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

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

По Вычислительной математике

Вариант №10

Выполнил:

Ступин Тимур Русланович

Группа № P3108

Поток № 1.3

Преподаватель:

Санкт-Петербург 2025

Содержание

1 Цель работы

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

2 Порядок выполнения работы

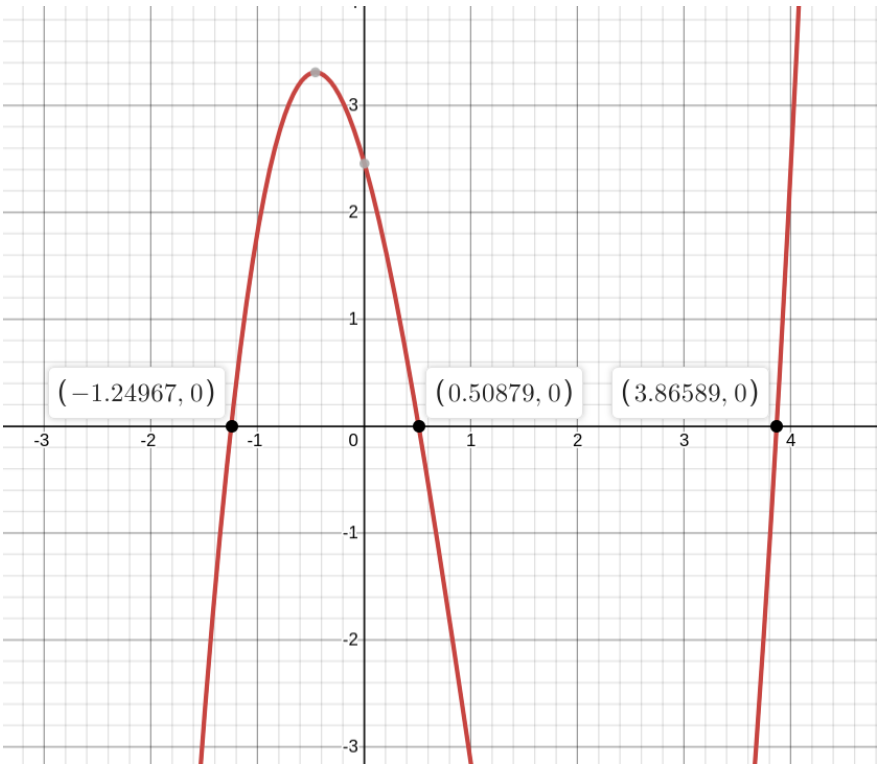
2.1 Вычислительная реализация задачи

2.1.1 Решение нелинейного уравнения

Уравнение в соответствии с вариантом:

$$x^3 - 3.125x^2 - 3.5x + 2.458$$

Отделение корней уравнения графически:



Определение интервалов изоляции корней.

Для нахождения интервалов изоляции корней используем табличный способ.

Протабулируем значения функции на промежутке от -5 до 5 с шагом 0.5:

x	$f(x)$
-5.000	-183.167
-4.500	-136.198
-4.000	-97.542
-3.500	-66.448
-3.000	-42.167
-2.500	-23.948
-2.000	-11.042
-1.500	-2.698
-1.000	1.833
-0.500	3.302
0.000	2.458
0.500	0.052
1.000	-3.167
1.500	-6.448
2.000	-9.042
2.500	-10.198
3.000	-9.167
3.500	-5.198
4.000	2.458
4.500	14.552
5.000	31.833

Получаем три интервала изоляции корней: $(-1.5; -1.0)$, $(-0.5; 1.0)$ и $(3.5; 4.0)$.

Для уточнения крайнего левого корня используем метод половинного деления. Рабочая формула метода:

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

№ шага	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\lvert a - b \rvert$
1	-1.500	-1.000	-1.250	-2.698	1.833	-0.003	0.500
2	-1.250	-1.000	-1.125	-0.003	1.833	1.017	0.250
3	-1.250	-1.125	-1.188	-0.003	1.017	0.533	0.125
4	-1.250	-1.188	-1.219	-0.003	0.533	0.272	0.062

5	-1.250	-1.219	-1.234	-0.003	0.272	0.136	0.031
6	-1.250	-1.234	-1.242	-0.003	0.136	0.067	0.016
7	-1.250	-1.242	-1.246	-0.003	0.067	0.032	0.008

Для уточнения центрального корня используем метод простой итерации.

Проверим условие сходимости метода на выбранном интервале.

Для начала определим значения производной функции $f(x)$ на границах интервала:

$$f(x) = x^3 - 3.125x^2 - 3.5x + 2.458$$

$$f'(x) = 3x^2 - 6.25x - 3.5$$

$$f'(a) = f'(0.5) = -5.875$$

$$f'(b) = f'(1.0) = -6.75$$

Выбрав максимальное по модулю значение:

$$\max(|f'(a)|, |f'(b)|) = 6.75$$

Так как $f'[a, b] < 0$ получаем:

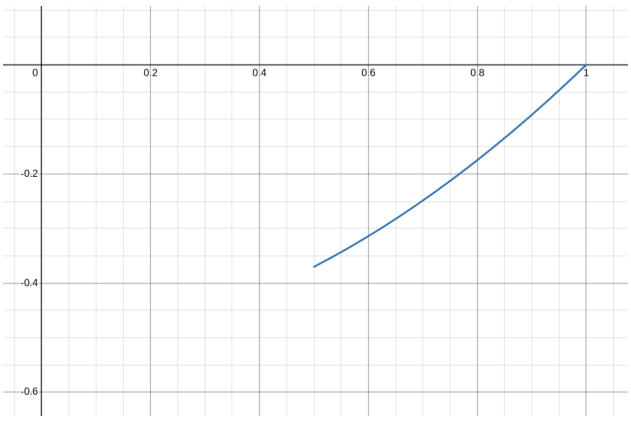
$$\lambda = \frac{1}{\max(|f'(x)|)} = \frac{1}{6.75}$$

Наконец определим функцию $\varphi(x)$ и вычислим $\varphi'(x)$:

$$\varphi(x) = x + \lambda f(x) = x + \frac{x^3 - 3.125x^2 - 3.5x + 2.458}{6.75}$$

$$\varphi'(x) = 1 + \lambda f'(x) = 1 + \frac{3x^2 - 6.25x - 3.5}{6.75}$$

Определим значение q:



$$\varphi'(a) = \varphi'(0.5) = -0.37$$

$$\varphi'(b) = \varphi'(1.0) = 0$$

Так как $0 \leq q \leq 1$ итерационная последовательность сходится.

Так как $0 \leq q \leq 0.5$ критерий окончания итерационного процесса:

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

В качестве начального приближения выберем $x_0 = b = 1.0$. Рабочая формула метода:

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

№ итерации	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
1	1.000	0.591	-0.498	0.409
2	0.591	0.527	-0.109	0.064
3	0.527	0.513	-0.025	0.014
4	0.513	0.510	-0.006	0.003

Для уточнения крайнего правого корня используем метод Ньютона.

Выберем начальное приближение:

$$f''(x) = 6x - 6.25$$

$$f(a) \cdot f''(a) = -76.674 < 0$$

$$f(b) \cdot f''(b) = 43.630 > 0$$

В качестве начального приближения выберем $x_0 = b = 4.0$. Рабочая формула метода:

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

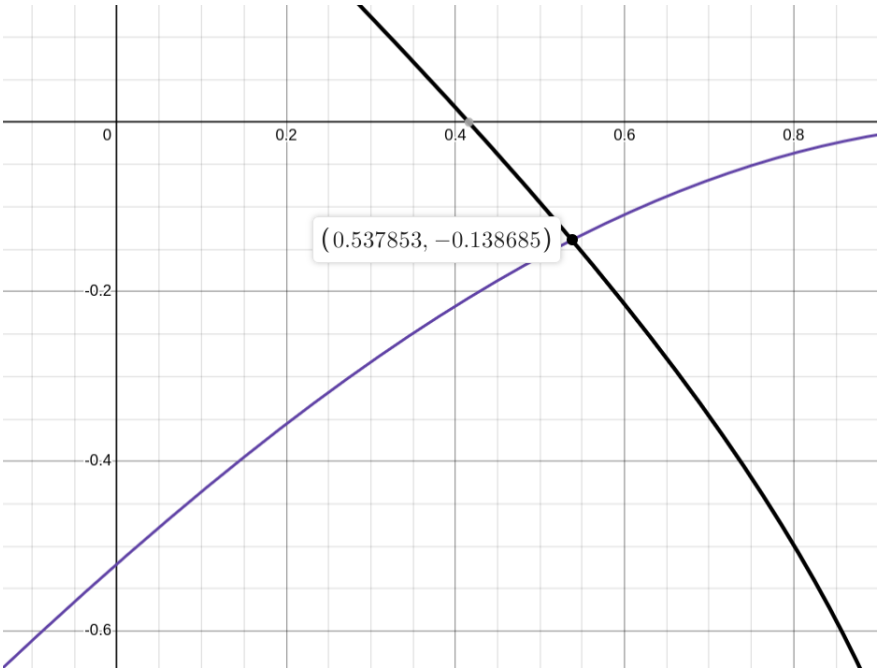
№ итерации	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_{k+1} - x_k \rvert$
1	4.000	2.458	19.500	3.874	0.126
2	3.874	0.139	17.310	3.866	0.008

2.1.2 Решение системы нелинейных уравнений

Исходная система в соответствии с вариантом:

$$\begin{cases} \sin(x + 0.5) - y = 1 \\ \cos(y - 2) + x = 0 \end{cases}$$

Найдем корни системы графически.



Определим область изоляции корней:

$$G = \{0 < x < 1; -1 < y < 0\}$$

Решим систему методом простых итераций.

Перепишем систему в виде:

$$\begin{cases} x = -\cos(y-2) \\ y = \sin(x+0.5) - 1 \end{cases}$$

Проверим выполнение условия сходимости:

$$\max_{[x \in G]} \max_{[i]} \sum_{j=1}^{n} \left| \frac{\partial \varphi_i(X)}{\partial x_j} \right| \leq q < 1$$

В случае данной системы на области G получим:

$$\left| \frac{\partial \varphi_1}{\partial x} \right| + \left| \frac{\partial \varphi_2}{\partial x} \right| = |\cos(x+0.5)| \leq 0.88 < 1$$

$$\left| \frac{\partial \varphi_1}{\partial y} \right| + \left| \frac{\partial \varphi_2}{\partial y} \right| = |\sin(y-2)| \leq 0.91 < 1$$

Условие сходимости выполняется. Рабочая формула метода:

$$\begin{cases} x^{(k+1)} = \varphi_1(x, y) \\ y^{(k+1)} = \varphi_2(x, y) \end{cases}$$

№ итерации	x_k	x_{k+1}	$\lvert x_{k+1} - x_k \rvert$	y_k	y_{k+1}	$\lvert y_{k+1} - y_k \rvert$
1	0.000	0.416	0.416	0.000	-0.521	0.521
2	0.416	0.813	0.397	-0.521	-0.207	0.314
3	0.813	0.594	0.219	-0.207	-0.033	0.174
4	0.594	0.446	0.148	-0.033	-0.112	0.079
5	0.446	0.515	0.069	-0.112	-0.189	0.077
6	0.515	0.580	0.065	-0.189	-0.151	0.038
7	0.580	0.548	0.032	-0.151	-0.118	0.032
8	0.548	0.521	0.027	-0.118	-0.134	0.015
9	0.521	0.534	0.013	-0.134	-0.148	0.014
10	0.534	0.545	0.012	-0.148	-0.141	0.007
11	0.545	0.540	0.006	-0.141	-0.135	0.006

2.2 Программная реализация задачи

2.2.1 Листинг программы

```
def chord_method(equation, a, b, eps):
    f = equation.f
    iterations = 0
    log = []

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

    while True:
        if iterations == MAX_ITERATIONS:
            raise Exception(f'Произведено {MAX_ITERATIONS} итераций, но
            ↪  решение не найдено.')
        iterations += 1

        if f(a) * f(x) <= 0:
            b = x
        else:
            a = x

        next_x = (a * f(b) - b * f(a)) / (f(b) - f(a))
        delta = abs(next_x - x)

        log.append({
            'a': a,
            'b': b,
            'x': x,
            'f(a)': f(a),
            'f(b)': f(b),
            'f(x)': f(x),
            'delta': delta})

        if delta < eps:
            break

        x = next_x

    return Result(x, iterations, log)
```

```
def secant_method(equation, a, b, eps):
    f = equation.f
    f__ = equation.snd_derivative
    iterations = 0
    log = []

    if f__(a) * f__(b) < 0:
        raise Exception(
            'Условия сходимости метода секущих не выполнены! Вторая
            ↪  производная не сохраняет знак на выбранном отрезке')

    x0 = a
    if f(a) * f__(a) > 0:
        x0 = a
    if f(b) * f__(b) > 0:
        x0 = b

    x1 = x0 + eps

    while True:
        if iterations == MAX_ITERATIONS:
            raise Exception(f'Произведено {MAX_ITERATIONS} итераций, но
            ↪  решение не найдено.')
        iterations += 1

        x = x1 - f(x1) * (x1 - x0) / (f(x1) - f(x0))
        delta = abs(x - x1)

        log.append({
            'x_{i-1}': x0,
            'x_i': x1,
            'x_{i+1}': x,
            'f(x_{i+1})': f(x),
            'delta': delta
        })

        if delta < eps:
            break

        x0 = x1
        x1 = x

    return Result(x, iterations, log)
```

```python
def simple_iteration_method(equation, a, b, eps):
    f = equation.f
    f_ = equation.fst_derivative
    iterations = 0
    log = []

    max_derivative = max(abs(f_(a)), abs(f_(b)))
    _lambda = 1 / max_derivative
    if f_(a) > 0: _lambda *= -1

    phi = lambda x: x + _lambda * f(x)

    phi_ = lambda x: derivative(phi, x).df
    q = np.max(abs(phi_(np.linspace(a, b, int(1 / eps)))))
    if q > 1:
        raise Exception(f'Метод не сходится так как значение q >= 1')

    prev_x = a
    while True:
        if iterations == MAX_ITERATIONS:
            raise Exception(f'Произведено {MAX_ITERATIONS} итераций, но
            ↪  решение не найдено.')
        iterations += 1

        x = phi(prev_x)
        delta = abs(x - prev_x)

        log.append({
            'x_i': prev_x,
            'x_{i+1}': x,
            'phi(x_{i+1})': phi(x),
            'f(x_{i+1})': f(x),
            'delta': delta
        })

        if delta <= eps:
            break

        prev_x = x

    return Result(x, iterations, log)
```

```python
def newton_method(system, x0, eps):
    x = x0
    iterations = 0
    log = []
    while True:
        if iterations == MAX_ITERATIONS:
            raise Exception(f'Произведено {MAX_ITERATIONS} итераций, но
            ↪  решение не найдено.')
        iterations += 1

        jcb = system.get_jacobi(x)
        b = system.get_value(x)
        try:
            dx = np.linalg.solve(np.array(jcb), -1 * np.array(b))
        except np.linalg.LinAlgError:
            raise Exception('Не удалось применить метод, промежуточная
            ↪  система не имеет решений!')
        nx = x + dx

        log.append({
            'x_i': x,
            'x_{i+1}': nx.tolist(),
            'dx': dx.tolist()
        })

        if np.max(np.abs(nx - x)) <= eps:
            break
        x = nx.tolist()

    return Result(x, iterations, log)
```

2.2.2 Результат выполнения программы

```
Выберите что будете решать:
1 -> Нелинейное уравнение
2 -> Система нелинейных уравнений
1
Выберите уравнение:
1 -> x^3 - x + 4
2 -> x^3 - x^2 - 25x + 2
3 -> arctg(x)
1
Выберите метод:
1 -> Метод хорд
2 -> Метод секущих
3 -> Метод простых итераций
1
Выберите способ ввода границ интервала и точности:
1 -> Консоль
2 -> Файл
1
Введите нижнюю и верхнюю границу диапазона корней: -2 -1
Введите точность: 0.01
Выберите способ вывода ответа:
1 -> Консоль
2 -> Файл
1
Найденный корень: -1.7944736520357012
Потребовалось итераций: 3
Лог решения:
+---+-------+-------+-------+-------+------+------+-------+
|   |   a   |   b   |   x   | f(a)  | f(b) | f(x) | delta |
+---+-------+-------+-------+-------+------+------+-------+
| 0 | -2.00 | -1.67 | -1.67 | -2.00 | 1.04 | 1.04 | 0.11  |
| 1 | -2.00 | -1.78 | -1.78 | -2.00 | 0.14 | 0.14 | 0.01  |
| 2 | -2.00 | -1.79 | -1.79 | -2.00 | 0.02 | 0.02 | 0.00  |
+---+-------+-------+-------+-------+------+------+-------+
```

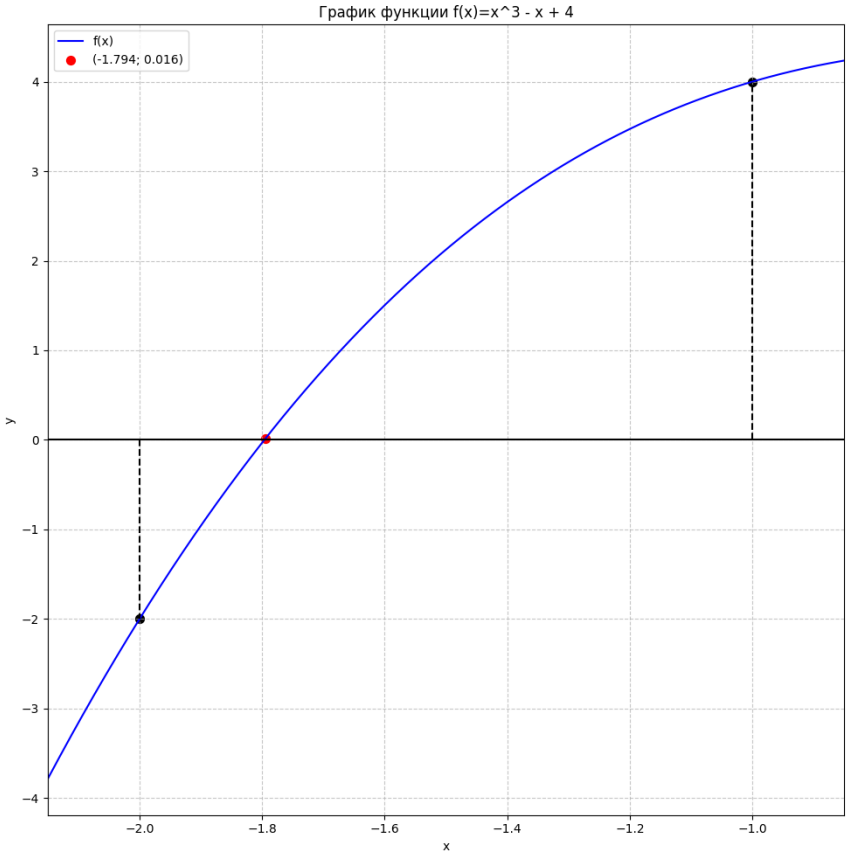
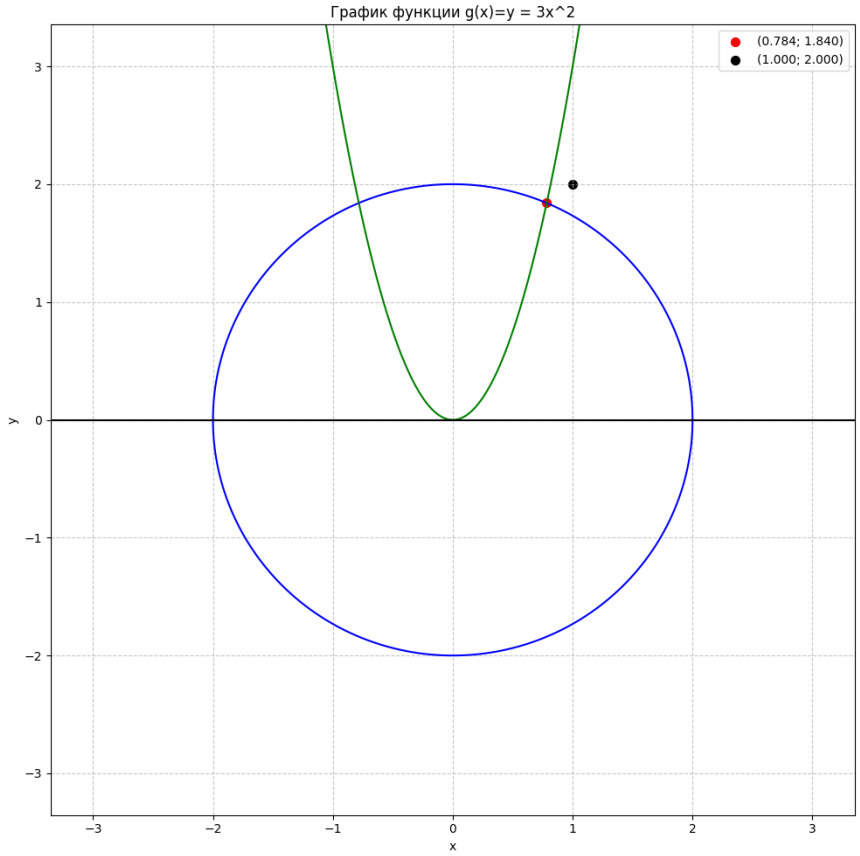


График функции f(x)=x^3 - x + 4
f(x)
(-1.794; 0.016)
4
3
2
1
0
−1
−2
−3
−4
y
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
x

```
Выберите что будете решать:
1 -> Нелинейное уравнение
2 -> Система нелинейных уравнений
2
Выберите систему:
1 -> {x^2 + y^2 = 4; y = 3x^2; }
2 -> {x^2 + y^2 = 4; y = sin(x); }
1
Выберите метод:
1 -> Метод Ньютона
1
Выберите способ ввода границ интервала и точности:
1 -> Консоль
2 -> Файл
1
Введите 2 координат начального приближения: 1 2
Введите точность: 0.01
Выберите способ вывода ответа:
1 -> Консоль
2 -> Файл
1
Найденный корень: [0.7835853106553743, 1.8402743753062225]
Потребовалось итераций: 3
Лог решения:
+---+------------------+------------------+--------------------+
|   |       x_i        |     x_{i+1}      |         dx         |
+---+------------------+------------------+--------------------+
| 0 | ['1.00', '2.00'] | ['0.81', '1.85'] | ['-0.19', '-0.15'] |
| 1 | ['0.81', '1.85'] | ['0.78', '1.84'] | ['-0.02', '-0.01'] |
| 2 | ['0.78', '1.84'] | ['0.78', '1.84'] | ['-0.00', '-0.00'] |
+---+------------------+------------------+--------------------+
```



3 Вывод

В результате выполнения данной лабораторной работой я познакомился с численными методами решения нелинейных уравнений и реализовал метод хорд, метод секущих и метод простой итерации на языке программирования Python. Также я познакомился с методом Ньютона для решения система нелинейных уравнений, также реализовав его на языке программирования Pyhton.

В ходе реализации вышеуказанных методов я на практике убедился в их преимуществах и недостатках.

Метод хорд является простым в реализации, но при этом обеспечивает линейную скорость сходимости. Также этот метод требует вычисления производных в случае применения фиксированной границы, однако я использовал другую реализацию, поэтому не столкнулся с данной проблемой.

Метод секущих требует меньше вычисления чем метод Ньютона, так как не нужно вычислять производные, однако порядок его сходимости ниже, и равен золотому сечению 1.618, что говорит о сверхлинейной сходимости, в то время как в методе Ньютона она квадратичная. Также минусом данного метода является необходимость вычисления второй производной для определения сходимости.

Метод простых итераций достаточно просто в реализации, однако он сходится в довольно малой окрестности и требует достаточно сложных вычислений для определения сходимости. Также нетривиальным может быть преобразование исходного уравнения к виду $x = \varphi(x)$.

Метод Ньютона для решения систем нелинейных уравнений обеспечивает хорошую сходимость при удачном выборе начального приближения, однако в противном случае сходимость может ухудшиться. Также достаточно большой проблемой является необходимость вычисления якобиана на каждой итерации, а также решения СЛАУ. Эти задачи сами по себе не являются тривиальными, а в случае умножения их сложности на число итераций могу привести к существенному снижению времени работы.