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

Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

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

Вариант №4

Группа: P3218

Студент: Горло Евгений Николаевич

Преподаватель: Бострикова Дарья Константиновна

Санкт-Петербург
2024

Содержание

1 Цель лабораторной 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.
- Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.
- Предусмотреть 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 Рабочие формулы использованных методов

3.1 Метод Ньютона (касательных)

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

3.2 Метод половинного деления

Метод состоит в том, что начальный интервал изоляции корня делится пополам. Вычисляем значение $f(x_i)$ и в качестве интервала выбираем ту половину отрезка, на концах которого функция имеет разные знаки: $[a_i, x_i]$ либо $[b_i, x_i]$
Рабочая формула метода: $x_i = \frac{a_i + b_i}{2}$
Условие прекращения итерационного процесса:
$|x_n - x_{\text{n-1}}| <= \epsilon$, где ϵ - заданная точность.
Приближенное значение корня: $x* = \frac{a_n + b_n}{2}$, или $x* = a_n$, или $x* = b_n$

3.3 Метод простой итерации

Приводим уравнение $f(x) = 0$ к эквивалентному виду $x = \phi(x)$, выразив x из исходного уравнения.
Зная начальное приближение на отрезке $[a, b]$, находим очередные приближения $x1 = \phi(x_0) -> x_2 = \phi(x_1)$
Рабочая формула метода: $x_{\text{i+1}} = \phi(x_i)$
Достаточное условие сходимости метода: $|\phi'(x) <= q < 1|$, где q - коэффициент Липшица.
$q = max|\phi'(x)|$
Критерий окончания итерационного процесса: $|x_n - x_(n - 1)| <= \epsilon$

4 Графики функций

4.1 Задание 1

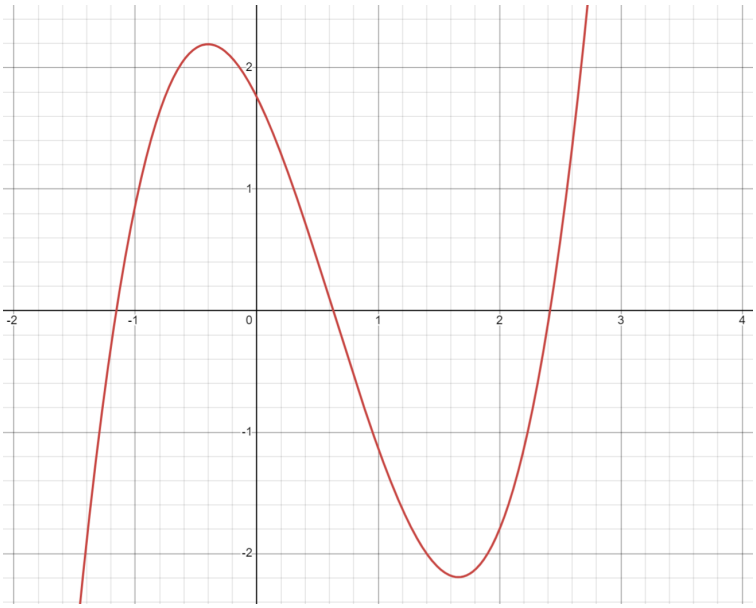

Рис. 1: График функции

$$x^3 - 1.89x^2 - 2x + 1.76$$

4.2 Задание 2

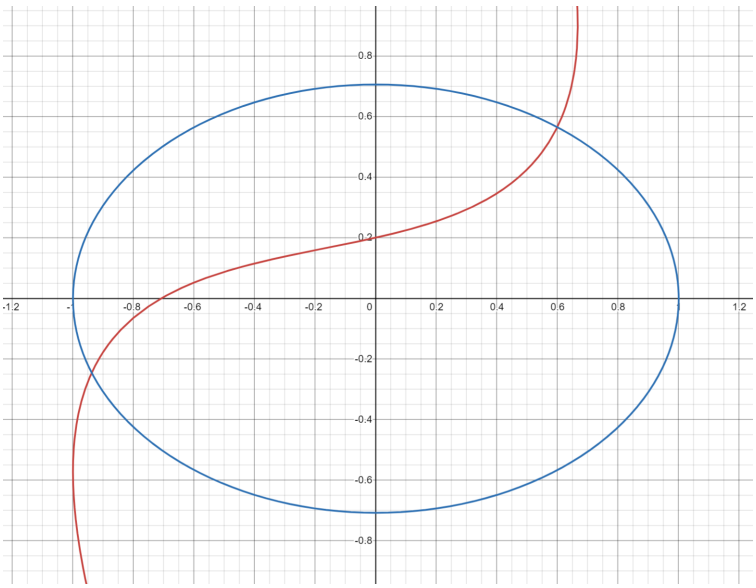

Рис. 2: График системы уравнений: $\sin(x+y) - 1.2x = 0.2$ и $x^2 + 2y^2 = 1$

5 Заполненные таблицы вычислительной части №1

$x^3 - 1.89x^2 - 2x + 1.76$
1) Отделим корни графически:
$x1 = -1,156$
$x2 = 0,63$
$x3 = 2,416$
2) Определим интервалы изоляции:
$[-2;0]$ для $x1$
$[0;2]$ для $x2$
$[2;4]$ для $x3$
3) Уточним корни с точностью $\epsilon = 0,01$

5.1 Метод половинного деления для уточнения корня x1

№	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	-2	0	-1	-9,800	1,760	0,870	2
2	-2	-1	-1,5	-9,800	0,870	-2,867	1
3	-1,5	-1	-1,25	-2,867	0,870	-0,646	0,5
4	-1,25	-1	-1,125	-0,646	0,870	0,194	0,25
5	-1,25	-1,125	-1,188	-0,646	0,194	-0,208	0,125
6	-1,188	-1,125	-1,157	-0,208	0,194	-0,005	0,063
7	-1,157	-1,125	-1,141	-0,005	0,194	0,096	0,032
8	-1.157	-1,141	-1,149	-0,005	0,096	0,046	0,016
9	-1.157	-1,149	**1,153**	-0,005	0,046	0,021	$0,008 < \epsilon$

5.2 Метод секущих для уточнения корня x2

№	$x_{\text{k-1}}$	x_k	$x_{\text{k+1}}$	$f(x_{\text{k+1}})$	$\lvert x_{\text{k+1}} - x_k \rvert$
1	0	1	0,609	0,06691	0,391
2	1	0,609	0,63086	-0,00282	0,02186
3	0,609	0,63086	**0,62997**	0	$0{,}00088 < \epsilon$

5.3 Метод простой итерации для уточнения x3

№	x_k	$x_{\text{k+1}}$	$f(x_{\text{k+1}})$	$\lvert x_{\text{k+1}} - x_k \rvert$
1	2,000	2,140	-1,375	0,140
2	2,140	2,236	-0,983	0,096
3	2,236	2,299	-0,674	0,064
4	2,299	2,341	-0,449	0,042
5	2,341	2,368	-0,294	0,027
6	2,368	2,386	-0,191	0,017
7	2,386	2,397	-0,123	0,011
8	2,397	**2,404**	-0,079	$0{,}007 < \epsilon$

6 Решение системы нелинейных уравнений вычислительной части №2

Левый корень:
Начальное приближение: $x_0 = -0,94, y_0 = -0,25$
Шаг 1.

$$\begin{cases} \left(\cos(x+y) + \dfrac{-6}{5}\right)\Delta x + (\cos(x+y))\,\Delta y = \dfrac{x \cdot 6}{5} + \dfrac{1}{5} - \sin(x+y) \\ (2 \cdot x)\Delta x + (4 \cdot y)\Delta y = -(2 \cdot y^2) + 1 - x^2 \end{cases}$$

На первой итерации система будет иметь вид:

$$\begin{cases} -0.82834\Delta x + 0.37166\Delta y = 0.00037 \\ -1.88\Delta x - 1\Delta y = -0.0086 \end{cases}$$

Шаг 2. Решаем полученную систему. Получаем $\Delta x = -0.06191$ и $\Delta y = -0.26827$.
Шаг 3. Вычисляем очередные приближения:

$$x_1 = x_0 + \Delta x = -0.94 - 0.06191 = -1.00191,$$
$$y_1 = y_0 + \Delta y = 0 - 0.26827 = -0.26827.$$

Шаг 4. Проверяем критерии окончания итерационного процесса при $\epsilon = 0.01$:

$$|x_1 - x_0| \le \epsilon \quad \text{и} \quad |y_1 - y_0| \le \epsilon,$$
$$|-1.00191 - (-0.94)| \le \epsilon \quad \text{и} \quad |-0.26827 - 0| > \epsilon.$$

Шаг 5. Подставляем очередные приближения в систему:

$$\begin{cases} -0.9039\Delta x + 0.2961\Delta y = -0.04714, \\ -2.00383\Delta x - 1.07308\Delta y = -0.14777. \end{cases}$$

Шаг 6. Решаем полученную систему. Получаем $\Delta x = 0.06035$ и $\Delta y = 0.02502$.
Шаг 7. Вычисляем очередные приближения:

$$x_2 = x_1 + \Delta x = -1.00191 + 0.06035 = -0.94157,$$
$$y_2 = y_1 + \Delta y = -0.26827 + 0.02502 = -0.24326.$$

Шаг 8. Проверяем критерии окончания итерационного процесса при $\epsilon = 0.01$:

$$|x_2 - x_1| \le \epsilon \quad \text{и} \quad |y_2 - y_1| \le \epsilon,$$
$$|-0.94157 - (-1.00191)| \le \epsilon \quad \text{и} \quad |-0.24326 - (-0.26827)| > \epsilon.$$

Шаг 9. Подставляем очередные приближения в систему:

$$\begin{cases} -0.82354\Delta x + 0.37646\Delta y = -0.00345, \\ -1.88313\Delta x - 0.97302\Delta y = -0.00489. \end{cases}$$

Шаг 10. Решаем полученную систему. Получаем $\Delta x = 0.00344$ и $\Delta y = -0.00163$.

Шаг 11. Вычисляем очередные приближения:

$$x_3 = x_2 + \Delta x = -0.94157 + 0.00344 = -0.93813,$$
$$y_3 = y_2 + \Delta y = -0.24326 - 0.00163 = -0.24489.$$

Шаг 12. Проверяем критерии окончания итерационного процесса при $\epsilon = 0.01$:

$$|x_3 - x_2| \le \epsilon \quad \text{и} \quad |y_3 - y_2| \le \epsilon,$$
$$|-0.93813 - (-0.94157)| \le \epsilon \quad \text{и} \quad |-0.24489 - (-0.24326)| \le \epsilon.$$

Правый корень:

Шаг 1. Выбираем $x_0 = 0.6$, $y_0 = 0.5$. Исходная система:

$$\begin{cases} \left(\cos(x+y) + \dfrac{-6}{5}\right)\Delta x + (\cos(x+y))\,\Delta y = \dfrac{x \cdot 6}{5} + \dfrac{1}{5} - \sin(x+y), \\ (2 \cdot x)\Delta x + (4 \cdot y)\Delta y = -(2 \cdot y^2) + 1 - x^2. \end{cases}$$

На первой итерации система будет иметь вид:

$$\begin{cases} -0.7464\Delta x + 0.4536\Delta y = 0.02879, \\ 1.2\Delta x + 2\Delta y = 0.14. \end{cases}$$

Шаг 2. Решаем полученную систему. Получаем $\Delta x = 0.00291$ и $\Delta y = 0.06826$.

Шаг 3. Вычисляем очередные приближения:

$$x_1 = x_0 + \Delta x = 0.6 + 0.00291 = 0.60291,$$
$$y_1 = y_0 + \Delta y = 0.5 + 0.06826 = 0.56826.$$

Шаг 4. Проверяем критерии окончания итерационного процесса при $\epsilon = 0.01$:

$$|x_1 - x_0| \le \epsilon \quad \text{и} \quad |y_1 - y_0| \le \epsilon,$$
$$|0.60291 - 0.6| \le \epsilon \quad \text{и} \quad |0.56826 - 0.5| > \epsilon.$$

Шаг 5. Подставляем очередные приближения в систему:

$$\begin{cases} -0.81092\Delta x + 0.38908\Delta y = 0.00228, \\ 1.20581\Delta x + 2.27303\Delta y = -0.00933. \end{cases}$$

Шаг 6. Решаем полученную систему. Получаем $\Delta x = -0.00381$ и $\Delta y = -0.00208$.

Шаг 7. Вычисляем очередные приближения:

$$x_2 = x_1 + \Delta x = 0.60291 - 0.00381 = 0.59909,$$
$$y_2 = y_1 + \Delta y = 0.56826 - 0.00208 = 0.56618.$$

Шаг 8. Проверяем критерии окончания итерационного процесса при $\epsilon = 0.01$:

$$|x_2 - x_1| \le \epsilon \quad \text{и} \quad |y_2 - y_1| \le \epsilon,$$
$$|0.59909 - 0.60291| \le \epsilon \quad \text{и} \quad |0.56618 - 0.56826| \le \epsilon.$$

7 Листинг программы

```
def half_method(quation, method):
    a, b, inaccuracy = input_selection(quation,method)

    iterations = 0
    max_iter = 10000

    if(float(b) - float(a) - 0.00001 < 0):
        print("The right boundary must be greater\nthan the left boundary
        exit()

    count_roots = count_roots_on_interval(quation, a, b, 0.001)
    if not (validate_roots(count_roots)):
        exit()


    # a = int(sys.argv[4])
    # b = int(sys.argv[5])
    # inaccuracy = int(sys.argv[6])
    a1 =a
    b1 = b

    iterations = 0
    max_iter = 1000
    while (b - a) / 2 > inaccuracy and iterations < max_iter:
        midpoint = (a + b) / 2
        if quation_solution(quation, midpoint) == 0:
            return midpoint, iterations
        elif quation_solution(quation, a) * quation_solution(quation, mid
            b = midpoint
        else:
            a = midpoint
        iterations += 1
    if iterations >= max_iter-1:
        print("Solution not found")
        exit()
```

Листинг 1: Метод половинного деления

```
def Newton_method(quation, method):
    a, b, inaccuracy = input_selection(quation,method)

    approximation = validate_initial_approximation(quation, a, b)

    iterations = 0
    max_iter = 1000
    x = approximation
    while abs(quation_solution(quation,x)) >
    inaccuracy and iterations < max_iter:
        x = x - quation_solution(quation,x) /
        quation_df_solution(quation,x)
        iterations += 1

    solution = try_to_convert_to_int(x)
    output_data(solution,
    quation_solution(quation, solution),
    iterations, str_quation(quation))
    draw_grapth(quation, str_quation(quation), a, b)
```

Листинг 2: Метод Ньютона для нелинейных уравнений

```
def Simple_iteration_method(quation, method):
    a, b, inaccuary = input_selection(quation,method)
    # lambda_L = -1/(max(abs(quation_df_solution(quation,a)),
    abs(quation_df_solution(quation,b))))
    # print("L = ",lambda_L)
    q = check_convergencecondition(quation, a, b)
    approximation = validate_initial_approximation(quation, a, b)
    x = approximation
    iterations = 0
    max_iter = 1000
    if q>1:
        print("The convergence condition is NOT met")
        exit()
    elif( 0< q <= 0.5):
        while abs(converted_quation(quation,x)-x) >
        inaccuary and iterations < max_iter:
            x = converted_quation(quation,x)
            print(x)
            iterations += 1
    elif(0.5 <q <1):
        while abs(converted_quation(quation,x)-x) >
        ((1-q)/q)*inaccuary and iterations < max_iter:
            x = converted_quation(quation,x)
            iterations += 1

    solution = try_to_convert_to_int(x)
    output_data(solution,
    quation_solution(quation, solution),
    iterations, str_quation(quation))
    draw_grapth(quation, str_quation(quation), a, b)
```

Листинг 3: Метод простой итерации для нелинейных уравнений

```
def simple_iteration_method(system,method) :
    x0 = choose_x()
    y0 = choose_y()
    a = x0
    b = y0
    inaccuracy = choose_inaccuracy()
    max_iter = 1000
    iteration =0

    if not check_convergence(system, method, x0, y0):
        print("
        The iteration matrix does not\nsatisfy the convergence
        condition.")
        exit()

    for i in range(max_iter):
        x = f1(x0, y0,system)
        y = f2(x0, y0,system)
        if (abs(x-x0)<
        inaccuracy)and(abs(y-y0)< inaccuracy):
            print(f"\n\nx = {x}\ny = {y}")
            print(f"Iterations = {iteration}")
            print(f"inncaury x = {x-f1(x,y,system)}\
            ninncaury y = {y-f2(x,y,system)}")
            break
        x0, y0 = x, y
        iteration += 1
    output_data(x0, y0,
    quation_solution(system, method), iteration)
```

Листинг 4: Метод простой итерации для систем

Ссылка на проект: https://github.com/Djerden/computation-math/tree/main/lab2

8 Результаты выполнения программы

```
Choose the variant:
1) Non-linear equation
2) System of non-linear equations
1
Choose the quation:
1) x^2 - 3x + 2
2) x^3 + 2x^2 - 5
3) sin(x) - cos(x)
1
Choose the method:
1) Half-division method
2) Newton's method
3) Simple iteration method
1
Choose the quation:
1) Hand input
2) File input
1
Enter the left boundary of the interval: 1
Enter the right boundary of the interval: 5
Enter the inaccuracy: 0.01
Answer = 4.9921875
f(answer) = 11.94537353515625
Iterations = 8
Choose the output format:
1) In console
2) In file
1
Quation:  x^2 - 3x + 2 Solution:  4.9921875
f( 4.9921875 ) =  11.94537353515625
iterations =  8
```

Пример 1

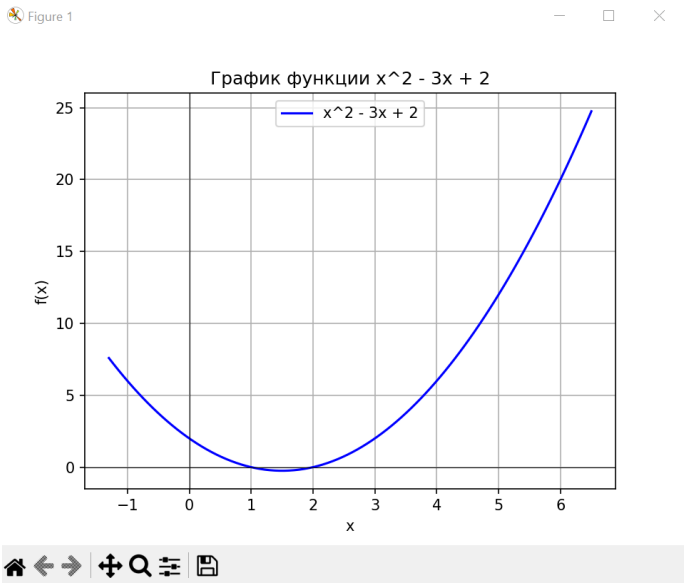

Рис. 2: График выбранной функции

9 Вывод

1. Метод половинного деления (метод дихотомии)

Идея метода:

- Начальный интервал изоляции корня делится пополам, получая начальное приближение к корню.
- Вычисляется значение функции в средней точке интервала.
- В зависимости от знаков функции на концах интервала выбирается новый интервал, который также делится пополам. Этот процесс продолжается до достижения заданной точности.

Скорость сходимости: Линейная

- Линейная сходимость означает, что ошибка между текущим приближением и истинным решением уменьшается линейно на каждом шаге итерационного процесса.
- Число итераций n для достижения точности ϵ оценивается как $n \approx \log_2 \left(\frac{b_0 - a_0}{\epsilon}\right)$.

Преимущества:

- Простота и надежность метода.
- Не требует дифференцируемости функции, достаточно её непрерывности.
- Абсолютная сходимость (гарантированное приближение к истинному решению).

Недостатки:

- Медленный метод из-за линейной сходимости.
- Если интервал содержит несколько корней, то неясно, к какому корню метод будет сходиться.

2. Метод Ньютона

Идея метода:

- Использует касательную для нахождения корня. Начальное приближение уточняется с помощью уравнения касательной в данной точке.
- Рабочая формула метода: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$.

Скорость сходимости: Быстрая, квадратичная

- Квадратичная сходимость означает, что ошибка уменьшается квадратично на каждом шаге, то есть, при каждом шаге точность удваивается.

Преимущества:

- Быстрая сходимость, особенно при хорошем начальном приближении.
- Высокая эффективность для хорошо дифференцируемых функций.

Недостатки:

- Требует дифференцируемости функции и вычисления производной на каждом шаге.
- Зависимость от выбора начального приближения. При плохом выборе сходимость может ухудшаться.

3. Метод простой итерации

Идея метода:

- Уравнение $f(x) = 0$ приводится к эквивалентному виду $x = \varphi(x)$.
- Итерационный процесс осуществляется по формуле: $x_{i+1} = \varphi(x_i)$.

Скорость сходимости: Линейная

- Линейная сходимость аналогична методу половинного деления, что означает медленное уменьшение ошибки при каждом шаге.

Преимущества:

- Простота реализации.
- Подходит для уравнений, которые можно привести к виду $x = \varphi(x)$.

Недостатки:

- Медленная сходимость.
- Необходимость выбора начального приближения из малой окрестности корня для обеспечения сходимости.
- Может не сходиться, если $\varphi'(x) \approx 1$.