Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

По вычислительной математике

Вариант 3

Выполнил:
Студент группы P3216
Векшин Арсений Иванович
Преподаватель:
Малышева Татьяна Алексеевна

г. Санкт-Петербург

2024 г.

Оглавление

Цель работы

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

Порядок выполнения работы

1. **Вычислительная реализация задачи**
   - 1.1. **Решение нелинейного уравнения**
     - 1.1.1. Отделить корни заданного нелинейного уравнения графически (вид уравнения представлен в табл. 6)
     - 1.1.2. Определить интервалы изоляции корней.
     - 1.1.3. Уточнить корни нелинейного уравнения (см. табл. 6) с точностью $\varepsilon=10^{-2}$.
     - 1.1.4. Используемые методы для уточнения каждого из 3-х корней многочлена представлены в таблице 7.
     - 1.1.5. Вычисления оформить в виде таблиц (1-5), в зависимости от заданного метода. Для всех значений в таблице удержать 3 знака после запятой.
       - 1.1.5.1. Для метода половинного деления заполнить таблицу 1.
       - 1.1.5.2. Для метода хорд заполнить таблицу 2.

1.1.5.3. Для метода Ньютона заполнить таблицу 3.
1.1.5.4. Для метода секущих заполнить таблицу 4.
1.1.5.5. Для метода простой итерации заполнить таблицу 5. Проверить условие сходимости метода на выбранном интервале.

1.2. **Решение системы нелинейных уравнений**

1.2.1. Отделить корни заданной системы нелинейных уравнений графически (вид системы представлен в табл. 8). 2.
1.2.2. Используя указанный метод, решить систему нелинейных уравнений с точностью до 0,01.
1.2.3. Для метода простой итерации проверить условие сходимости метода.
1.2.4. Подробные вычисления привести в отчете.

2. **Программная реализация задачи**

2.1. **Для нелинейных уравнений**

2.1.1. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.
2.1.2. 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.
2.1.3. Для методов, требующих начальное приближение к корню (методы Ньютона, секущих, хорд с фиксированным концом, простой итерации), выбор начального приближения $x0$(a или b) вычислять в программе.
2.1.4. Для метода простой итерации проверять достаточное условие сходимости метода на введенном интервале.
2.1.5. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.
2.1.6. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом). Пользователь должен видеть интервалы изоляции корней.

2.2. **Для систем нелинейных уравнений**

2.2.1. Организовать вывод графика функций.
2.2.2. Начальные приближения ввести с клавиатуры.
2.2.3. Для метода простой итерации проверить достаточное условие сходимости.
2.2.4. Организовать вывод вектора неизвестных: x_1, x_2.
2.2.5. Организовать вывод количества итераций, за которое было найдено решение.
2.2.6. Организовать вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$
2.2.7. Проверить правильность решения системы нелинейных уравнений.

Рабочие формулы используемых методов

Метод половинного деления (Решение НУ)

Идея метода: начальный интервал изоляции корня делим пополам, получаем начальное приближение к корню:

$$\boldsymbol{x_0 = \frac{a_0 + b_0}{2}}$$

Вычисляем 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]. Другую половину отрезка [a_0 , b_0], на которой функция f(x) знак не меняет, отбрасываем. Новый интервал вновь делим пополам, получаем очередное приближение к корню: $x_1=(a_1+b_1)/2$. и т.д.

Рабочая формула метода: $\boldsymbol{x_0 = \frac{a_i + b_i}{2}}$

Приближенное значение корня: $\boldsymbol{x^* = \frac{a_i + b_i}{2}}$ или $x^* = a_n$ или $x^* = b_n$

Метод простой итерации (Решение НУ)

Уравнение $f\ x = 0$ приведем к эквивалентному виду: $x = \varphi(x)$, выразив x из исходного уравнения. Зная начальное приближение: $x0 \in a, b$, найдем очередные приближения: $x1 = \varphi(x0) \rightarrow x2 = \varphi\ x1$ ...
Рабочая формула метода: $\boldsymbol{x_{i+1} = \varphi(xi)}$
Условия сходимости метода простой итерации определяются следующей теоремой.
Теорема. Если на отрезке локализации a, b функция $\varphi(x)$ определена, непрерывна и дифференцируема и удовлетворяет неравенству:
$|\varphi'(x)| < q$, где $0 \le q < 1$, то независимо от выбора начального приближения $x_0 \in a, b$ итерационная последовательность $\{x_n\}$ метода будет сходиться к корню уравнения.
Достаточное условие сходимости метода:
$|\varphi'(x)| \le q < 1$, где q – некоторая константа (коэффициент Липшица или коэффициент сжатия)
$q = \max[a,b]\ |\varphi'\ x|$
При $q \approx 0$ - скорость сходимости высокая,
При $q \approx 1$ - скорость сходимости низкая,
При $q > 1$ - нет сходимости.
Чем меньше q, тем выше скорость сходимости.

Метод Ньютона (Решения НУ)

Функция $y = f(x)$ на отрезке [a, b] заменяется касательной и в качестве приближенного значения корня принимается точка пересечения касательной с осью абсцисс.
Пусть $x_0 \in [a, b]$ - начальное приближение. Запишем уравнение касательной к графику функции $y = f(x)$в этой точке: $y=f(x_0) + f'(x_0)(x-x_0)$
Найдем пересечение касательной с осью x: $x_1 = x_0 - f(x_0)/f'(x_0)$
Рабочая формула метода: $\boldsymbol{x_i = x_{i-1} - f(x_{i-1})/f'(x_{i-1})}$

Метод простой итерации (Решение СНУ)

Приведем систему уравнений к эквивалентному виду:

$$\begin{cases} F_1(x_1, x_2, \dots, x_n) = 0 \\ F_2(x_1, x_2, \dots, x_n) = 0 \\ \dots\dots\dots\dots\dots\dots \\ F_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \qquad \begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n) \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n) \\ \dots\dots\dots\dots\dots\dots \\ x_n = \varphi_n(x_1, x_2, \dots, x_n) \end{cases}$$

Если выбрано начальное приближение:

$^{(0)} = x_1^{(0)}, x_2^{(0)}, \dots, x_n{}^{(0)}$, получим первые приближения к корням:

$$\begin{cases} x_1^{(1)} = \varphi_1(x_1^0, x_2^0, \dots, x_n^0) \\ x_2^{(1)} = \varphi_2(x_1^0, x_2^0, \dots, x_n^0) \\ \dots\dots\dots\dots\dots\dots \\ x_n^{(1)} = \varphi_n(x_1^0, x_2^0, \dots, x_n^0) \end{cases}$$

Последующие приближения находятся по формулам:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^k, x_2^k, \dots, x_n^k) \\ x_2^{(k+1)} = \varphi_2(x_1^k, x_2^k, \dots, x_n^k) \\ \dots\dots\dots\dots\dots\dots \\ x_n^{(k+1)} = \varphi_n(x_1^k, x_2^k, \dots, x_n^k) \end{cases} \qquad k = 0, 1, 2, \dots$$

Вычислительная реализация

Уравнение: $x^3+2{,}84x^2-5{,}606x-14{,}766$

Интервалы изоляции корней:

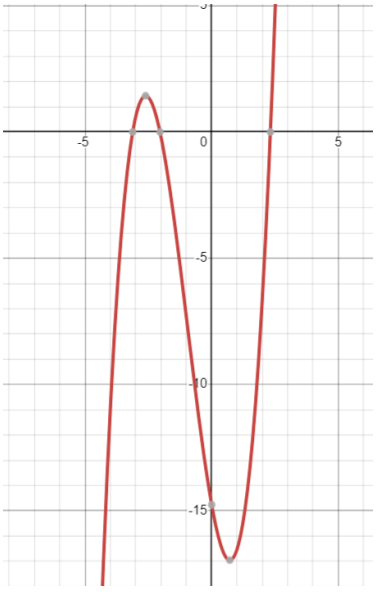

Рисунок 1 - Графический способ определения интервалов изоляции корней НУ

Крайний левый корень: [-3.7; -2.7]
Крайний правый корень: [2; 4]
Центральный корень: [-3; -1]

Проверка сходимости метода простой итерации

$f(x) = x^3 + 2{,}84x^2 - 5{,}606x - 14{,}766$
$f'(x) = 3x^2 + 5.68x - 5.606$
$\varphi(x) = x + \lambda(x^3 + 2.84x^2 - 5.606x - 14.766)$

$$f'(-3.7) = 14.488 \quad f'(x) = 0.928 \quad \lambda = -\frac{1}{14.448} \approx -0.069$$

a = -3.7 b = -2.7
$|\varphi'(-3.7)| \approx 0.069 < 1$ и $|\varphi'(-2.7)| \approx 0.936 < 1$

Условие сходимости выполняется

Таблицы приближения корней:

Таблица 1. Уточнение корня уравнения методом простой итерации

№ итерации	x_k	x_{k+1}	$f(x_k)$	$\lvert x_{k+1}-x_k\rvert$
0	-3.1645	-3.2	-0.2752	0.0355
1	-3.1454	-3.1645	-0.1546	0.0191
2	-3.1347	-3.1454	-0.0889	0.0107
3	-3.1286	-3.1347	-0.0518	0.0062
4	-3.125	-3.1286	-0.0304	0.0036

Таблица 2. Уточнение корня уравнения методом половинного деления

№ шага	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	2.0	3.0	3.0	-6.618	20.976	20.976	1.0
2	2.0	2.5	2.5	-6.618	4.594	4.594	0.5
3	2.25	2.5	2.25	-1.6114	4.594	-1.6114	0.25
4	2.25	2.375	2.375	-1.6114	1.3356	1.3356	0.125
5	2.3125	2.375	2.3125	-0.1761	1.3356	-0.1761	0.0625
6	2.3125	2.3438	2.3438	-0.1761	0.5701	0.5701	0.0312
7	2.3125	2.3281	2.3281	-0.1761	0.1946	0.1946	0.0156
8	2.3125	2.3203	2.3203	-0.1761	0.0087	0.0087	0.0078
9	2.3164	2.3203	2.3164	-0.0838	0.0087	-0.0838	0.0039
10	2.3184	2.3203	2.3184	-0.0376	0.0087	-0.0376	0.002
11	2.3193	2.3203	2.3193	-0.0145	0.0087	-0.0145	0.001

Таблица 3. Уточнение корня уравнения методом Ньютона

№ итерации	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_{k+1}-x_k\rvert$
1	-2.0	-0.0049	-4.7145	-2.0391	0.0391
2	-2.0391	-0.0	-4.7077	-2.0401	0.001

Система НУ: $\begin{cases} cos(x - 1) + y = 0{,}5 \\ x - \cos(y) = 3 \end{cases}$
Область корня: 3 < x < 4; 1 < y < 2
Эквивалентная система: $\begin{cases} y = 0{,}5 - cos(x - 1) & \varphi 1 \\ x = 3 + cos(y) & \varphi 2 \end{cases}$
Частные производные:
$\partial\varphi_1/\partial x = \sin(1 - x) < 1$ на промежутке 3 < x < 4
$\partial\varphi_1/\partial y = 0$
$\partial\varphi_2/\partial x = 0$
$\partial\varphi_2/\partial y = \sin(y) \le 1$ на промежутке ; 1 < y < 2

Очевидно, что процесс сходящийся, так как синус и косинус не дают значений выше 1 и суммируются с нулем. Сходимость близка к 1 -> может быть очень медленной

Начальное приближение: x = 3; y = 1

Таблица 4. Уточнение корней системы методом простой итерации

№ итерации	x	y	x_{i+1}	y_{i+1}	$F_1(x_{i+1}; y_{i+1})$	$F_2(x_{i+1}; y_{i+1})$	$\lvert x_{i+1}-x_i \rvert$	$\lvert y_{i+1}-y_i \rvert$
1	3	1	3.54	0.916	-0.408	-0.069	0.54	0.083
2	3.54	0.916	3.609	1.324	-0.037	0.365	0.068	0.408
3	3.609	1.324	3.244	1.361	0.238	0.036	0.364	0.037
4	3.244	1.361	3.208	1.123	0.028	-0.225	0.036	0.238
5	3.208	1.123	3.433	1.095	-0.164	-0.025	0.224	0.028
6	3.433	1.095	3.455	1.259	-0.016	0.151	0.025	0.164
7	3.455	1.259	3.307	1.275	0.104	0.015	0.151	0.016
8	3.307	1.275	3.292	1.171	0.011	-0.097	0.015	0.103
9	3.292	1.171	3.389	1.16	-0.07	-0.01	0.97	0.011
10	3.389	1.16	3.399	1.23	-0.007	0.065	0.01	0.07
11	3.399	1.23	3.334	1.237	0.046	0.006	0.064	0.007
12	3.334	1.237	3.328	1.191	0.004	-0.043	0.006	0.046
13	3.328	1.191	3.371	1.187	-0.03	-0.003	0.043	0.004
14	3.371	1.187	3.374	1.217	-0.003	0.028	0.003	0.03
15	3.374	1.217	3.346	1.22	0.02	0.002	0.028	0.003
16	3.346	1.22	3.344	1.2	0.002	-0.018	0.002	0.02
17	3.344	1.2	3.362	1.198	-0.013	-0.002	0.018	0.003
18	3.362	1.198	3.364	1.211	-0.002	0.012	0.002	0.02
19	3.364	1.211	3.352	1.213	0.009	0.002	0.012	0.002
20	3.352	1.213	3,35	1.204	0.001	-0.009	0.002	0.009

Исходный код программы

Метод половинного деления

```
def solve(f, a, b, accuracy):
    iter = 0
    x = 0.0
    print_table_header(["#", "a", "b", "x_i", "f(a)", "f(b)", "f(x_i)", "|a-b|"])
    while(abs(a-b) > accuracy and abs(f(x)) >= accuracy):
        iter += 1
        x = mid(a, b)
        if f(a)*f(x) > 0: a = x
        else: b = x
        print_table_row([iter, a, b, x, f(a), f(b), f(x), abs(a-b)])
    x = mid(a, b)
    return x
```

Метод секущих

```
def solve(f, start, base_move, accuracy):
    iter = 0
    x, prev_x = base_move, start

    def find_x():
        return x - (x-prev_x) / (f(x)-f(prev_x)) * f(x)

    print_table_header(["#", "x_{i-1}", "x_i", "x_{i+1}", "f(x_i+1)" ,"delta_x"])
    while (abs(x - prev_x) > accuracy and abs(f(x)) >= accuracy):
        iter += 1
        _out = [iter, prev_x, x]
        x = find_x()
        prev_x = _out[-1]
        print_table_row(_out + [x, f(x), abs(x - prev_x)])

    return x
```

Метод простой итерации

```python
def get_lambda(f, a, b, accuracy):
    x = a
    x_max = x
    l = 0
    sign = 1
    while (x <= b):
        if l < abs(f(x)):
            l = max(l, abs(f(x)))
            sign = abs(f(x)) / f(x)
            x_max = x
        x += accuracy
    return 1 / l * sign, x_max


def solve(f, deriv, a, b, accuracy):
    x = a
    prev_x = x - 2 * accuracy
    iter = 1
    lambd, x0 = get_lambda(deriv, a, b, accuracy)
    q = abs(1 + lambd * deriv(x0))
    print(1 + lambd * deriv(a), 1 + lambd * deriv(b), lambd)
    if q > 1:
        print('Достаточное условие сходимости не выполняется!')
    elif q > 0.5:
        accuracy = (1 - q) / q * accuracy

    print_table_header(["#", "x_i", "x_{i+1}", "f(x_i)", "delta_x"])
    while abs(f(x)) > accuracy and iter < 10000:
        # if (q >= 1 and iter > 3):
        #     break
        prev_x = x
        x = x + f(x) * lambd
        print_table_row([iter, x, prev_x, f(x), abs(x - prev_x)])
        iter += 1
    return x
```

Метод Ньютона

```
def solve(f, deriv, a, b, start, accuracy):
    def find_x():
        return x - f(x) / deriv(x)

    iter = 0
    x, prev_x = start, 0
    print_table_header(["#", "x_i", "f(x)", "f'(x)", "x_{i+1}", "delta_x"])
    while (abs(f(x)/deriv(f(x))) > accuracy
           and abs(x - prev_x) > accuracy and abs(f(x)) >= accuracy):
        iter += 1
        prev_x = x
        x = find_x()
        print_table_row([iter, prev_x, f(x), deriv(x), x, abs(x - prev_x)])

    return x
```

Метод Хорд

```
def solve(f, a, b, accuracy):
    def find_x():
        return a - (b-a)*f(a)/(f(b)-f(a))

    iter = 0
    x = find_x()
    prev_x = 0.0
    print_table_header(["#", "a", "b", "x", "f(a)", "f(b)", "f(x)", "delta_x"])
    while(abs(a-b) > accuracy and abs(x-prev_x) > accuracy and abs(f(x)) >= accuracy):
        iter += 1
        if iter > 1: prev_x = x
        x = find_x()
        if is_dif_sign(f(a), f(x)):
            b = x
        elif is_dif_sign(f(b), f(x)):
            a = x
        else:
            print("ERROR: функция не пересекает 0 на заданном промежутке", a, b)
            return -1
        print_table_row([iter, a, b, x, f(a), f(b), f(x), abs(x - prev_x)])

    return x
```

Метод итераций для системы

```
def solve(functions, x0, y0, accuracy):
    x, y = x0, y0
    prev_x, prev_y = 0, 0

    iter = 0
    print_table_header(["#", "x", "y", "delta_x", "delta_y"])
    while max(abs(x-prev_x), abs(y-prev_y)) > accuracy:
        iter += 1

        prev_x, prev_y = x, y
        x = functions[0](x,y)
        y = functions[1](x,y)
        print_table_row([iter, x, y, abs(x-prev_x), abs(y-prev_y)])
    return x, y

def is_possible(deriv_functions, x0, y0):
    for df in deriv_functions:
        print(df(x0, y0))
        if (df(x0, y0)>1):
            print("ERROR: условие сходимости не выполняется в окресности заданной точки")
            sys.exit(1)
    print("Условие сходимости выполняется")
```

Полный код программы:
https://github.com/ArsenyVekshin/ITMO/tree/master/CompMath/lab2

Вывод

Таким образом, в результате выполнения лабораторной работы я изучил принцип работы метода Гаусса на практике.