

## **Отчет для программы по численным методам – Худобин В. О.**

**Задача №4:** Решение системы нелинейных уравнений с внешними итерациями по Зейделю и внутренними по Ньютону.

**Язык программирования:** Python

**Постановка задачи:** Задача заключается в решении системы нелинейных уравнений вида:

$$f_1(x_1, x_2, \dots, x_n) = 0, \quad f_2(x_1, x_2, \dots, x_n) = 0, \dots, \quad f_n(x_1, x_2, \dots, x_n) = 0$$

Необходимо реализовать два численных метода для нахождения решения этой системы:

1. Метод Ньютона для нахождения корней нелинейной системы.
2. Метод Зейделя с внутренним методом Ньютона для ускорения сходимости.

**Метод решения:** Для решения системы нелинейных уравнений используется два метода:

1. Метод Ньютона: Этот метод использует итерационную процедуру, где на каждом шаге вычисляется якобиан системы и решается линейная система для нахождения приращения переменных.
2. Метод Зейделя с внутренним методом Ньютона: Для ускорения сходимости применяется метод Зейделя, где на каждом шаге используется метод Ньютона для решения системы уравнений.

**Алгоритм решения:**

1. Чтение данных: Программа читает уравнения, начальные приближения и переменные из файла input.txt.
2. Парсинг уравнений: Уравнения анализируются с использованием регулярных выражений, и для них генерируются соответствующие функции.
3. Решение системы уравнений:
  - Для метода Ньютона вычисляются значения функций в текущей точке и якобиан, после чего решается линейная система.
  - Для метода Зейделя на каждом шаге используется метод Ньютона для обновления значений переменных.
4. Запись результата: Полученные решения записываются в файл output.txt.

**Реализация программы:** Программа состоит из нескольких функций:

- `read_equations(filename)`: читает уравнения из файла.
- `parse_equations(equations)`: анализирует уравнения и преобразует их в функции.
- `jacobian(f, X, sym_vars)`: вычисляет якобиан системы.
- `newton_method(f, X, sym_vars)`: применяет метод Ньютона для нахождения решения системы.
- `gauss_seidel_method(f, initial_guess, sym_vars)`: реализует метод Зейделя с внутренним методом Ньютона.
- `write_output(filename, solution, sym_vars, precision)`: записывает результат в файл.

Весь код программы:

```

import numpy as np
import sympy as sp
import re

def read_equations(filename):
    with open(filename, 'r') as file:
        equations = file.readlines()
    return [equation.strip() for equation in equations]

def parse_equations(equations):
    symbols = set()
    for eq in equations:
        symbols.update(re.findall(r'[a-zA-Z]+', eq))
    symbols = list(symbols)
    sym_vars = sp.symbols(symbols)

    functions = []

    for eq in equations:
        eq = re.sub(r'(\d)([a-zA-Z])', r'\1*\2', eq)
        eq = re.sub(r'([a-zA-Z])(\d)', r'\1*\2', eq)

        try:
            sym_eq = sp.sympify(eq)
            f = sp.lambdify(sym_vars, sym_eq, 'numpy')
            functions.append(f)
        except Exception as e:
            print(f"Ошибка при разборе уравнения: {eq}. Ошибка: {e}")
            raise

    return functions, sym_vars

def jacobian(f, X, sym_vars):
    jac = np.zeros((len(sym_vars), len(sym_vars)))
    h = 1e-6
    for i in range(len(sym_vars)):
        for j in range(len(sym_vars)):
            X1 = X.copy()
            X1[j] += h
            jac[i, j] = (f[i](X1) - f[i](X)) / h
    return jac

def newton_method(f, X, sym_vars):
    max_iter = 100
    tol = 1e-6

```

```

for _ in range(max_iter):
    F = np.array([eq(*X) for eq in f])
    J = jacobian(f, X, sym_vars)
    delta_X = np.linalg.solve(J, -F)
    X = X + delta_X
    if np.linalg.norm(delta_X) < tol:
        return X
raise ValueError("Метод Ньютона не сошелся")

def gauss_seidel_method(f, initial_guess, sym_vars,
max_outer_iter=50, tol=1e-6):
    X = np.array(initial_guess, dtype=float)
    for _ in range(max_outer_iter):
        X_new = X.copy()
        X_new = newton_method(f, X_new, sym_vars)
        if np.linalg.norm(X_new - X) < tol:
            return X_new
    X = X_new
    raise ValueError("Метод Зейделя не сошелся")

def write_output(filename, solution, sym_vars, precision=6):
    with open(filename, 'w') as file:
        for var, sol in zip(sym_vars, solution):
            file.write(f"{var} = {sol:.{precision}f}\n")

def main():
    equations = read_equations('input.txt')
    functions, sym_vars = parse_equations(equations)
    initial_guess = [0.0] * len(sym_vars)
    solution = gauss_seidel_method(functions, initial_guess,
sym_vars)
    write_output('output.txt', solution, sym_vars)

if __name__ == "__main__":
    main()

```

**Тестовые данные:** Для проверки программы используются следующие тестовые данные:

$$\begin{aligned}
 6x + 3y &= 27 \\
 -2x + 14y &= 12
 \end{aligned}$$

**Результат:**

$$\begin{aligned}
 x &= 3.800000 \\
 y &= 1.400000
 \end{aligned}$$

$$\begin{aligned}6 \cdot x + 3 \cdot y &= 27 \\ -2 \cdot x + 14y &= 12\end{aligned}$$

Решить подстановкой



Найдем первую переменную в одном из уравнений, затем подставим результат в другое уравнение.

В виде точки:

$$\left(\frac{19}{5}, \frac{7}{5}\right)$$

Форма уравнения:

$$x = \frac{19}{5}, y = \frac{7}{5}$$

[Нажмите для просмотра решения...](#)

**Заключение:** Программа успешно решает систему нелинейных уравнений, используя методы Ньютона и Зейделя с внутренним методом Ньютона для ускорения сходимости. Применение метода Зейделя значительно улучшает эффективность решения, особенно для сложных систем.