Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники

Вариант №9
Лабораторная работа №2
по теме
Нелинейные уравнения и система нелинейных уравнений
по дисциплине
Вычислительная математика

Выполнил Студент группы P3212
Кобелев Р.П.
к. т. н. Преподаватель:
Наумова Н.А.

г. Санкт-Петербург
2024г.

Содержание

1 Цель работы **2**

2 Порядок выполнения работ **2**

3 Вычислительная реализация задачи **3**
3.1 Левый корень методом половинного деления 3
3.2 Левый корень методом простых итераций 4
3.3 Центральный корень методом секущих 5
3.4 Система нелинейных уравнений методом Ньютона 5

4 Программная реализация задачи **7**
4.1 Метод хорд 7
4.1.1 Блок-схема 7
4.1.2 Программная реализация 7
4.1.3 Результаты выполнения программы при различных исходных данных 8
4.2 Метод Ньютона 9
4.2.1 Блок-схема 9
4.2.2 Программная реализация 10
4.2.3 Результаты выполнения программы при различных исходных данных 10
4.3 Метод Простых итераций 11
4.3.1 Блок-схема 11
4.3.2 Программная реализация 11
4.3.3 Результаты выполнения программы при различных исходных данных 13
4.4 Метод Простых итераций для системы 13
4.4.1 Блок-схема 13
4.4.2 Программная реализация 14
4.4.3 Результаты выполнения программы при различных исходных данных 15

5 Github **16**

6 Вывод **16**

1 Цель работы

Разработать приложение для решения нелинейных уравнений методом хорд, Ньютона и простых итераций и ситсем нелинейных уравнений методом простых итераций

2 Порядок выполнения работ

1. Решение нелинейного уравнения
   - Правый корень уточняется методом половинного деления
   - Левый корень уточняется методом простых итераций
   - Центральный корень уточняется методом секущих
2. Решение системы нелинейного уравнения методом Ньютона
3. Программно реализовать 4 метода
   - Метод хорд
   - Метод Ньютона
   - Метод простых итераций
   - Метод простых итераций для систем

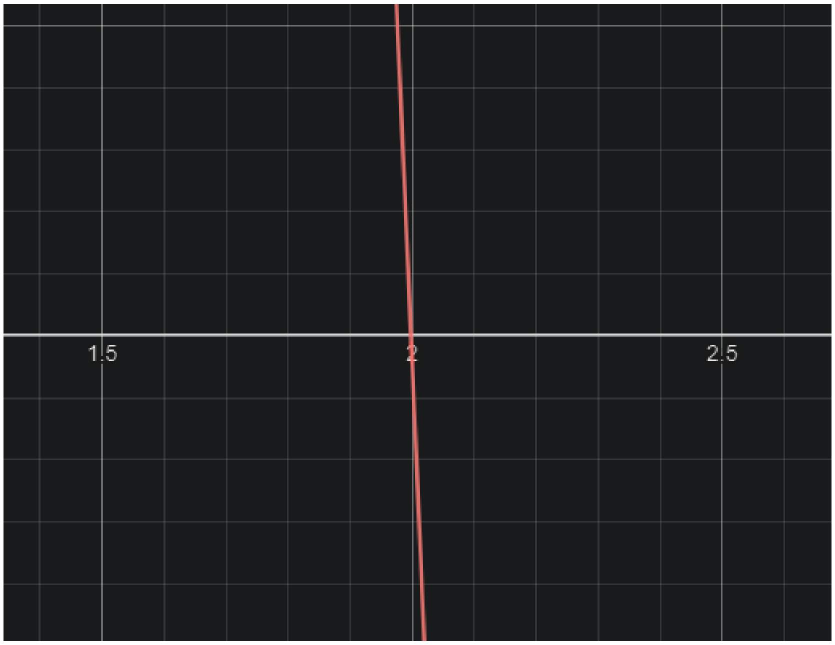
3 Вычислительная реализация задачи

3.1 Левый корень методом половинного деления

Рабочая формула метода:

$$x_i = \frac{a_i + b_i}{2}$$

$$f(x) = -1.8x^3 - 2.94x^2 + 10.37x + 5.38$$



Интервал изоляции корня: [1.5, 2.5]

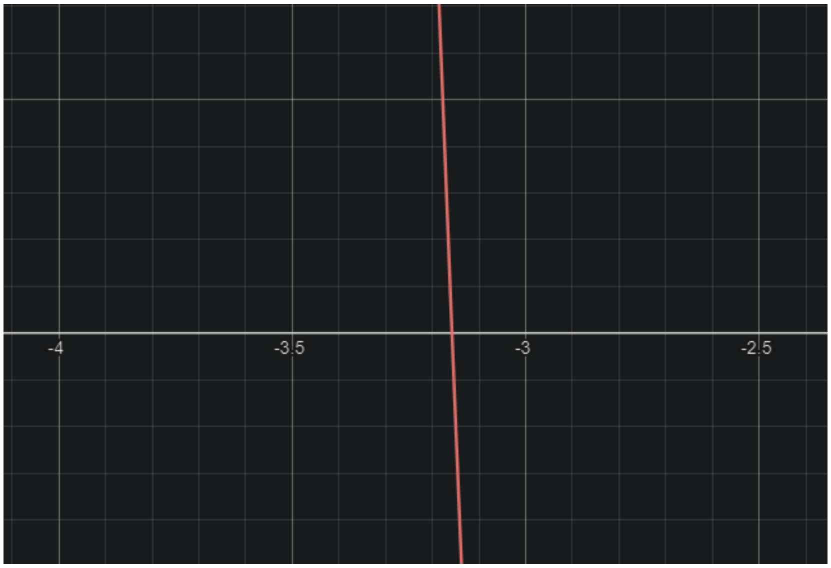
№ итерации	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\|a-b\|$
0	1.5	2.5	2	8.24499	-15.195	-0.04	1
1	1.5	2	1.75	8.24499	-0.04	4.87687	0.5
2	1.75	2	1.875	4.87687	-0.04	2.62257	0.25
3	1.875	2	1.9375	2.62257	-0.04	1.34364	0.125
4	1.9375	2	1.96875	1.34364	-0.04	0.66507	0.0625
5	1.96875	2	1.98437	0.66507	-0.04	0.31587	0.03125
6	1.984375	2	1.99218	0.31587	-0.04	0.13877	0.015625

3.2 Левый корень методом простых итераций

Рабочая формула метода:

$$x_i = \varphi(x_i)$$

$$f(x) = -1.8x^3 - 2.94x^2 + 10.37x + 5.38$$



$$x = \varphi(x) = -\frac{5.38}{10.37} + \frac{2.94}{10.37}x^2 + \frac{1.8}{10.37}x^3$$

$$\varphi'(x) = \frac{5.88}{10.37} + \frac{5.4}{10.37}x^2$$

Интервал изоляции корня: [-3.5, -3]
Проверка достаточного условия схожимости метода:

$$max(\varphi'(x)) = \varphi'(-3.5) = 6.946 < 1$$

Пойдём по другому способу, где применяется приём введения параметра λ:

$$f(x) = -1.8x^3 - 2.94x^2 + 10.37x + 5.38$$

$$\lambda f(x) = 0 \ (\lambda! = 0)$$

$$\phi(x) \ = x + \lambda f(x)$$

$$\phi'(x) \ = 1 + \lambda f'(x)$$

$$f'(x) = -5.4x^2 - 5.88x + 10.37$$

$$f'(-3.5) = -5.4 \cdot (-3.5)^2 - 5.88 \cdot (-3.5) + 10.3 = -35.27$$

$$f'(-3) = -5.4 \cdot (-3)^2 - 5.88 \cdot (-3) + 10.3 = -20.66$$

Так как $f'[a, b] < 0$, то рассматриваем:

$$\lambda = \frac{1}{max|f'(x)|} = -\frac{1}{20.66} = -0.048$$

Подставим:

$$\phi(x) = x - 0.02835 \cdot (-1.8x^3 - 2.94x^2 + 10.37x + 5.38) =$$

$$= -0.25824 + 0.50224x + 0.14112x^2 + 0.0864x^3$$

$$\phi'(x) = 0.50224 + 0.28224x + 0.2592x^2$$

Проверим точки:

$$\phi'(-3.5) = 0.50224 + 0.28224 \cdot (-3.5) + 0.2592 \cdot (-3.5)^2 = 2.6896 > 1$$

$$\phi'(-3) = 0.50224 + 0.28224 \cdot (-3) + 0.2592 \cdot (-3)^2 = 1.988 > 1$$

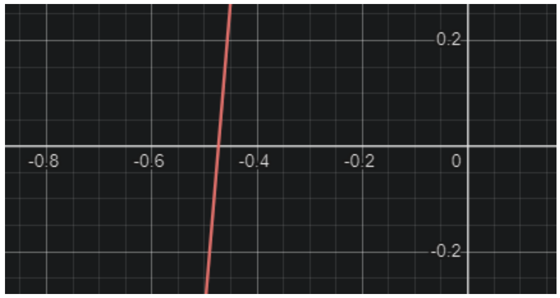
Функция не сходится

3.3 Центральный корень методом секущих

Рабочая формула метода:

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$$

$$f(x) = -1.8x^3 - 2.94x^2 + 10.37x + 5.38$$

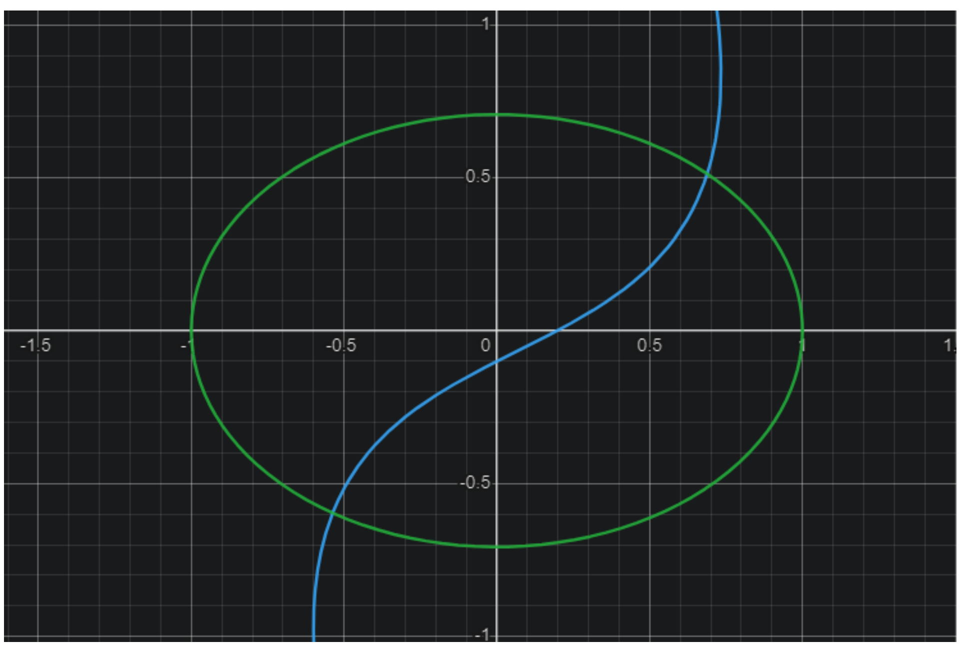


Интервал изоляции корня: [-0.6, 0]

№ итерации	x_{i-1}	x_i	x_{i+1}	$f(x_{i+1})$	$\lvert x_{i+1} - x_i \rvert$
0	-0.6	0	-0.46839	0.06268	0.46839
1	0	-0.46839	-0.47391	-0.00325	0.00552

3.4 Система нелинейных уравнений методом Ньютона

$$\begin{cases} \sin x + y = 1.5x - 0.1 \\ x^2 + 2y^2 = 1 \end{cases}$$



$$\begin{cases} sin(x+y) - 1.5x + 0.1 = 0 \\ x^2 + 2y^2 - 1 = 0 \end{cases}$$

Построим матрицу Якоби:

$$\frac{\partial f}{\partial x} = cos(x+y) - 1.5$$

$$\frac{\partial f}{\partial y} = cos(x+y)$$

$$\frac{\partial g}{\partial x} = 2x$$

$$\frac{\partial g}{\partial y} = 4y$$

Получим матрицу Якоби:

$$\begin{vmatrix} cos(x+y) - 1.5 & cos(x+y) \\ 2x & 4y \end{vmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -\begin{pmatrix} sin(x+y) - 1.5x + 0.1 \\ x^2 + 2y^2 - 1 \end{pmatrix}$$

Возьмём точку $x_0 = 0.8;\ y_0 = 0.6$

$$\begin{cases} -1.33\Delta x + 0.16996\Delta y = 0.11455 \\ 1.6\Delta x + 2.4\Delta y = -0.36 \end{cases}$$

Решения:

$$\Delta x = -0.097; \quad \Delta y = -0.0853$$

Проверка:

$$x_1 = x_0 + \Delta x = 0.8 - 0.097 = 0.703$$

$$y_1 = y_0 + \Delta y = 0.6 - 0.0853 = 0.5147$$

Продолжим вычисление при новом приближении $x_1 = 0.703;\ y_1 = 0.5147$

$$\begin{cases} -1.1542\Delta x + 0.3458\Delta y = -0.00135 \\ 2.812\Delta x + 2.0588\Delta y = -0.03893 \end{cases}$$

Решения:

$$\Delta x = -0.00319; \quad \Delta y = -0.01455;$$

Проверка:

$$x_2 = x_1 + \Delta x = 0.703 - 0.00319 = 0.6998$$

$$y_2 = y_1 + \Delta y = 0.5147 - 0.01455 = 0.5$$

Продолжим вычисление при новом приближении $x_2 = 0.6998;\ y_2 = 0.5$

$$\begin{cases} -1.13746\Delta x + 0.362544\Delta y = -0.00072 \\ 1.3996\Delta x + 2\Delta y = -0.00103 \end{cases}$$

Решения:

$$\Delta x = -0.00038 < \epsilon; \quad \Delta y = -0.000783 < \epsilon;$$

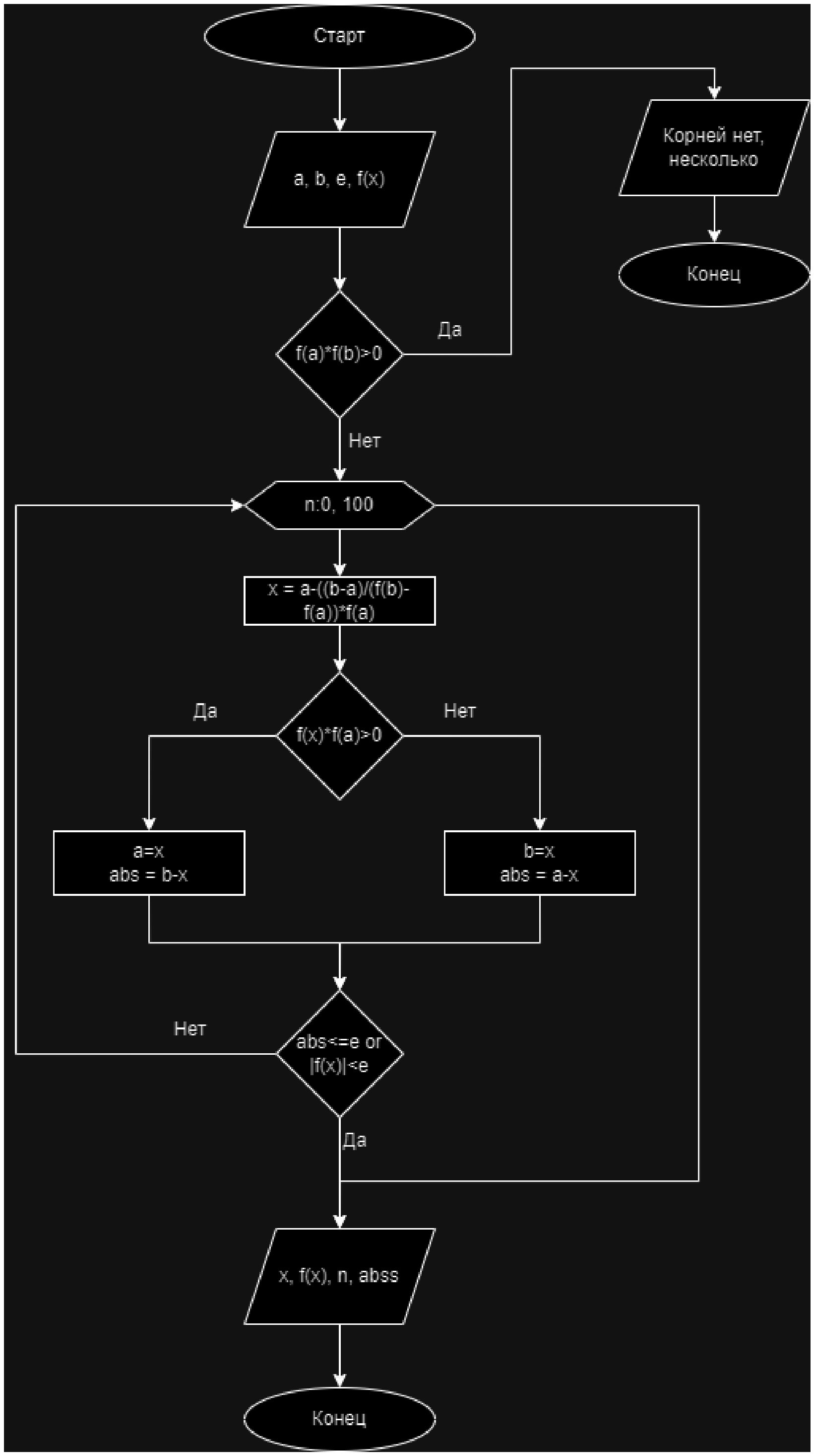
$$x_3 = x_2 + \Delta x = 0.6998 - 0.00038 = 0.69942$$

$$y_3 = y_2 + \Delta y = 0.5 - 0.000783 = 0.49922$$

4 Программная реализация задачи

4.1 Метод хорд

4.1.1 Блок-схема



4.1.2 Программная реализация

ch.py

```
def solve(self, a, b, e):
```

```
    self.data.append(
        ("№ итерации", "a", "b", "x", "f(a)", "f(b)", "f(x)", "|x_i+1-x_i|")
    )
    if self.equation.get_value(a) * self.equation.get_value(b) > 0:
        return ["На данномучасткенеткорнейнесколько\ корней"]
    while True:
        x = a - (
            (b - a) / (self.equation.get_value(b) - self.equation.get_value(a))
        ) * self.equation.get_value(a)
        draw_point(x, 0, self.n, "b", "x")
        draw_chord(a, self.equation.get_value(a), b, self.equation.get_value(b))
        if self.equation.get_value(x) * self.equation.get_value(a) > 0:
            self.print_line(
                self.n,
                a,
                b,
                x,
                self.equation.get_value(a),
                self.equation.get_value(b),
                self.equation.get_value(x),
                b - x,
            )
            a = x
            abss = b - x
        else:
            self.print_line(
                self.n,
                a,
                b,
                x,
                self.equation.get_value(a),
                self.equation.get_value(b),
                self.equation.get_value(x),
                a - x,
            )
            b = x
            abss = a - x
        if abs(abss) <= e or abs(self.equation.get_value(x)) < e:
            break
        self.increment()

    return [
        self.data,
        "\nx = %6.5f f(x) = %4.5f Количествоитераций: %2d"
        % (int_r(x), int_r(self.equation.get_value(x)), self.n + 1),
    ]
```

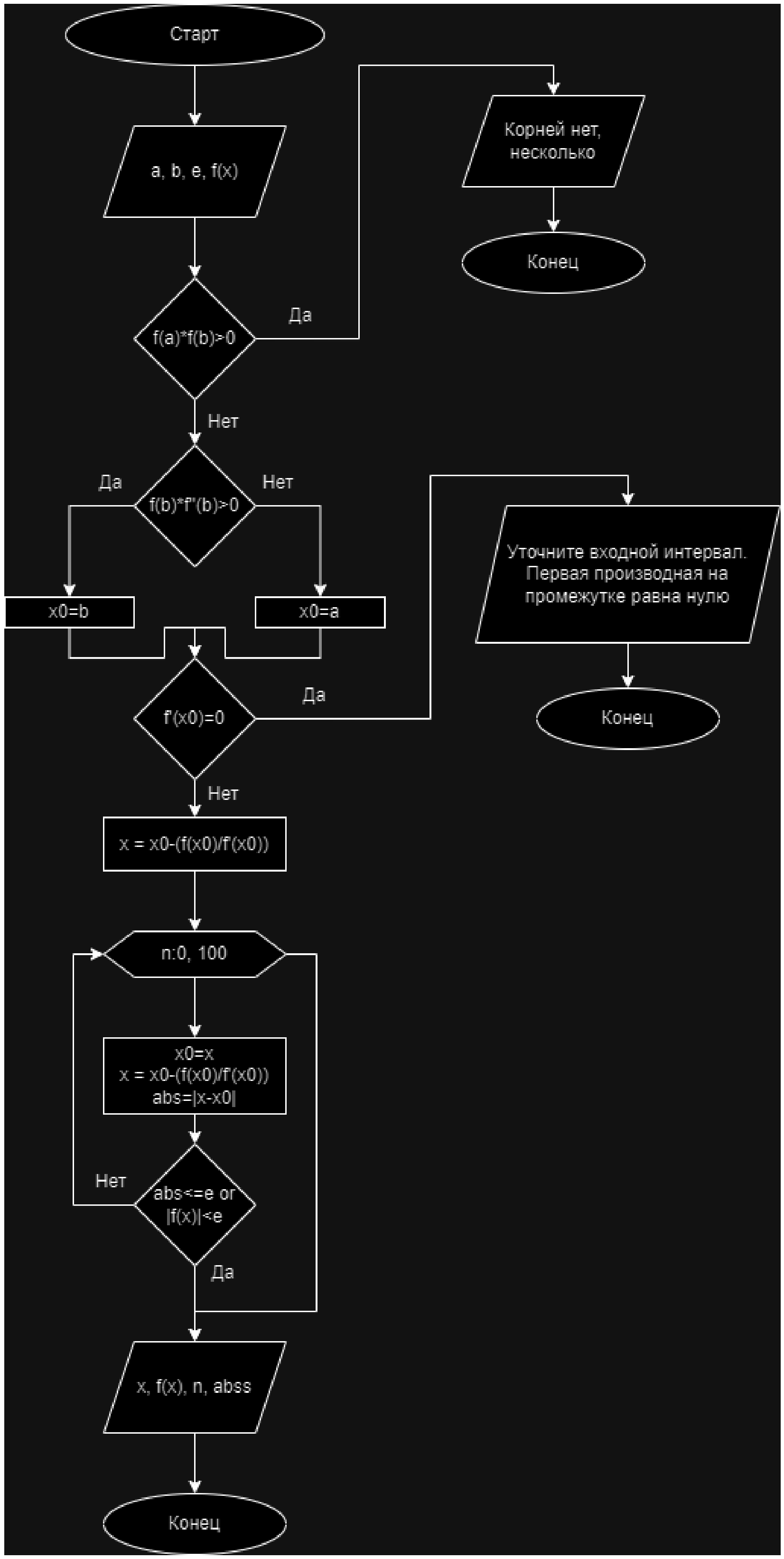
4.1.3 Результаты выполнения программы при различных исходных данных

На данном участке нет корней
несколько корней

x = -1.62869 f(x) = 0.00933 Количество итераций: 6

4.2 Метод Ньютона

4.2.1 Блок-схема



4.2.2 Программная реализация

newt.py

```
def solve(self, a, b, e):
    self.data.append(
        ("№ итерации", "x_i", "f(x_i)", "f'(x_i)", "x_i+1", "|x_i+1 - x_i|")
    )
    if self.equation.get_value(a) * self.equation.get_value(b) > 0:
        return ["На данномучасткенеткорнейнесколько\ корней"]
    if self.equation.get_value(b) * self.equation.second_derivative(b) > 0:
        x0 = b
    else:
        x0 = a
    try:
        x = x0 - (self.equation.get_value(x0) / self.equation.first_derivative(x0))
    except ZeroDivisionError:
        return [
            "Уточните входнойинтервал. Перваяпроизводнаянапромежуткеравнанулю"
        ]
    self.print_line(self.n, x0, x)
    draw_point(x, 0, self.n, "b", "x")
    draw_tangent(x0, self.equation.get_value(x0), x)
    while True:
        self.increment()
        x0 = x
        x = x0 - (self.equation.get_value(x0) / self.equation.first_derivative(x0))
        draw_point(x, 0, self.n, "b", "x")
        draw_tangent(x0, self.equation.get_value(x0), x)
        self.print_line(self.n, x0, x)
        if abs(x - x0) ⩽ e and abs(self.equation.get_value(x)) < e:
            draw_point(x, 0, self.n, "r", "x")
            break

    return [
        self.data,
        "x = %6.5f f(x) = %4.5f Количествоитераций: %2d"
        % (int_r(x), int_r(self.equation.get_value(x)), self.n + 1),
    ]
```

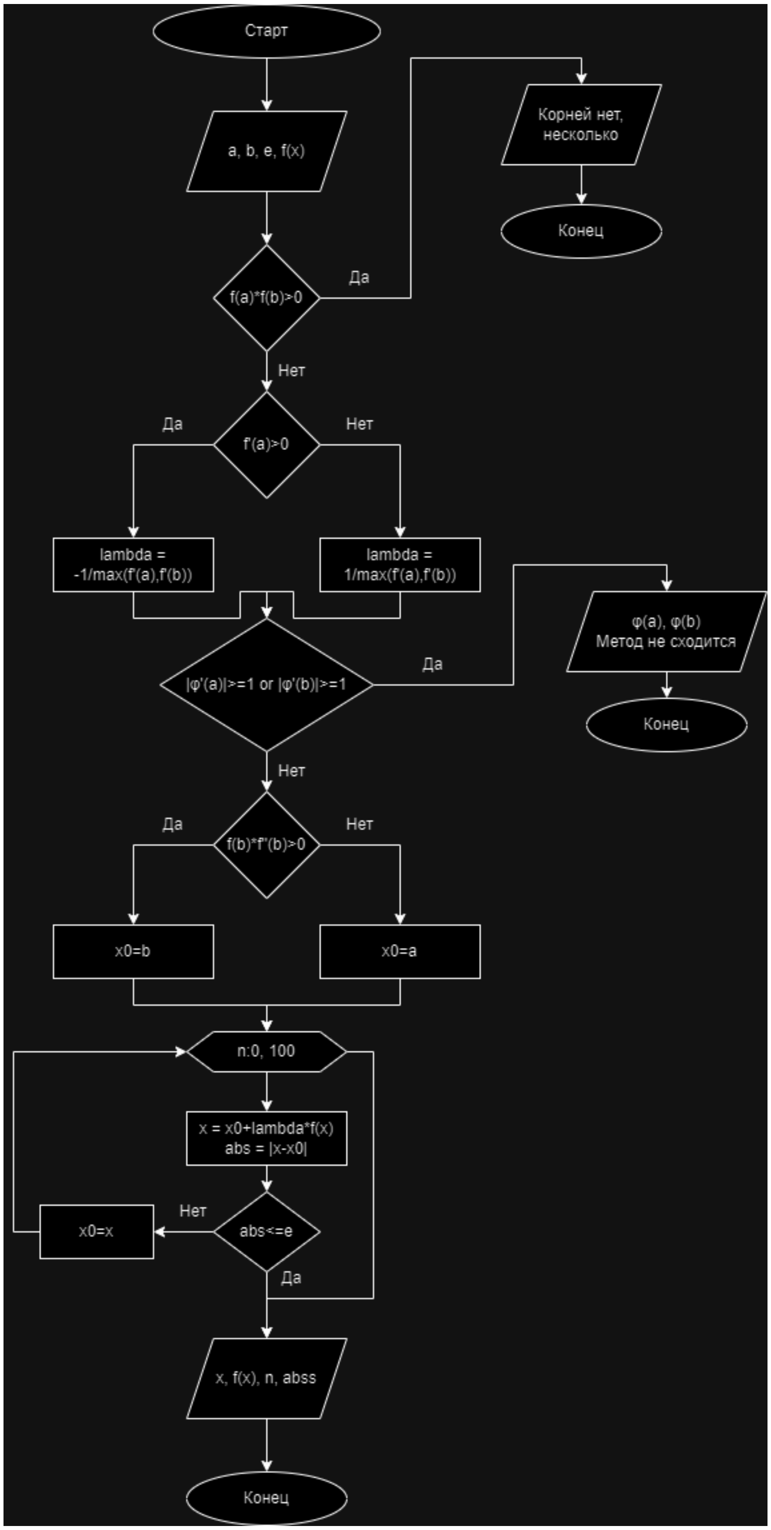
4.2.3 Результаты выполнения программы при различных исходных данных

На данном участке нет корней
несколько корней

x = -1.62919 f(x) = -0.00071 Количество итераций: 3

4.3 Метод Простых итераций

4.3.1 Блок-схема



4.3.2 Программная реализация

it.py

```
def solve(self, left, right, estimate):
```

```
if self.equation.get_value(left) * self.equation.get_value(right) > 0:
    return ["На данномучасткенеткорнейнесколько\ корней"]

self.data.append(
    ("№ итерации", "x_i", "x_i+1", "φ(x_i+1)", "f(x_i+1)", "|x_i+1 - x_i|")
)
if self.equation.first_derivative(left) > 0:
    parameter_lambda = -1 / max(
        self.equation.first_derivative(left),
        self.equation.first_derivative(right),
    )
else:
    parameter_lambda = 1 / max(
        self.equation.first_derivative(left),
        self.equation.first_derivative(right),
    )
if (
    abs(self.new_function_first_derivative(left, parameter_lambda)) ⩾ 1
    or abs(self.new_function_first_derivative(right, parameter_lambda)) ⩾ 1
):
    self.output += "φ(a) = %9.5f\n" % self.new_function_first_derivative(
        left, parameter_lambda
    )
    self.output += "φ(b) = %9.5f\n" % self.new_function_first_derivative(
        right, parameter_lambda
    )
    self.output += "Метод несходится. Нарушенодостаточноеусловиесходимостиметода .\Уменьшитеn рассматриваемыйпромежуток\n"
    return [self.output]

if self.equation.get_value(right) * self.equation.second_derivative(right) > 0:
    x0 = right
else:
    x0 = left
self.output += "φ'(x) = %9.5f\n" % self.new_function_first_derivative(
    x0, parameter_lambda
)

self.draw_new_function(left, right, parameter_lambda)
while True:
    x = self.new_function(x0, parameter_lambda)
    draw_point(x0, self.new_function(x0, parameter_lambda), self.n, "g", "φ")
    draw_point(x0, self.equation.get_value(x0), self.n, "b", "x")
    self.print_line(self.n, x0, x, parameter_lambda)
    if abs(x - x0) ⩽ estimate:
        self.increment()
        draw_point(x, self.equation.get_value(x), self.n, "r", "x")
        break
    x0 = x
    self.increment()
self.output += "\Результатn выполнения: \n"
self.output += "x = %6.5f f(x) = %4.5f Количествоитераций: %2d\n" % (
    int_r(x),
    int_r(self.equation.get_value(x)),
    self.n,
)
```

```python
    return [self.data, self.output]
```

4.3.3 Результаты выполнения программы при различных исходных данных

На данном участке нет корней
несколько корней

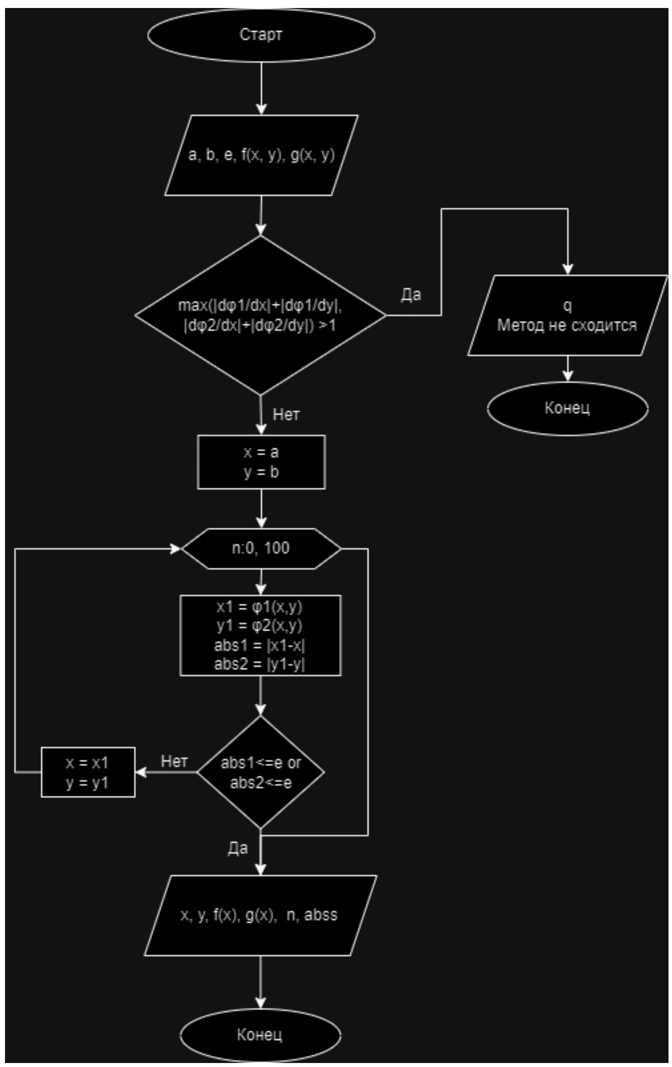
φ'(x) = 0.29790

Результат выполнения:
x = 0.44868 f(x) = -0.01401 Количество итераций: 4

φ(a) = 1.415122
φ(b) = 2.51165
Метод не сходится. Нарушено достаточное условие сходимости метода.
Уменьшите рассматриваемый промежуток

4.4 Метод Простых итераций для системы

4.4.1 Блок-схема



4.4.2 Программная реализация

sysit.py

```
def solve(self, a, b, estimate):
    self.data.append(
        (
            "№ итерации",
            "x_1",
            "x_2",
            "|x1_i+1 - x1_i|",
            "|x2_i+1 - x2_i|",
        )
    )
    self.output += "Проверка сходимостиметода\n"
    q = max(
        max(
            abs(self.equation1.first_derivative(a, b, 1)),
            abs(self.equation1.first_derivative(a, 0, 1)),
            abs(self.equation1.first_derivative(0, b, 1)),
            abs(self.equation1.first_derivative(0, 0, 1)),
        )
        + max(
            abs(self.equation1.first_derivative(a, b, 2)),
            abs(self.equation1.first_derivative(a, 0, 2)),
            abs(self.equation1.first_derivative(0, b, 2)),
            abs(self.equation1.first_derivative(0, 0, 2)),
        ),
        max(
            abs(self.equation2.first_derivative(a, b, 1)),
            abs(self.equation2.first_derivative(a, 0, 1)),
            abs(self.equation2.first_derivative(0, b, 1)),
            abs(self.equation2.first_derivative(0, 0, 1)),
        )
        + max(
            abs(self.equation2.first_derivative(a, b, 2)),
            abs(self.equation2.first_derivative(a, 0, 2)),
            abs(self.equation2.first_derivative(0, b, 2)),
            abs(self.equation2.first_derivative(0, 0, 2)),
        ),
    )
    self.output += "q = %9.5f\n" % q
    if q > 1:
        self.output += "Процесс несходится\n"
        return [self.output]
    else:
        self.output += "Процесс сходится\n\n"
    self.output += "\Выборn начальногоприближения\n"
    x = a
    y = b
    while True:
        x1 = self.equation1.get_value(x, y, 1)
        y1 = self.equation2.get_value(x, y, 1)
        e1 = abs(x1 - x)
        e2 = abs(y1 - y)
        self.print_line(self.n, x1, y1, e1, e2)
        if e1 ⩽ estimate and e2 ⩽ estimate:
```

```
            self.increment()
            break
        x = x1
        y = y1
        self.increment()
    self.output += ”\Результатn выполнения: \n”
    self.output += (
        ”x1 = %6.5f x2 = %6.5f f1(x1, x2) = %4.5f f2(x1, x2) = %4.5f Количествоитераций: %2d”
        % (
            int_r(x1),
            int_r(y1),
            int_r(self.equation1.get_value(x, y)),
            int_r(self.equation2.get_value(x, y)),
            self.n,
        )
    )
    return [self.data, self.output]
```

4.4.3 Результаты выполнения программы при различных исходных данных

Проверка сходимости метода q = 2.85780 Процесс не сходится

Проверка сходимости метода
q = 0.59847
Процесс сходится

Выбор начального приближения

Результат выполнения:
x1 = -0.95267 x2 = 0.09737 f1(x1, x2) = -0.01543 f2(x1, x2) = -0.00257
Количество итераций: 8

5 Github

Ссылка на GitHub

6 Вывод

В этой работе я ознакомился с различными методами решения нелинейного уравнения, а также систем нелинейных уравнений. Некоторые я реализовал в программе, а некоторые использовал для решения вручную.