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

Мегафакультет компьютеных технологий и управления
Факультет программной инженерии и компьютерной техники

Отчёт
по лабораторной работе №2
по дисциплине «Вычислительная математика»

Вариант 2

Группа: P3218

Студент: Богданова Мария Михайловна

Преподаватель: Бострикова Дарья Константиновна

Санкт-Петербург
2024

Содержание

1 Цель работы

Цель работы: изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

2 Задание

Требования к программе:

- Все численные методы должны быть реализованы в виде отдельных подпрограмм/методов/классов.
- ользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.
- редусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя
- Выполнить верификацию исходных данных. Необходимо анализировать наличие корня на введенном интервале. Если на интервале несколько корней или они отсутствуют – выдавать соответствующее сообщение. Программа должна реагировать на некорректные введенные данные.
- Для методов, требующих начальное приближение к корню (методы Ньютона, секущих, хорд с фиксированным концом, простой итерации), выбор начального приближения (a или b) вычислять в программе
- Для метода простой итерации проверять достаточное условие сходимости метода на введенном интервале.
- Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.
- Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

3 Исходный код программы

Репозиторий на GitHub: ссылка

```
def newton_method(a, b, quation_num, e):
    if validator.check_epsilon(e) is False:
        return "Epsilon cannot be less than zero."
    if validator.check_range(a, b) is False:
        return "The left border cannot be larger than the right!"

    if count_roots(a, b, quation_num, e) == 1:

        iterations = 0
        max_iter = 250
        x = 0
        answer = []

        f_a = f(quation_num, a)
        f_b =  f(quation_num, b)
        dx2df_a = f2dx(quation_num, a)
        dx2df_b = f2dx(quation_num, b)

        if f_a * dx2df_a > 0:
            x = a
        elif f_b * dx2df_b > 0:
            x = b
        else:
            return "No suitable initial approximation\nfound on the interval [a, b]"

        while abs(f(quation_num,x)) > e and iterations < max_iter:
            x = x - f(quation_num,x) / fdx(quation_num,x)
            iterations += 1

        answer.append(x)
        answer.append("{:.8f}".format((f(quation_num, x))))
        answer.append(iterations)

        return answer
    else:
        return "The system has several solutions \nor none at all."
```

Листинг 1: Метод Ньютона

```
def iteration_method_single(a, b, quation_num, epsilon):
    if validator.check_epsilon(epsilon) is False:
        return "Epsilon cannot be less than zero."
    if validator.check_range(a, b) is False:
        return "The left border cannot be larger than the right!"

    if count_roots(a, b, quation_num, epsilon) == 1:

        def phi(x, lambda_factor):
            phi = x + lambda_factor * f(quation_num, x)
            return phi

        def phi_dx(x, lambda_factor):
            phi_dx = 1 + lambda_factor * fdx(quation_num, x)
            return phi_dx
        answer = []
        iterations = 0
        x = a
        max_fdx = max(abs(fdx(quation_num, a)), abs(fdx(quation_num, b)))
        if fdx(quation_num, a) * fdx(quation_num, b) > 0:
            lambda_factor = -1/(max_fdx)
        elif fdx(quation_num, a) * fdx(quation_num, b) < 0:
            lambda_factor = 1/(max_fdx)

        if abs(phi_dx(x, lambda_factor)) < 0.79:
            # print(abs(phi_dx(x, lambda_factor)))

            while True:
                x_prev = x
                x = phi(x, lambda_factor)
                iterations += 1
                # print(x)

                if abs(x - x_prev) < epsilon or abs(f(quation_num, x)) < epsilon:
                    break
                elif iterations > 100:
                    return "Convergence speed is too slow."

            answer.append(x)
            answer.append(f(quation_num, x))
            answer.append(iterations)
            return answer
        else:
            return "The convergence condition is not met\n
            or the convergence rate is too slow."
    else:
        return "The system has several solutions \nor none at all."
```

Листинг 2: Метод простых итераций

```
def chorde_method(a, b, quation_num, e):
    if validator.check_epsilon(e) is False:
        return "Epsilon cannot be less than zero."
    if validator.check_range(a, b) is False:
        return "The left border cannot be larger than the right!"

    if count_roots(a,b, quation_num, e) == 1:
        prev_x = 0
        x = a - ((b-a)/(f(quation_num, b)- f(quation_num, a)))*
            f(quation_num, a)
        iter_cnt = 1
        max_iter = 200
        answer = []
        while abs(f(quation_num, x)) <= e or abs(a-b) <= e
                or abs(x - prev_x) <= e or iter_cnt < max_iter:
            if (f(quation_num, a) * f(quation_num, x)) < 0:
                b = x
            else:
                a = x
            prev_x = x
            x = a - ((b-a)/(f(quation_num, b)-
                    f(quation_num, a)))* f(quation_num, a)
            iter_cnt += 1

            if abs(f(quation_num, x)) <= e or abs(a-b) <= e or
                abs(x - prev_x) <= e or iter_cnt >= max_iter:
                break

        answer.append(x)
        answer.append((f(quation_num, x)))
        answer.append(iter_cnt)
        return answer
    else:
        return "The system has several solutions \n or none at all."
```

Листинг 3: Метод хорд

4 Расчётные формулы

Метод Ньютона

Функция $y = f(x)$ на отрезке $[a, b]$ заменяется касательной и в качестве приближенного значения корня принимается точка пересечения касательной с осью абсцисс.
Начальное приближение - x_0 на отрезке $[a, b]$.
Уравнение касательной к графику функции $y = f(x)$ в этой точке:
$y = f(x) + f'(x_0)(x - x_0)$
Пересечение касательной с осью x: $x_1 = x_0 - (f(x_0))/(f'(x_0))$ Условие прекращения итерационного процесса: $|x_n - x_{n-1}| <= \epsilon$, где ϵ - заданная точность.
x_{n+1} - приближенное решение.

Метод простых итераций

Уравнение $y = f(x)$ приведем к эквивалентному виду: $x = \psi(x)$ выразив x из исходного уравнения.
Зная начальное приближение на отрезке $[a, b]$, найдем очередные приближения $x_1 = \psi(x_0) -> x_2 = \psi(x_1)$
Условия сходимости метода простой итерации определяются следующей теоремой.
Если на отрезке локализации $[a, b]$ функция (x) определена, непрерывна и дифференцируема и удовлетворяет неравенству:
$|\psi'(x)| < q$, где $0 <= q < 1$, то независимо от выбора начального приближения $x[a, b]$ итерационная последовательность x_n метода будет сходится к корню уравнения.
Достаточное условие сходимости метода: $|\psi'(x)| <= q < 1$
Критерий окончания итерационного процесса:$|x_n - x_{n-1}| <= \epsilon$

Метод хорд

Функция $y = f(x)$ на отрезке $[a, b]$ заменяется хордой, и в качестве приближенного значения корня принимается точка пересечения хорды с осью абсцисс. Уравнение хорды, проходящей через точки $A(a, f(a))$ и $B(b, f(b))$:

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}$$

Точка пересечения хорды с осью абсцисс $(y = 0)$:

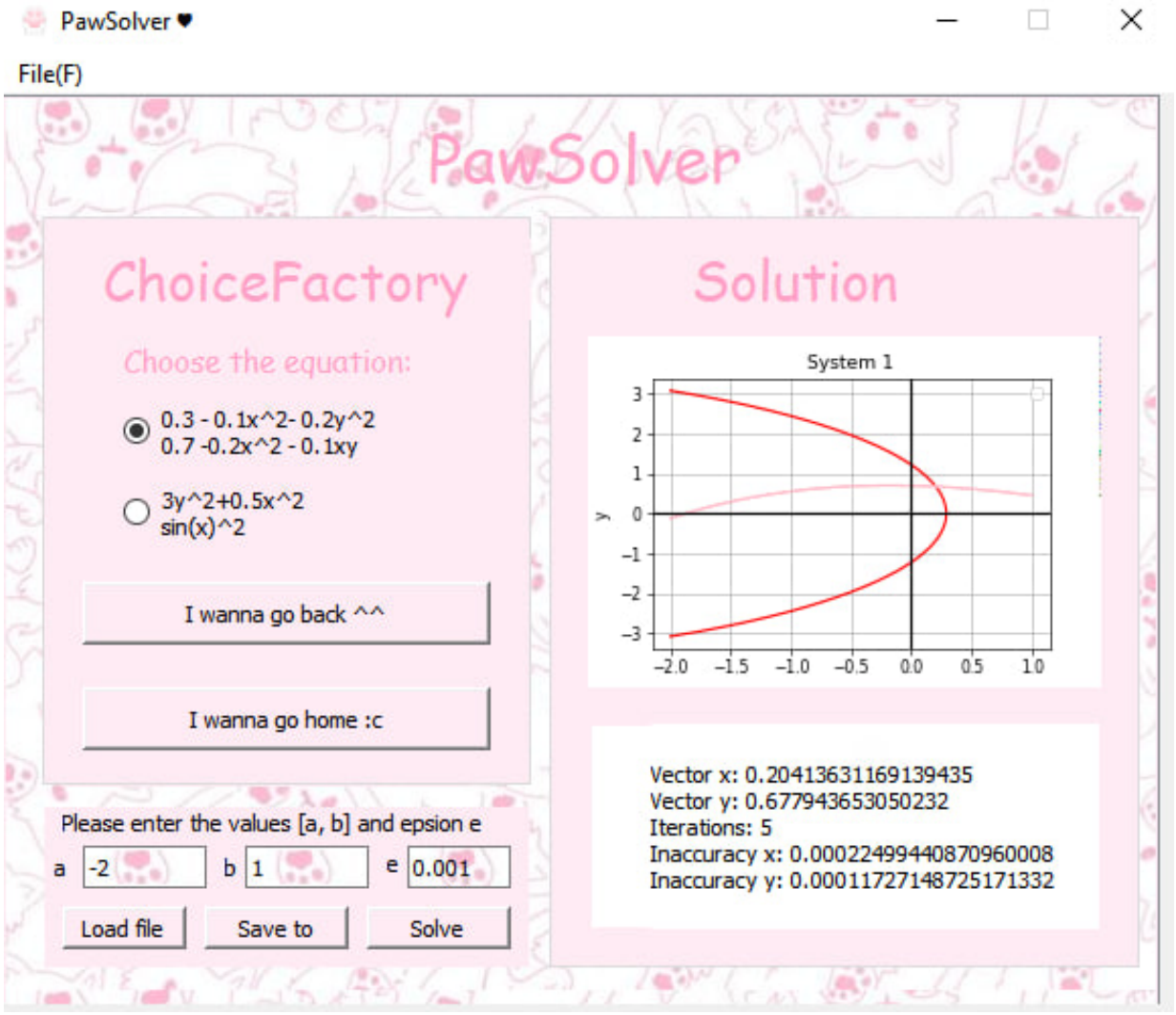
$$x = a - \frac{b - a}{f(b) - f(a)} f(a)$$

Интервал изоляции корня $[a_0, b_0]$ x_0: $x_0 = a_0 - \frac{b_0 - a_0}{f(b_0) - f(a_0)} f(a_0)\ f(x_0)$. В качестве нового интервала выбираем ту половину отрезка, на концах которого функция имеет разные знаки: $[a_0, x_0]$ либо $[b_0, x_0]$. x_1 Рабочая формула метода:

$$x_n = x_{n-1} - \frac{b_{n-1} - a_{n-1}}{f(b_{n-1}) - f(a_{n-1})} f(a_{n-1})$$

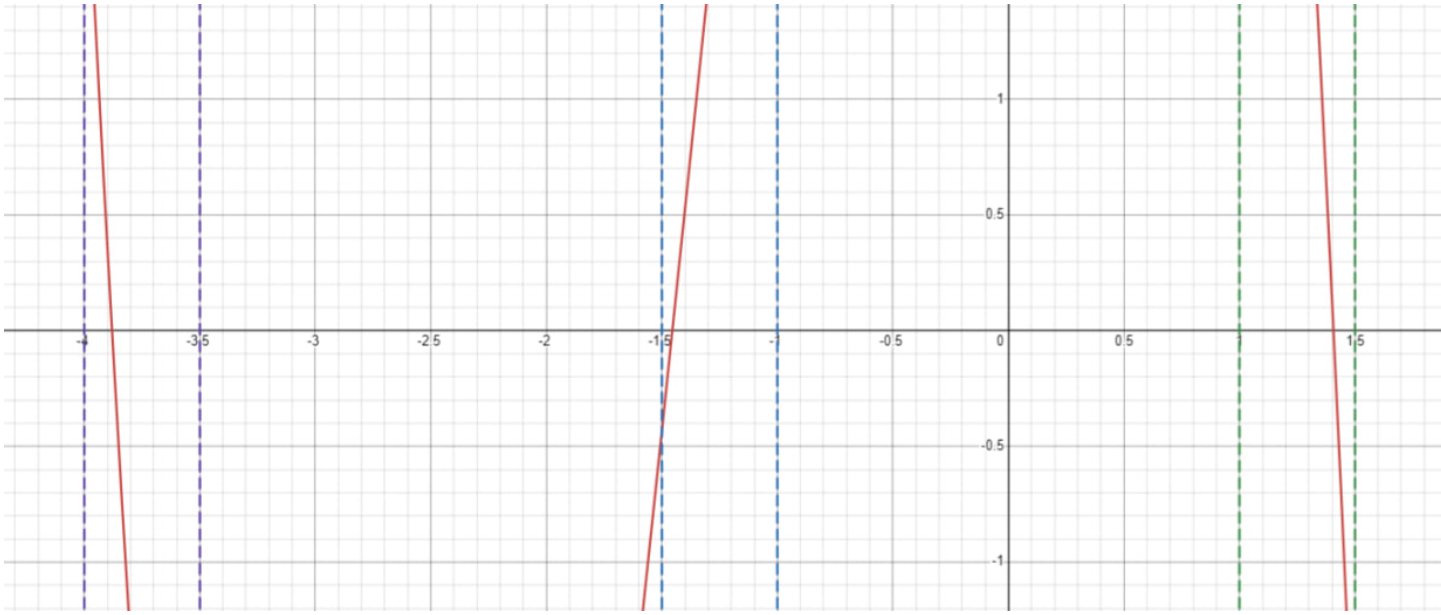
Критерии окончания итерационного процесса: $|x_n - x_{n-1}| \leq \varepsilon$ или $|a_n - b_n| \leq \varepsilon$ или $|f(x_n)| \leq \varepsilon$ Приближенное значение корня: $x^* = x$

5 Пример вывода программы



Пример вывода программы

6 Решение нелинейного уравнения (1 часть)



Графическое отделение корней

№ итерации	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\|a-b\|$
0	-4	-3.5	-3.622	2.27	-5.2725	-3.89	0.238
1	-4	-3.622	-3.86	2.27	-3.89	-0.36	0.02
2	-4	-3.86	-3.88	2.27	-0.36	-0.009	0..

Метод хорд (крайний левый)

№ итерации	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\|a-b\|$
0	-1.5	-1	-1.25	-0.4425	4.34	1.964	0.5
1	-1.5	-1.25	-1.375	-0.4425	1.964	0.757	0.25
2	-1.5	-1.375	-1.4375	-0.4425	0.757	0.155	0.125
3	-1.5	-1.4375	-1.46875	-0.4425	0.155	-0.1444	0.0625
4	-1.46875	-1.4375	-1.453125	-0.1444	0.155	0.0051	0.03125

Метод половинного деления (центральный)

№ итерации	x_i	x_{i+1}	$\psi'(x_i+1)$	$f(x_{i+1})$	$\|x_{i+1}-x_i\|$
0	1	1.296	1.395	2.172	0.296
1	1.296	1.395	1.408	0.24	0.099
2	1.395	1.408	1.411	-0.02	0.016
3	1.408	1.411	1.411	-0.09	0.003

Метод простых итераций (крайний правый)

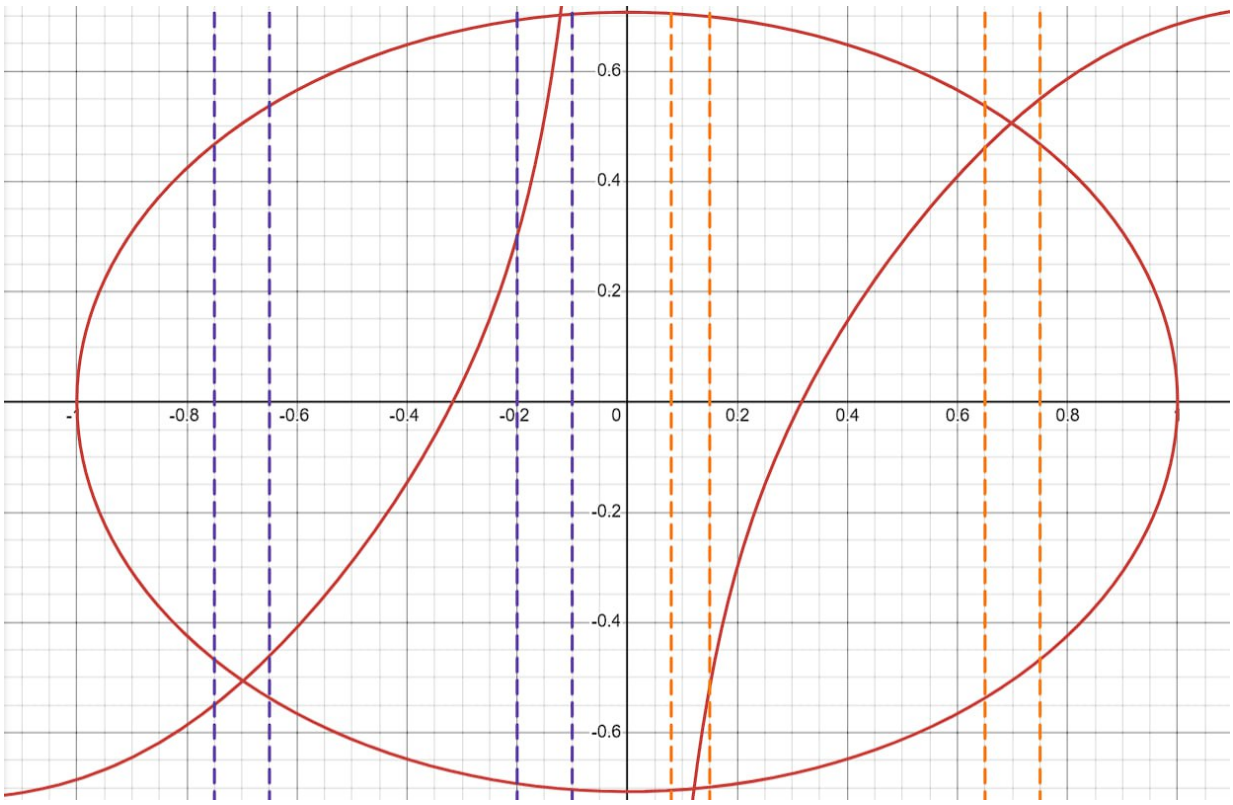


Рис. 1: Графическое отделение корней системы

Правый верхний корень. Начальное приближение, полученное путем графического отделения корней: x0 = 0.7, y0 = 0.5

Шаг 1.

$$\begin{cases} -0.78333\Delta x + 0.86334\Delta y = 0.00633 \\ 1.4\Delta x + 2\Delta y = 0.01 \end{cases}$$

Шаг 2. Решим систему: Δx = -0.00189 и Δy = 0.00694

Шаг 3. Вычислим очередные приближения:

$x_1 = x_0 + \Delta x$ = 0.7 - 0.00189 = 0.69811

$y_1 = y_0 + \Delta y$ = 0.5 - 0.00694 = 0.50633

Шаг 4. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_1$ - $x_0| <= \epsilon$ $|y_1$ - $y_0| <= \epsilon$ |0.69811 - 0.7| $<= \epsilon$ |0.506331 - 0.5| $<= \epsilon$

Правый нижний корень. Начальное приближение, полученное путем графического отделения корней: x0 = 0.1, y0 = -0.7

Шаг 1.

$$\begin{cases} -0.90063\Delta x + 0.10009\Delta y = -0.02001 \\ 0.2\Delta x - 2.8\Delta y = 0.01 \end{cases}$$

Шаг 2. Решим систему: Δx = 0.02199 и Δy = -0.002

Шаг 3. Вычислим очередные приближения:

$x_1 = x_0 + \Delta x$ = 0.1 - 0.02199 = 0.12199

$y_1 = y_0 + \Delta y$ = -0.7 - 0.002 = -0.702

Шаг 4. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_1$ - $x_0| <= \epsilon$ $|y_1$ - $y_0| <= \epsilon$

$|0.12199$ - $0.1| > \epsilon$ $|-0.702$ - $0.7| <= \epsilon$

Шаг 5. Подставляем очередные приближения в систему:

$$\begin{cases} -0.94613\Delta x + 0.12202\Delta y = 0.00052 \\ 0.24399\Delta x - 2.808\Delta y = -0.00049 \end{cases}$$

Шаг 6. Решим систему: Δx = -0.00053 и Δy = 0.00013

Шаг 7. Вычислим очередные приближения:

$x_2 = x_1 + \Delta x$ = 0.12199 - 0.00053 = 0.12146

$y_2 = y_1 + \Delta y$ = -0.702 - 0.00013 = -0.70187 Шаг 8. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_2$ - $x_1| <= \epsilon$ $|y_2$ - $y_1| <= \epsilon$

$|0.12146$ - $0.12199| <= \epsilon$ $|-0.70187$ - $(-0.702)| <= \epsilon$

Левый нижний корень. Начальное приближение, полученное путем графического отделения корней: x0 = -0.8, y0 = -0.6

Шаг 1.

$$\begin{cases} 0.74245\Delta x - 1.1434\Delta y = -0.01517 \\ -1.6\Delta x - 2.4\Delta y = -0.36 \end{cases}$$

Шаг 2. Решим систему: Δx = 0.1039 и Δy = -0.08073

Шаг 3. Вычислим очередные приближения:

$x_1 = x_0 + \Delta x$ = -0.8 + 0.1039 = -0.6961

$y_1 = y_0 + \Delta y$ = -0.6 + 0.08073 = -0.51927

Шаг 4. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_1$ - $x_0| <= \epsilon$ $|y_1$ - $y_0| <= \epsilon$

$|-0.6961$ - $(-0.8)| > \epsilon$ $|-0.51927$ - $0.6| > \epsilon$

Шаг 5. Подставляем очередные приближения в систему:

$$\begin{cases} 0.74453\Delta x - 0.86823\Delta y = -0.01272 \\ -1.3922\Delta x - 2.07707\Delta y = -0.02383 \end{cases}$$

Шаг 6. Решим систему: Δx = -0.00208 и Δy = 0.01287

Шаг 7. Вычислим очередные приближения:

$x_2 = x_1 + \Delta x$ = -0.6961 -0.00208 = 0.69818

$y_2 = y_1 + \Delta y$ = 0.51927 + 0.01287 = -0.5064 Шаг 8. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_2$ - $x_1| <= \epsilon$ $|y_2$ - $y_1| <= \epsilon$

$|0.69818$ - $(-0.6961)| <= \epsilon$ $|(-0.5064)$ - $(0.51927)| > \epsilon$

Шаг 9. Подставляем очередные приближения в систему:

$$\begin{cases} 0.76962\Delta x - 0.86407\Delta y = 0 \\ -1.39635\Delta x - 2.02561\Delta y = -0.00034 \end{cases}$$

Шаг 10. Решим систему: Δx = 0.0001 и Δy = 0.00009

Шаг 11. Вычислим очередные приближения:

$x_3 = x_2 + \Delta x$ = -0.69818 -0.0001 = -0.69807

$y_3 = y_2 + \Delta y$ = -0.5064 + 0.00009 = -0.50631 Шаг 12. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_3$ - $x_2| <= \epsilon$ $|y_3$ - $y_2| <= \epsilon$

|-0.69807 - (-0.69818)| <= ϵ |(-0.50631) - (-0.5064)| <= ϵ

Левый верхний корень. Начальное приближение, полученное путем графического отделения корней: x0 = -0.1, y0 = 0.7

Шаг 1.

$$\begin{cases} 0.90063\Delta x - 0.10009\Delta y = -0.02001 \\ -0.2\Delta x + 2.8\Delta y = 0.01 \end{cases}$$

Шаг 2. Решим систему: Δx = -0.02199 и Δy = 0.002

Шаг 3. Вычислим очередные приближения:

$x_1 = x_0 + \Delta x$ = 0.1 - 0.02199 = - 0.12199

$y_1 = y_0 + \Delta y$ = -0.7 - 0.002 = 0.702

Шаг 4. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_1$ - $x_0| <= \epsilon$ $|y_1$ - $y_0| <= \epsilon$

|0.12199 - 0.1| > ϵ |-0.702 - 0.7| <= ϵ

Шаг 5. Подставляем очередные приближения в систему:

$$\begin{cases} -0.94613\Delta x + 0.12202\Delta y = 0.00052 \\ 0.24399\Delta x - 2.808\Delta y = -0.00049 \end{cases}$$

Шаг 6. Решим систему: Δx = -0.00053 и Δy = -0.00013

Шаг 7. Вычислим очередные приближения:

$x_2 = x_1 + \Delta x$ = - 0.12146 - 0.00053 = -0.12146

$y_2 = y_1 + \Delta y$ = -0.702 - 0.00013 = 0.70187 Шаг 8. Проверяем критерий окончания итерационного процесса ($\epsilon = 0.01$):

$|x_2$ - $x_1| <= \epsilon$ $|y_2$ - $y_1| <= \epsilon$

|-0.12146 - (-0.12199)| <= ϵ |0.70187 - 0.702| <= ϵ

7 Вывод

В ходе выполнения лабораторной работы были изучены численные методы решения систем нелинейных алгебраических уравнений методами Ньютона, простых итераций и методом хорд.

Метод Ньютона:

Принцип работы: использует касательную для нахождения корня. Скорость сходимости: быстрая, квадратичная.

Метод хорд:

Идея метода: функция f(x) на отрезке [a, b] заменяется касательной и в качестве приближенного значения корня принимается точка пересечения касательной с осью абсцисс

Метод половинного деления:

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

Принцип работы: делит интервал пополам, ищет интервал смены знака.
Скорость сходимости: линейная, но гарантированная.
Пример использования: решение уравнений, где функция меняет знак на интервале.

Метод простой итерации:
Принцип работы: преобразует уравнение $f(x) = 0$ к эквивалентному виду $x = g(x)$ и итерирует.
Скорость сходимости: зависит от выбора функции $g(x)$ и начального приближения.