ИНСТИТУТ ТРАНСПОРТА И СВЯЗИ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ТЕЛЕКОММУНИКАЦИЙ

Лабораторная работа №1
По дисциплине
«Численные методы и прикладное программирование»

Тема:
«Методы решения системы линейных уравнений.
Число обусловленности матрицы»

Работу выполнили:

Дзенис Ричард
Кобелев Денис
Якушин Владислав

Рига
2017 г.

1 Формулировка задания

- Реализовать программным путём метод исключения Гаусса и итерационный метод Гаусса-Зейделя.
- Результат работы программы проверить с помощью предоставленных примеров.
- Ручным или программным путём рассчитать число обусловленности матриц для примеров (3) и (5).
- Для расчёта обусловленности выбрать Манхэттенскую или Евклидову норму.
- Составить отчёт с результатами вычислений и выводами, содержащими сравнение двух реализованных методов, а так же объяснить значения полученный при вычислении числа обусловленности матриц.

1.1 Примеры

$$\begin{cases} x_1 - 2x_2 + x_3 = 2 \\ 2x_1 - 5x_2 - x_3 = -1 \\ -7x_1 + x_3 = -2 \end{cases} \tag{1}$$

$$\begin{cases} 2x_1 - x_2 - x_3 = 5 \\ x_1 + 3x_2 - 2x_3 = 7 \\ x_1 + 2x_2 + 3x_3 = 10 \end{cases} \tag{2}$$

$$\begin{cases} 5x_1 - 5x_2 - 3x_3 + 4 = -11 \\ x_1 - 4x_2 + 6x_3 - 4 = -10 \\ -2x_1 - 5x_2 + 4x_3 - 5 = -12 \\ -3x_1 - 3x_2 + 5x_3 - 5 = 8 \end{cases} \tag{3}$$

$$\begin{cases} 8x_1 + 5x_2 + 3x_3 = 30 \\ -2x_1 + 8x_2 + x_3 = 15 \\ x_1 + 3x_2 - 10x_3 = 42 \end{cases} \tag{4}$$

$$\begin{cases} 0.78x_1 + 0.563x_2 = 0.217 \\ 0.913x_1 + 0.659x_2 = 0.254 \end{cases} \tag{5}$$

2 Метод исключения Гаусса с ведущим элементом

2.1 Листинг

```
// Input: linear system 'Ax=b',
//        where Ab is matrix A combined with vector b.
//        Size of Ab is (n, n+1).
// Output: x.
void solve(double **Ab, ssize_t n, double *x) {
    ssize_t base, r, c;
    double sum;

    // Forward elimination
    for (base = 0; base < n; base++) {
        // Select maximal element in the column;
        // optimize by skipping base row.
        c = base;
        size_t leading_row = base;
        double max_value = Ab[base][base];
        for (r = base + 1; r < n; r++) {
            double abs_val = fabs(Ab[r][c]);
            if (abs_val > max_value) {
                leading_row = r;
                max_value = abs_val;
            }
        }

        // Swap base row with with leading row
        std::swap(Ab[base], Ab[leading_row]);

        // Eliminate base column
        for (r = base + 1; r < n; r++) {
            double coef = Ab[r][base] / Ab[base][base];
            for (c = base; c <= n; c++) { // including vector B
                Ab[r][c] -= coef * Ab[base][c];
            }
        }
    }

    // Backward substitution
    for (base = n - 1; base >= 0; base--) {
        sum = 0.0;
        for (c = base + 1; c < n; c++) {
            sum += Ab[base][c] * x[c];
        }
        x[base] = (Ab[base][n] - sum) / Ab[base][base];
    }
}
```

2.2 Результаты работы алгоритма

2.2.1 Результат работы алгоритма на примере (1)

```
0.52        0.08        1.64
```

2.2.2 Результат работы алгоритма на примере (2)

```
-12.8235        -2.29412        11.7647        19.2353
```

2.2.3 Результат работы алгоритма на примере (5)

```
1        -1
```

3 Метод Гаусса-Зейделя

3.1 Листинг

```
bool solve(double **Ab, ssize_t n, double *x, double eps) {
    ssize_t i, j;
    double acc, prev_acc = HUGE_VALF;
    do {
        acc = 0.0f;
        for (i = 0; i < n; i++) {
            double denom = Ab[i][i];
            double new_xi = Ab[i][n] / denom;
            for (j = 0; j < n; j++) {
                if (i == j)
                    continue;
                new_xi -= Ab[i][j] / denom * x[j];
            }
            acc = std::fmaxf(acc, fabs(new_xi - x[i]));
            x[i] = new_xi;
        }
        if (acc >= prev_acc)
            return false;
        prev_acc = acc;
    } while (acc > eps);
    return true;
}
```

3.2 Результаты работы алгоритма с точностью 10^{-3}

3.2.1 Результат работы алгоритма на примере (1)

```
does not converge
```

3.2.2 Результат работы алгоритма на примере (3)

```
        3.8126          1.68737          0.937552
```

3.2.3 Результат работы алгоритма на примере (4)

```
       2.99984          2.99995          -3.00003
```

4 Экспериментальное определение числа обусловленности матрицы

4.1 Часть листинга для нахождения числа обусловленности

```
X = solve(A, B) # find X in AX=B

max_B_id = np.argmax(B) # find maximal element in vector B

# create delta B vector with one non-zero element, which
# equals to 1% of the maximal element in vector B.
deltaB = np.zeros(B.shape)
deltaB[max_B_id] = B[max_B_id] * 0.01

deltaX = solve(A, B + deltaB) # find deltaX in A * deltaX = B + deltaB

# Calculate and print condition number
print("cond(A) is greater or equals to: ")
print(norm(deltaX) / norm(X) * norm(B) / norm(deltaB))
```

4.2 Полученные результаты

4.2.1 Полученные результаты для метода Гаусса

4.2.1.1 Пример (1)

```
cond(A) is greater or equals to:
151.20581277
```

4.2.1.2 Пример (2)

```
cond(A) is greater or equals to:
260.247717314
```

4.2.1.3 Пример (5)

```
cond(A) is greater or equals to:
227109.909242
```

4.2.2 Полученные результаты для метода Гаусса-Зейделя

4.2.2.1 Пример (3)

```
cond(A) is greater or equals to:
131.761504857
```

4.2.2.2 Пример (4)

```
cond(A) is greater or equals to:
127.904080532
```

5 Выводы