Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиоэлектроника и лазерная техника (РЛ)»

Кафедра «Технология приборостроения (РЛ6)»

Лабораторная работа №1

"Метод Гаусса решения СЛАУ. Оценка числа обусловленности матрицы."

по дисциплине "Численные методы"

Вариант №1

Выполнили студенты группы РЛ6-71

Филимонов С.В.

Преподаватель Чигирева О.Ю.

Москва, 2023

**Исходные данные:**

1. Хорошо обусловленная матрица:

Вектор точного решения:

2. Плохо обусловленная матрица:

Вектор точного решения:

**Цель работы:** изучение метода Гаусса численного решения квадратной СЛАУ с невырожденной матрицей; оценка числа обусловленности матрицы и исследование его влияния на погрешность приближенного решения.

**Содержание работы:**

1. Реализовать метод Гаусса решения СЛАУ (с выбором главного элемента по столбцу).

2. Провести решение двух заданных систем линейных алгебраических уравнений методом Гаусса, вычислить нормы невязок полученных приближенных решений, их абсолютные и относительные погрешности (при расчетах пользоваться 1-нормой и inf-нормой).

3. Для каждой из систем оценить порядок числа обусловленности ее матрицы и сделать вывод о его влиянии на точность полученного приближенного решения и отвечающую ему невязку.

**Алгоритм метода Гаусса с выбором главного элемента по столбцу**

Прямой ход (алгоритм):

*Шаг 1:*  – главный элемент 1-го шага;

исключим неизвестное из уравнений с номерами , для этого разделим 1-е уравнение на и вычтем из -го () уравнения 1-е уравнение, умноженное на ;

*Шаг 2:*  – главный элемент 2-го шага;

исключим неизвестное из уравнений с номерами , для этого разделим 2-е уравнение на и вычтем из -го () уравнения 2-е уравнение, умноженное на ;

*Шаг (n-1):*  – главный элемент -го шага;

исключим неизвестное из уравнений с номером , для этого разделим -е уравнение на и вычтем из -го уравнения -е уравнение, умноженное на ;

*Шаг (n):*  – разделим -е уравнение на ;

Обратный ход (алгоритм):

из -го уравнения находим неизвестное , затем в -е уравнение подставляем найденное значение и вычисляем и т.д.

Расчетные формулы обратного хода:

*Метод Гаусса с выбором главного элемента по столбцу* эквивалентен применению метода Гаусса к системе, в которой на каждом шаге исключения неизвестного осуществляется перестановка уравнений.

На -ом шаге исключения найдем наибольший по модулю коэффициент при неизвестном :

и осуществим перестановку уравнений с номерами и .

Затем выполним исключение неизвестного согласно описанному выше алгоритму.

**Результаты расчетов:**

Хорошо обусловленная матрица:

Вектор точного и полученного решений, невязка:

1-норма невязки:

inf-норма невязки:

Погрешности полученных приближенных решений:

Абсолютная (1-норма):

Абсолютная (inf-норма):

Относительная (1-норма):

Относительная (inf-норма):

Порядок числа обусловленности матрицы:

Плохо обусловленная матрица:

Вектор точного и полученного решений, невязка:

1-норма невязки:

inf-норма невязки:

Погрешности полученных приближенных решений:

Абсолютная (1-норма):

Абсолютная (inf-норма):

Относительная (1-норма):

Относительная (inf-норма):

Порядок числа обусловленности матрицы:

**Вывод:**

Исследовав два вида матриц для СЛАУ (хорошо обусловленную и плохо обусловленную) мы подтвердили данную обусловленность матрицы, проведя расчет порядка числа обусловленности.

Чем меньше порядок числа обусловленности матрицы, тем меньше невязка и выше точность (значения абсолютной и относительной точности хорошо обусловленной матрицы меньше, чем эти же значения плохо обусловленной матрицы).

Применимость метода Гаусса можно оценить, вычислив порядок числа обусловленности до разработки алгоритма и выполнения вычислений, и, в случае большого порядка числа обусловленности (много больше единицы) следует выбрать другой метод для увеличения точности вычислений.

**Текст программы:**

void Matrix::rationingRow(size\_t r, double elem) {

for (size\_t j = 0; j < cols; ++j)

matrix[(j + (r \* cols))] /= elem;

}

void Matrix::rationingCol(size\_t c, double elem) {

for (size\_t i = 0; i < rows; ++i)

matrix[(c + (i \* cols))] /= elem;

}

void Matrix::minusRowRow(size\_t from\_i, size\_t i) {

for (size\_t j = 0; j < cols; ++j) {

matrix[(j + (from\_i \* cols))] -= matrix[(j + (i \* cols))];

}

}

Matrix Matrix::getExtendedMatrixOfTheSystem(Matrix B) const {

Matrix AB(rows, (cols + B.cols));

for (size\_t i = 0; i < AB.rows; ++i)

for (size\_t j = 0, ja = 0; j < cols; ++j, ++ja)

AB(i, j) = matrix[(ja + (i \* cols))];

for (size\_t i = 0; i < AB.rows; ++i)

for (size\_t j = cols, jb = 0; j < AB.cols; ++j, ++jb)

AB(i, j) = B(i, jb);

return AB;

}

Matrix SLEmethodGauss(const Matrix& A, const Matrix& B) {

if (A.det() == 0) // Проверка существует ли единственное решение

throw SingularMatrix();

if ((A.rows != B.rows) || (B.cols > 1) || (A.isNull()) || (B.isNull())) // проверка корректно ли вычисл.

throw DimensionMismatch(A, B);

Matrix AB = A.getExtendedMatrixOfTheSystem(B); // Расширенная матрица системы

const size\_t row = AB.getRows();

const size\_t col = AB.getCols();

for (size\_t i = 0; i < row; ++i) { // Выбор главного элемента по столбцу

for (size\_t j = i; j < row; ++j)

AB.rationingRow(j, AB(j, i)); // Деление строки, на элемент

for (size\_t j = (i + 1); j < row; ++j)

AB.minusRowRow(j, i); // Вычитание строки

}

Matrix x(row, 1); // Вычисление ответа путем обратного хода

for (size\_t i = (row - 1), k = (col - 2); ((i < row) && (col)); --i, --k) {

x(i, 0) = AB(i, (col - 1));

for (size\_t j = (i - 1); j < row; --j)

AB(j, (col - 1)) -= (AB(j, k) \* x(i, 0));

}

return x;

}