Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

Лабораторная работа № 6

«Решение нелинейных уравнений и систем нелинейных уравнений методом Ньютона»

по курсу «Численные методы»

Выполнил:

студент группы ИУ9-61Б

Окутин Денис

Проверила:

Домрачева А. Б.

Москва, 2024

1. Постановка задачи

1.1 Сравнение приближённых методов решения нелинейных уравнений

Дано: $f(x) = x^n + x^{n-1} + \ldots + x^0$

Задание:

- нарисовать график функции f(x) и найти отрезки, где функция имеет простые корни и отличные от нуля две первые производные;

- найти с точностью 0, 001 все корни уравнения f (x) = 0 методом деления отрезка пополам и методом Ньютона; определить число приближений в каждом случае.

- сравнить полученные результаты.

Индивидуальный вариант: $f(x) = x^3 - 5 * x - 1$

1.2 Сравнение приближённых методов решения нелинейных уравнений

Дано: $f(x) = x^n + x^{n-1} + \cdots + x^0$

Задание:

- решить систему нелинейных уравнений графически и принять полученное решение за начальное приближение;

- решить систему методом Ньютона с точностью ε = 0.01.

Индивидуальный вариант: $\cos(y - 1) + x = 0.5$

$$y - \cos(x) = 3$$

2. Основные теоретические сведения

2.1 Сравнение приближённых методов решения нелинейных уравнений

Пусть на отрезке [a, b] нелинейное уравнение f(x) = 0 имеет единственный простой корень, т.е. f(a)f(b) < 0.

В методе деления отрезка пополам поиск корня функции f(x) начинается с деления отрезка [a, b] пополам точкой x = (a+b)/2 . Из двух получившихся отрезков выбирается тот, где находится корень уравнения.

Обозначим этот отрезок $[a_1,b_1]$. Если $f(a)f(x) < 0$, то это отрезок $[a,x]$, иначе — $[x,b]$. Отправляясь от отрезка $[a_1, b_1]$ вдвое меньшей длины, опять находим середину отрезка $x = (a1+b1)/2$ и определяем по описанному алгоритму отрезок $[a_2, b_2]$ и т.д. На k-м шаге длина получившегося отрезка $[a_k , b_k]$ будет равна $b_k - a_k = (b-a)/2$. Процесс продолжается, пока $b_k - a_k > 2\varepsilon$, где ε — требуемая точность нахождения корня. Тогда приближённое значение корня уравнения $x = (ak+bk)/2$.

Метод Ньютона решения нелинейного уравнения является итерационным. Для получения (k+1)-й итерации метода x_{k+1} из точки $(x_k,f(x_k))$, лежащей на графике функции, проводится касательная. Точка пересечения касательной с осью абсцисс и есть следующее, (k + 1)-е приближение к корню уравнения. Алгебраически метод Ньютона сводится к рекуррентной зависимости: $x_{k+1} = x_k - (f(x_k) /f'(x_k))$ $k = 0,1,....$ $f'(x_k)$.

Достаточным условием сходимости метода является отличие от нуля первых двух производных функции $f(x)$ на отрезке $[a, b]$. В качестве начального приближения x_0 выбирается тот конец отрезка, где знак функции совпадает со знаком второй производной. Заданная погрешность ε считается достигнутой, если $f(x_k)f(x_k + \mathrm{sgn}(x_k - x_{k-1})\varepsilon) < 0$.

2.2 Решение систем нелинейных уравнений методом Ньютона

Пусть $\vec{f}(\vec{x}) = \vec{0}$, где $\vec{x} = (x_1, . . . , x_n)$ — вектор неизвестных; $\vec{f} = (f_1,...,f_n)$ — вектор-функция. Выбрав начальное приближение $\vec{x}^0 = (x^0_1, . . . , x^0_n)$ к решению системы, следующие приближения в методе Ньютона строим по рекуррентной зависимости $\vec{x}^{k+1}=\vec{x}^k-[\vec{f}'(\vec{x}^k)]^{-1}\vec{f}(\vec{x}^k)$, $k=0,1,....$ Здесь $\vec{f}'(\vec{x}^k)$ — матрица Якоби системы, являющаяся производной вектор- функции $\vec{f}(\vec{x})$ в точке $\vec{x}^k$. Мы предполагаем, что матрица Якоби обратима в достаточно большой окрестности точного решения системы.

Для решения системы нелинейных уравнений с заданной точностью ε необходимо сравнить ε с погрешностью k-го приближения

$||\vec{x}^k - \vec{x}^{k-1}|| = \max |x^k - x^{k-1}|.$ $i \le i \le n$ i i

Метод Ньютона сходится, если все функции $f_i(x_1, . . . , x_n)$ дважды непре- рывно дифференцируемы по всем переменным, и начальное приближение $\vec{x}^0$ находится достаточно близко к точному решению системы. Рецепта выбора начального приближения при n > 1 нет, поэтому желательно оценить, хотя бы грубо, значение точного решения, например, решив систему графически.

3. Реализация

Листинг 1. Сравнение приближённых методов решения нелинейных уравнений

```
import numpy as np
import matplotlib.pyplot as plt


def func(x):
    return x ** 3 - 5 * x - 1


def bisect_root(a, b, tolerance=1e-6):
    assert func(a) * func(b) < 0, "The function must change sign between a and b"

    while abs(b - a) > tolerance:
        mid = (a + b) / 2
        if func(mid) == 0:
            return mid
        elif func(a) * func(mid) < 0:
            b = mid
        else:
            a = mid
    return (a + b) / 2


def derivative(x):
    return 3 * x ** 2 - 5


def newton_root(initial_guess, tolerance=1e-6):
    x = initial_guess

    while abs(func(x)) > tolerance:
        x = x - func(x) / derivative(x)

    return x

def show_func():
    x = np.linspace(-3, 3, 100)  # генерируем значения x от -3 до 3
```

```
    y = func(x)  # вычисляем соответствующие значения функции

    plt.plot(x, y, label='f(x) = x^3 - 5x - 1')
    plt.axhline(y=0, color='k', linestyle='--')  # добавляем горизонтальную линию y=0
для наглядности
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('График функции f(x) = x^3 - 5x - 1')
    plt.grid(True)
    plt.legend()
    plt.show()


def main():
    show_func()
    xs = [-3, -2, -1, 1, 2, 3]
    res_bisec = []
    i = 0
    print("Метод деления пополам:")
    while i < len(xs):
        res_bisec.append(bisect_root(xs[i],xs[i+1]))
        i += 2
    print(res_bisec)


    print()

    print("Метод Ньютона:")
    roots = [-2,0,2]
    res_newton = []
    for root in roots:
        res_newton.append(newton_root(root))
    print(res_newton)

    print()
    print("Разность:")
    razn=[]
    for i in range(len(res_newton)):
        razn.append(abs(res_newton[i]-res_bisec[i]))
    print(razn)

if __name__ == '__main__':
    main()
```

Листинг 2. Решение систем нелинейных уравнений методом Ньютона

```
import math

import numpy as np

eps = 0.01


def jacobian_matrix(f, x):
    n = len(x)
    J = np.zeros((n, n))
    h = 1e-6

    for i in range(n):
        x_h = x.copy()
        x_h[i] += h
        J[:, i] = (f(x_h) - f(x)) / h
```

```
    return J


def newton_method(f, x0, max_iter=1000):
    x = x0
    for _ in range(max_iter):
        J = jacobian_matrix(f, x)
        dx = np.linalg.solve(J, -f(x))
        x += dx
        if np.linalg.norm(dx) < eps:
            return x
    return x


def equations(vars):
    x, y = vars
    eq1 = math.cos(y - 1) + x - 0.5
    eq2 = y - math.cos(x) - 3
    return np.array([eq1, eq2])


# аналитическое решение [1,207 3,356]
initial_guess = np.array([1.0, 3.0])
sol = newton_method(equations, initial_guess)

x_sol, y_sol = sol

print(f"Решение: x = {x_sol}, y = {y_sol}")
```

4. Результаты

Сравнение приближённых методов решения нелинейных уравнений:

```
Метод деления пополам:
[-2.1284193992614746, -0.20163965225219727, 2.3300585746765137]

Метод Ньютона:
[-2.1284190810970376, -0.20163967572339103, 2.3300587397822348]

Разность:
[3.1816443701870867e-07, 2.3471193766333442e-08, 1.6510572109496024e-07]
```

Решение систем нелинейных уравнений методом Ньютона

```
Аналитическое решение: x = 1.207, y = 3.356
Метод Ньютона: x = 1.2069068181457039, y = 3.355911738933581
Погрешность: x = 9.318185429618708e-05, y = 8.826106641901532e-05
```

5. Вывод

В ходе выполнения лабораторной работы был изучен и реализован метод Ньютона для решения нелинейных уравнений и систем нелинейных уравнений, а также данный метод был сравнён с методом деление отрезка пополам, оказавшись чуть более точным.