Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Образовательная программа
«Прикладная математика»

ОТЧЕТ
По лабораторной работе № 3

По предмету

«Численные методы»

По теме

«Решение систем алгебраических уравнений итерационными методами»

Выполнил
студент группы БПМ211
Кудряшов Максим Дмитриевич

Проверил
Брандышев Петр Евгеньевич

Москва - 2024

Содержание

1 Метод Ньютона (№ 4.1.8)

1.1 Формулировка задания

Найти с точностью $\epsilon = 10^{-6}$ все корни системы нелинейных уравнений:

$$f_1(x_1,x_2) = 0$$

$$f_2(x_1,x_2) = 0$$

Использовать метод Ньютона для системы нелинейных уравнений, найти корни с помощью встроенного функционала решения уравнений.

1. Используя встроенные функции, локализовать корни системы уравнений графически.

2. Написать программу-функцию, вычисляющую корень системы двух нелинейных уравнений по методу Ньютона с точностью ε. Предусмотреть подсчет количества итераций. Для решения соответствующей системы линейных алгебраических уравнений использовать встроенную функцию.

3. Используя написанную программу, вычислить все корни заданной системы с точностью ε

4. Используя встроенные функции, найти все корни системы с точностью ε. Сравнить с результатами, полученнными в п. 3.

1.2 Код на Python

1.2.1 Построение графика

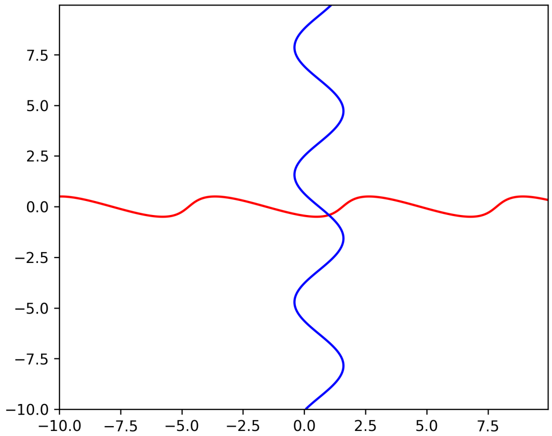
```
import numpy as np
import matplotlib.pyplot as plt

x1, x2 = np.meshgrid(np.arange(-10, 10, 0.05), np.arange(-10, 10, 0.05))
plt.figure(figsize=(6, 5))
plt.contour(x1, x2, np.cos(x1 + x2) + 2 * x2, [0], colors=['red'])
plt.contour(x1, x2, x1 + np.sin(x2) - 0.6, [0], colors=['blue'])
plt.savefig('nonlinear_newton.png', dpi=300)
plt.show()
```

1.2.2 Решение системы методом Ньютона

```
import numpy as np
import sympy as sp
import scipy
import matplotlib.pyplot as plt

x1 = sp.Symbol('x1')
x2 = sp.Symbol('x2')

f1 = sp.cos(x1 + x2) + 2 * x2
f2 = x1 + sp.sin(x2) - 0.6

```

```python
F = sp.Matrix([f1, f2])


def eq(vars):
    x1, x2 = vars
    f1 = sp.cos(x1 + x2) + 2 * x2
    f2 = x1 + sp.sin(x2) - 0.6
    return [f1, f2]


def calculate_root(F, X_num: np.array, eps: float = 0.000001) -> np.array:
    iter_count = 0
    jacobian = F.jacobian([x1, x2]).inv() * F
    while True:
        jacobian_num = jacobian.subs([(x1, X_num[0]), (x2, X_num[1])])
        X_num -= np.array(jacobian_num, dtype=float).flatten()
        iter_count += 1
        norm = np.linalg.norm(np.array(F.subs([(x1, X_num[0]), (x2, X_num[1])]), dtype=float).flatten())
        if norm < eps:
            break
    return X_num, iter_count


X1 = np.array([2.5, 0.1])
X2 = np.array([-4.5, -0.1])

X1_my, count1 = calculate_root(F, X1, 0.000001)
X2_my, count2 = calculate_root(F, X2, 0.000001)


X1_scipy = scipy.optimize.fsolve(eq, X1)
X2_scipy = scipy.optimize.fsolve(eq, X2)

print(f'{X1_my = }')
print(f'Количество итераций: {count1}')
print()

print(f'{X1_my = }')
print(f'Количество итераций: {count1}')
print()

print(f'{X1_scipy = }')
print(f'{X2_scipy = }')

plt.plot(*X1_my, marker='o', color='green', ls='')
x1, x2 = np.meshgrid(np.arange(-10, 10, 0.05), np.arange(-10, 10, 0.05))
plt.contour(x1, x2, np.cos(x1 + x2) + 2 * x2, [0], colors=['red'])
plt.contour(x1, x2, x1 + np.sin(x2) - 0.6, [0], colors=['blue'])
plt.savefig('nonlinear_newton_with_point.png', dpi=300)
plt.show()
```

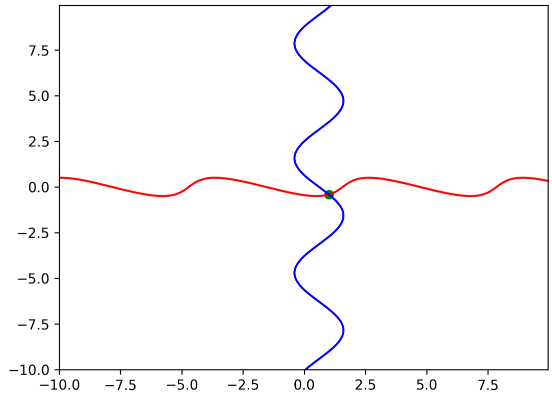
1.3 Результаты



Видно, что корень находится в пределах (0, 2.5) по оси x, и (-1, 1) по оси y.

Результат работы программы:

```
X1_my = array([ 1.00410184, -0.41599671])
Количество итераций: 4

X1_my = array([ 1.00410184, -0.41599671])
Количество итераций: 4

X1_scipy = array([ 1.00410184, -0.41599671])
X2_scipy = array([ 1.00410184, -0.41599671])
```



2 Расстояния от поверхности до точки (№ 4.5.2)

2.1 Формулировка задания

Дана система уравнений $Ax = b$. Найти решение системы с помощью метода Гаусса. Выполнить 10 итераций по методу Зейделя. Принимая решение, полученное с помощью метода Гаусса за точное, найти величину абсолютной погрешности итерационного решения.

2.2 Код на Python

3 Метод Зейделя (№ 5.1.8)

3.1 Формулировка задания

Дана система уравнений $Ax = b$. Найти решение системы с помощью метода Гаусса. Выполнить 10 итераций по методу Зейделя. Принимая решение, полученное с помощью метода Гаусса за точное, найти величину абсолютной погрешности итерационного решения.

1. Задать матрицу системы A и вектор правой части b. Найти решение системы Ax=b с помощью метода Гаусса.

2. Преобразовать систему $Ax = b$ к виду $x = Bx + c$, удобному для итераций. Проверить выполнение достаточного условия сходимости итерационных методов $\|B\|_\infty < \infty$

3. Написать программу-функцию zeid, решающую систему уравнений с помощью метода Зейделя, выполнить 10 итераций по методу Зейделя; взять любое начальное приближение. Принимая решение, полученное в п. 1 за точное, найти величину абсолютной погрешности итерационного решения.

4. Взять другое начальное приближение. Объяснить полученные результаты.

3.2 Код на Python

```
import numpy as np


def zeid(A, b, x0, n):
    B = np.empty(A.shape, dtype=float)
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            B[i, j] = - A[i, j] / A[i, i] if i != j else 0

    c = np.empty(b.shape, dtype=float)
    for i in range(c.shape[0]):
        c[i] = b[i] / A[i, i]

    # B1 = np.zeros(B.shape)
    # B2 = np.zeros(B.shape)
    # for i in range(A.shape[0]):
    #     for j in range(A.shape[1]):
    #         if j < i:
    #             B1[i, j] = B[i, j]
    #         if j > i:
    #             B2[i, j] = B[i, j]

    x = x0
    for _ in range(n):
        x_new = np.zeros(x.shape, dtype=float)
```

```
        for i in range(B.shape[0]):  # x_new = B1 @ x_new + B2 @ x + c
            x_new[i] = np.sum(B[i][:i] * x_new[:i]) + np.sum(B[i][i:] * x[i:]) + c[i]
        x = x_new
    return x, B





A = np.array([[118.8, -14, -5, -89.1],
              [-59.4, 194, 5, 128.7],
              [148.5, 12, -310, 148.5],
              [0, 18.5, 90, -108.9]])

b = np.array([92.5, -340.1, -898, 184.1])

x_true = np.linalg.solve(A, b)

x1 = np.zeros(b.shape[0])
x2 = np.ones(b.shape[0])

x1_zaid, B = zeid(A, b, x1, 10)
x2_zaid, _ = zeid(A, b, x2, 10)

error1_abs = np.linalg.norm(x_true - x1_zaid, ord=np.inf)
error2_abs = np.linalg.norm(x_true - x2_zaid, ord=np.inf)

print(f'Решение методом Гаусса: {x_true}')
print()

print(f'Начальная точка: {x1}')
print(f'Решение методом Зейделя: {x1_zaid}')
print(f'Абсолютная погрешность: {error1_abs}')
print()

print(f'Начальная точка: {x2}')
print(f'Решение методом Зейделя: {x2_zaid}')
print(f'Абсолютная погрешность: {error2_abs}')

print(f'Норма матрицы B: {np.linalg.norm(B, ord=np.inf)}')
```

3.3 Результаты

```
Решение методом Гаусса: [ 1.90875981 -2.36310552  4.49843079  1.62572378]

Начальная точка: [0. 0. 0. 0.]
Решение методом Зейделя: [ 1.90048704 -2.35749698  4.48910538  1.61896961]
Абсолютная погрешность: 0.00932540991025288
```

```
Начальная точка: [1. 1. 1. 1.]
Решение методом Зейделя: [ 1.90673816 -2.36173495  4.49615191  1.62407324]
Абсолютная погрешность: 0.0022788803781175204

Норма матрицы B: 0.996774193548387
```

4 Метод релаксации (№ 5.5.1)

4.1 Формулировка задания

Дана система уравнений $Ax = b$, где A – симметричная положительно определенная матрица. Найти решение системы с точностью $\varepsilon = 10^-5$ с помощью метода релаксации (для этого модифицировать функцию zeid из задачи 5.2, реализующую метод Зейделя). Определить экспериментально параметр релаксации ω, при котором точность ε достигается при наименьшем числе итераций. Построить график зависимости числа итераций от параметра релаксации.

4.2 Код на Python

```python
import numpy as np


def relax(A, b, x0, eps, w):
    B = np.empty(A.shape, dtype=float)
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            B[i, j] = - A[i, j] / A[i, i] if i != j else 0

    c = np.empty(b.shape, dtype=float)
    for i in range(c.shape[0]):
        c[i] = b[i] / A[i, i]

    B1 = np.zeros(B.shape)
    B2 = np.zeros(B.shape)
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            if j < i:
                B1[i, j] = B[i, j]
            if j > i:
                B2[i, j] = B[i, j]

    x = x0
    while True:
        x_new = np.zeros(x.shape, dtype=float)
        for i in range(B.shape[0]):  # x_new = B1 @ x_new + B2 @ x + c
            x_new[i] = np.sum(B[i][:i] * x_new[:i]) + np.sum(B[i][i:] * x[i:]) + c[i]
            x_new[i] = w * x_new[i] + (1 - w) * x[i]
        norma = np.linalg.norm(x_new - x)
        x = x_new

```

```
        eps_new = ((1 - np.linalg.norm(B, ord=np.inf)) * eps) / np.linalg.norm(B2, ord=np.inf)
        if norma < eps_new:
            break
    return x


A = np.array([[118.8, -14, -5, -89.1],
              [-59.4, 194, 5, 128.7],
              [148.5, 12, -310, 148.5],
              [0, 18.5, 90, -108.9]])

b = np.array([92.5, -340.1, -898, 184.1])

x_true = np.linalg.solve(A, b)
x_relax = relax(A, b, np.zeros(b.shape[0]), 0.00001, 1.25)

print(f'Решение методом Гаусса: {x_true}')
print(f'Решение системы методом релаксации: {x_relax}')
```

4.3 Результаты