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

Факультет программной инженерии и компьютерной техники
Направление подготовки: 09.03.04 — Системное и прикладное программное обеспечение
Дисциплина «Вычислительная математика»

Лабораторная работа №2

Вариант 6

Выполнил:

Капарулин Тимофей Иванович

Преподаватель:

Машина Екатерина Алексеевна

г.Санкт-Петербург 2025 г.

Цель работы

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

Вычислительная реализация метода

$$f(x) = 2x^3 + 3.41x^2 - 23.74x + 2.95$$
$$f'(x) = 6x^2 + 6.82x - 23.74$$
$$f''(x) = 12x + 6.82$$

Часть 1. Решение нелинейного уравнения

1) Графическое отделение корней

2) Интервалы изоляции корней

левый корень: [-4.5, -4]
центральный корень: [0, 0.5]
правый корень: [2.5, 3]

3) Уточнение корней

3.1 крайний левый корень

Рассмотрим крайний левый корень, так как мы используем метод Ньютона, выберем начальное приближение:

$$f(-4.5) * f'(-4.5) > 0$$
$$f(-4) * f'(-4) < 0$$

Значит $x_0 = -4.5$.

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

Уточнение левого корня методом Ньютона

№	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_k - x_{k+1}\rvert$
0	-4.5	-3.418	67.07	-4.449	0.051
1	-4.449	-0.061	64.682	-4.448	0.001

Таким образом левый корень равен -4.449

3.2 центральный корень

Для вычисления центрального корня воспользуемся методом хорд. Проверим возможность зафиксировать точку:

$$f(0) * f'(0) > 0$$
$$f(0.5) * f'(0.5) < 0$$

Значит зафиксируем a и тогда $x_0 = 0.5$

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

Уточнение центрального корня методом хорд

№	a	b	x	f(a)	f(b)	f(x)	$\lvert x_k - x_{k+1}\rvert$
1	0	0.5	0.137	2.95	-7.817	-0.233	0.363
2	0	0.137	0.127	2.95	-0.233	-0.005	0.01

Таким образом центральный корень равен 0.127

3.3 крайний правый корень

Выразим x для метода простой итерации:

$$x = x + \lambda(2x^3 + 3.41x^2 - 23.74x + 2.95)$$

$$f'(2.5) = 30.81$$
$$f'(3) = 50.72$$

Тогда

$$\lambda = -0.02$$

$$x = x - 0.04x^3 - 0.068x^2 + 0.475x - 0.059$$
$$\phi(x) = x - 0.04x^3 - 0.068x^2 + 0.475x - 0.059$$
$$\phi'(x) = 1 - 0.12x^2 - 0.136x + 0.475$$

Проверим условие сходимости:

$$|\phi'(2.5)| = 0.385$$

$$|\phi'(3)| = 0.013$$

Тогда $x_0 = 3$ и алгоритм сходится.

Формула $x_{i+1} = \phi(x_i)$

Уточнение крайнего правого корня методом простых итераций

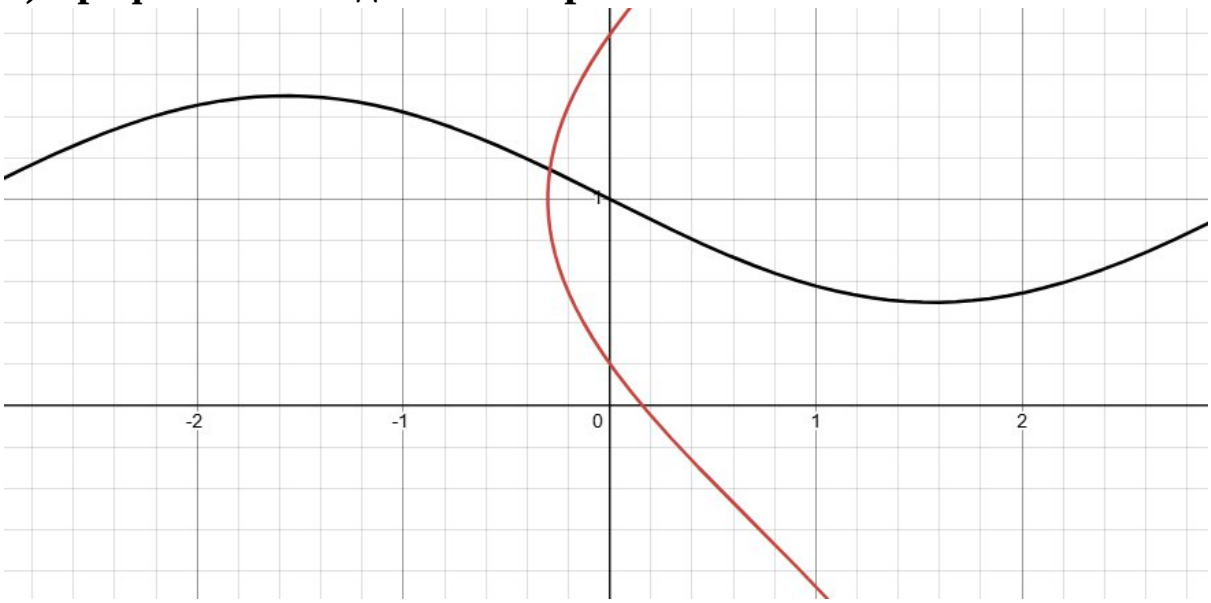
№	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_k - x_{k+1}\rvert$
1	3	2.674	2.091	0.326
2	2.674	2.634	0.631	0.04
3	2.634	2.623	0.249	0.01

Таким образом крайний правый корень равен 2.623

Часть 2. Решение системы нелинейных уравнений

$$\begin{cases} \sin(x) + 2y = 2 \\ x + \cos(y-1) = 0.7 \end{cases}$$

1) Графическое отделение корней



Интервал изоляции корня по x: [-0.4, -0.2], по y: [1, 1.2]

2) Решение методом простых итераций

Выразим x и y:

$$\begin{cases} x = 0.7 - \cos(y-1) \\ y = -0.5 * \sin(x) + 1 \end{cases}$$

$$\phi(X) = \begin{cases} 0.7 - \cos(y-1) \\ -0.5 * \sin(x) + 1 \end{cases}$$

$$\phi'(X) = \begin{matrix} 0 & sin(y-1) \\ -0.5 * cos(x) & 0 \end{matrix}$$

Для минимизации q возьмем $X_0 = (-0.4,\ 1)$.
Тогда $max|\phi'(X_0)| \leq 1$ и алгоритм сходится.

№	X_k	X_{k+1}	$\lvert X_k - X_{k+1}\rvert$
1	(-0.4, 1)	(-0.3, 1.195)	(0.1, 0.195)
2	(-0.3, 1.195)	(-0.281, 1.148)	(0.019, 0.047)
3	(-0.281, 1.148)	(-0.289, 1.139)	(0.008, 0.009)

Таким образом приближенным решением является X =(-0.289, 1.139)

Листинг программы

Половинное деление:

```
    def __solution(self, f):
        interval = self.interval.copy()
        diff = float('inf')
        self.cnt = 0

        while(diff > self.e):
            interval = self.__iter(f, interval)
            diff = interval[1] - interval[0]
            self.cnt+=1

        self.solution = (interval[1] + interval[0])/2
        self.f_x = f(self.solution)


    def __iter(self, f, interval):
        x = (interval[1] + interval[0])/2

        if(f(x)*f(interval[1])<0):
            interval[0] = x
        elif(f(x)*f(interval[1])>0):
            interval[1] = x
        else:
            interval[0] = x
            interval[1] = x

        return interval
```

Метод секущих:

```
def __solution(self, f):
        diff = float('inf')
        x_p = self.x0
        x = x_p + self.e
        self.cnt = 0

        while(diff > self.e):
            x_n = self.__iter(f, x_p, x)
            self.cnt+=1
            x_p = x
            x = x_n
            diff = abs(x - x_p)

        self.solution = x
```

```
        self.f_x = f(self.solution)

    def __iter(self, f, x_p, x):
        return x - (x - x_p)/(f(x) - f(x_p))*f(x)
```

Метод простых итераций:

```
    def __solution(self, f):
        diff = float('inf')
        x = self.x0
        self.cnt = 0

        while(diff > self.e):
            x_n = self.__phi(f, x)
            self.cnt+=1
            diff = abs(x - x_n)
            x = x_n

        self.solution = x
        self.f_x = f(self.solution)

    def __phi(self, f, x):
        return x + self.Lambda * f(x)
```

Пример работы программы

- Пример 1

$$f(x) = sin(x) + cos(x)$$

Начально придлижение: [-1, 0]
E: 0.01
Метод половинного деления: x = -0.7852
Метод секущих: x = -0.7854
Метод простых итерации: x = -0.7855
Истинное решение: x = -0,7854

- Пример 2

$$\begin{cases} x^2 + y^2 - 4 = 0 \\ -3x^2 + y = 0 \end{cases}$$

Начально придлижение: [-1, 2]
E: 0.01
Метод Ньютона: X = (-0.7974, 1.8066)
Истинное решение:X = (-0.7832, 1.8403)

Выводы

В данной работе были реализованы методы половинного деления, секущих и простых итераций для решения нелинейных уравнений, метод Ньютона – для систем нелинейных уравнений. Методы были протестированы на различных примерах с различной степенью начального приближения. Результаты показали, что реализованные алгоритмы успешно справляется с поставленной задачей и находят решения в пределах допустимых погрешностей.