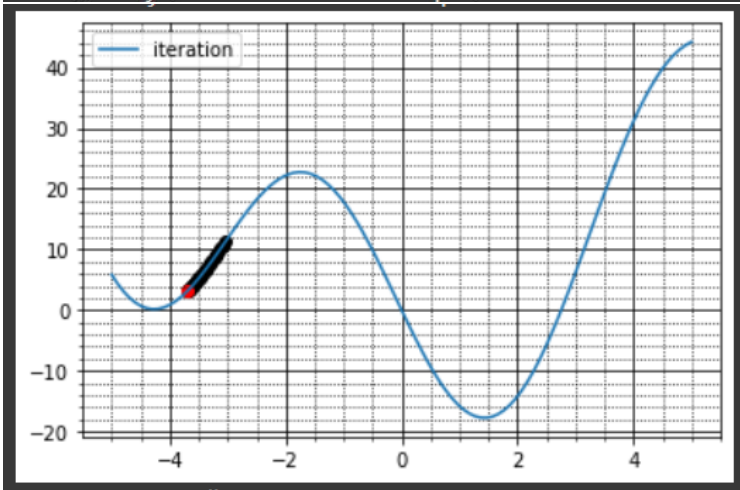



```
Метод итераций: Ответ -2.8709855296467395
Метод итераций: Количество итераций 22
```

```
Введите левую границу: -5
Введите правую границу: 3
Введите точность: 0.01
```

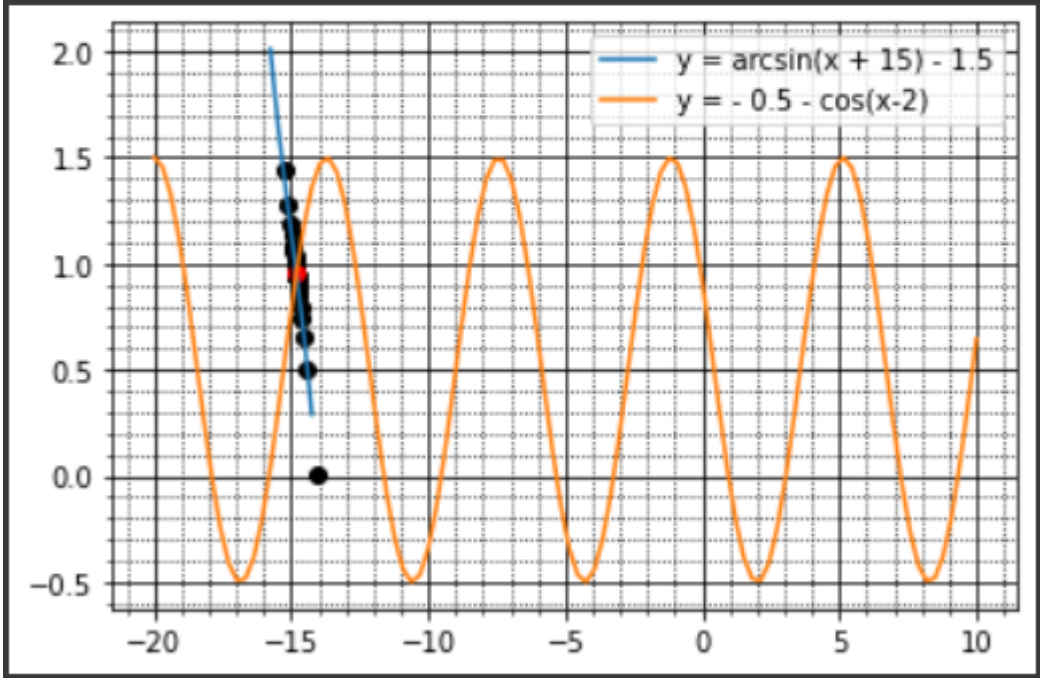



```
Метод секущих: Ответ -4.23689246816665
Метод секущих: Количество итераций 116
```




```
Метод итераций: Ответ -3.6933682293100714
Метод итераций: Количество итераций 72
```

```
Введите номер системы уравнений:
1)=======================
|y + cos(x-2) = -0.5
|y-arcssin(x+15)=1.5
=========================
q) Выйти1
Введите погрешность0.001
```




```
Первый корень системы:  0.9511498328793181
Второй корень системы:  -14.810706271741322
Количество итераций:  1000
```

Таблицы

Уточнение корня уравнения методом половинного деления (хорд)

№ шага	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	-2.5	-2	-2.25	-0.4470	1.0809	0.5963	0.5
2	-2.25	-2	-2.125	0.5963	1.0809	0.9026	0.25
3	-2,125	-2	-2,0625	0.9026	1.0809	1.0070	0.125
4	-2,125	-2,0625	-2.09375	0.9026	1.0070	0.9587	0.0625
5	-2,125	-2.09375	-2.109375	0.9026	0.9587	0.9317	0.03125
6	-2,125	-2.109375	-2.117187	0.9026	0.9317	0.9174	0.015625
7	-2,125	-2.117187	-2.12109375	0.9026	0.9174	0.91012	0.0078125

Уточнение корня уравнения методом Ньютона

№ итерации	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_k - x_{k+1} \rvert$
1	-1	-0.6930	-5.494	-1.1261	0.12613
2	-1.1261	-0.2658	-5.8009	-1.1719	0.04583
3	-1.1719	-0.1187	-5.9045	-1.1920	0.02010
4	-1.1920	-0.0557	-5.9486	-1.2013	0.00937

Уточнение корня уравнения методом простой итерации

№ итерации	x_k	$f(x_k)$	x_{k+1}	$\varphi(x_k)$	$\lvert x_k - x_{k+1} \rvert$
1	-3	-4.429654	-2.98624	-2.972956	0.0279
2	-2.986245	-4.281972	-2.972956	-2.960110	0.0026
3	-2.972956	-4.14144	-2.96011	-2.947685	0.0127
4	-2.96011	-4.00759	-2.947685	-2.93566	0.0124
5	-2.947685	-3.8800	-2.93566	-2.92402	0.012