**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Крагель Алина Олеговна

**Решение систем линейных алгебраических уравнений методом Гаусса**

**(с выбором максимального элемента по столбцу)**

Отчет по лабораторной работе №1

(«Вычислительные методы алгебры»)

Студента 2 курса 10 группы

Работа сдана \_\_\_октября 2020 г. Преподаватель:

Зачтена \_\_\_\_\_\_\_\_\_ 2020 г. Горбачева Юлия Николаевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ассистент кафедры

(Подпись преподавателя) вычислительной математики

Минск 2020

**Постановка задачи**

Написать программу, которая решает систему линейных алгебраических уравнений *Ax=f* с матрицей *A* порядка *n* методом Гаусса с выбором главного элемента по столбцу, а также вычисляет определитель матрицы *det A*, обратную матрицу . Предусмотреть сообщения, предупреждающие о невозможности решения указанной задачи.

Для проведения вычислительного эксперимента необходимо решить систему размерности *n =10* . Матрицу *A* и вектор точного решения *x* заполнить случайными числами с двумя знаками после запятой из диапазона от *-10* до *10*. Правую часть *f* задать умножением матрицы *A* на вектор *x*: *f=Ax*.

В результатах выполнения тестовой задачи необходимо привести следующую информацию:

* Условие: матрица *A* (построчно), вектор *f* , точное решение *x*.
* Полученное приближенное решение *x*.
* Максимум-норма невязки .
* Максимум-норма погрешности .
* Определитель матрицы *det A*.
* Обратную матрицу (построчно) и матрицу (построчно).

**Теория**

Метод Гаусса  — классический метод решения [системы линейных алгебраических уравнений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D1%85_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9), который представляет собой метод последовательного исключения [переменных](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0), когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних, находятся все переменные системы.

Рассмотрим систему:

,

где А – невырожденная матрица системы размерности , f — столбец свободных членов, - неизвестные переменные, *n* – число неизвестных.

Совокупность значений переменных, при которых все уравнения системы обращаются в тождества, называется решением СЛАУ.

Исходная система выглядит следующим образом:

Верхний индекс - количество раз изменений данного элементов матрицы.

Будем считать, что . Мы всегда можем этого добиться, если поменяем местами некоторые уравнения системы, что и позволяет нам сделать метод Гаусса с выбором главного элемента по столбцу.

На первом шаге будем умножать первое уравнение исходной системы на коэффициент и вычитать из *i-*го уравнения исходной системы для . Тем самым исключаем из этих уравнений *x1*. Элемент называется ведущим. После первого шага исходная система примет вид:

Продолжая эту процедуру при условии, что ведущий элемент  , , через *n-1* получим следующую систему:

Элементы данной системы находятся по следующим формулам – формулам прямого хода метода Гаусса:

Для нахождения решения системы выполним обратный ход метода Гаусса:

Определитель матрицы после приведения ее к диагональному виду по методу Гаусса:

где - диагональные элементы на *(i-1)*-м шаге преобразования, а *l* – количество перестановок строк во время выполнения метода.

Обратная матрица находится при решении *n* систем из *n* неизвестных. Данные системы выглядят следующим образом:

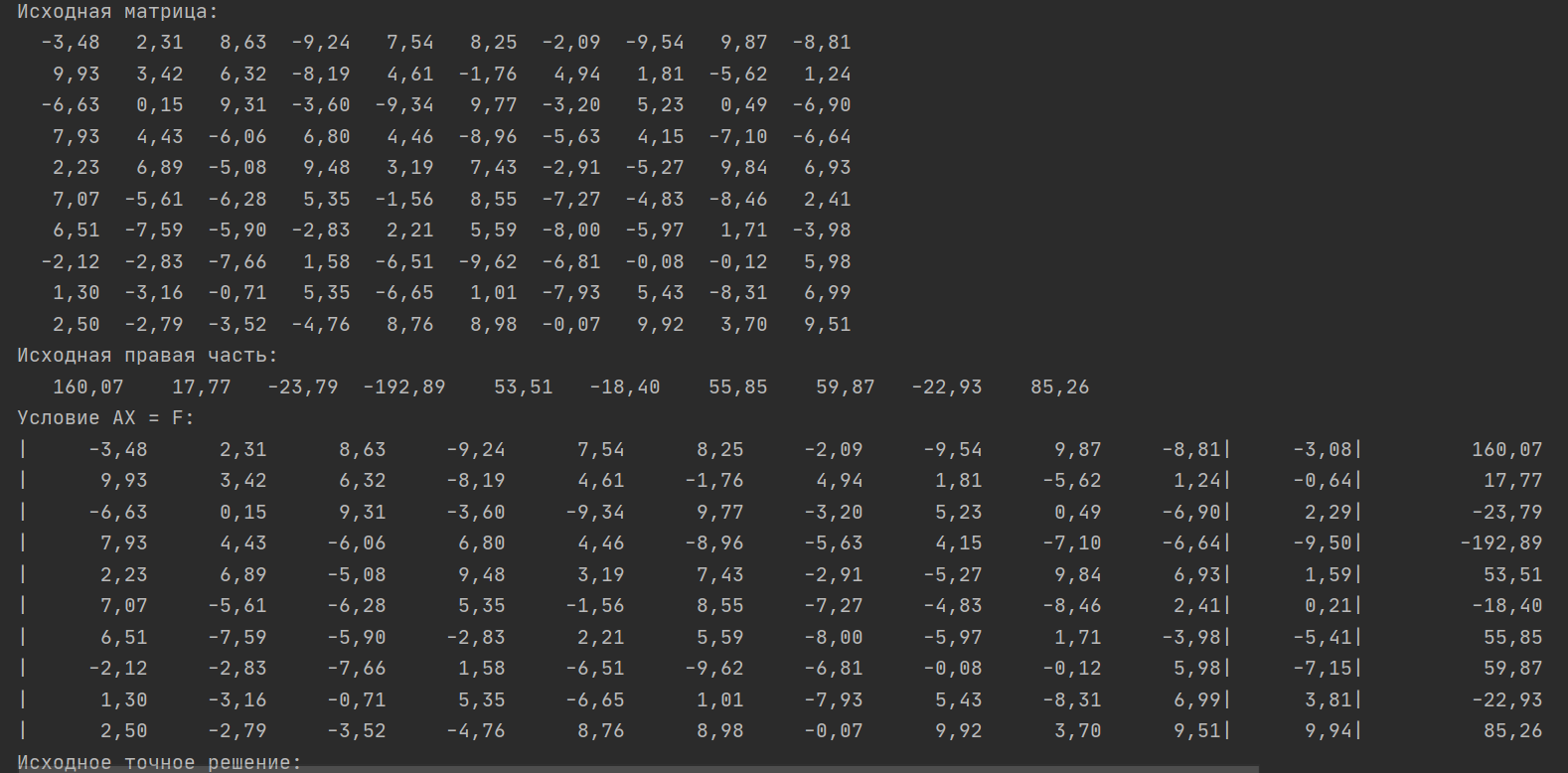
Решение каждой из систем есть один из столбцов обратной матрицы. Проверить правильность нахождения обратной матрицы можно так:

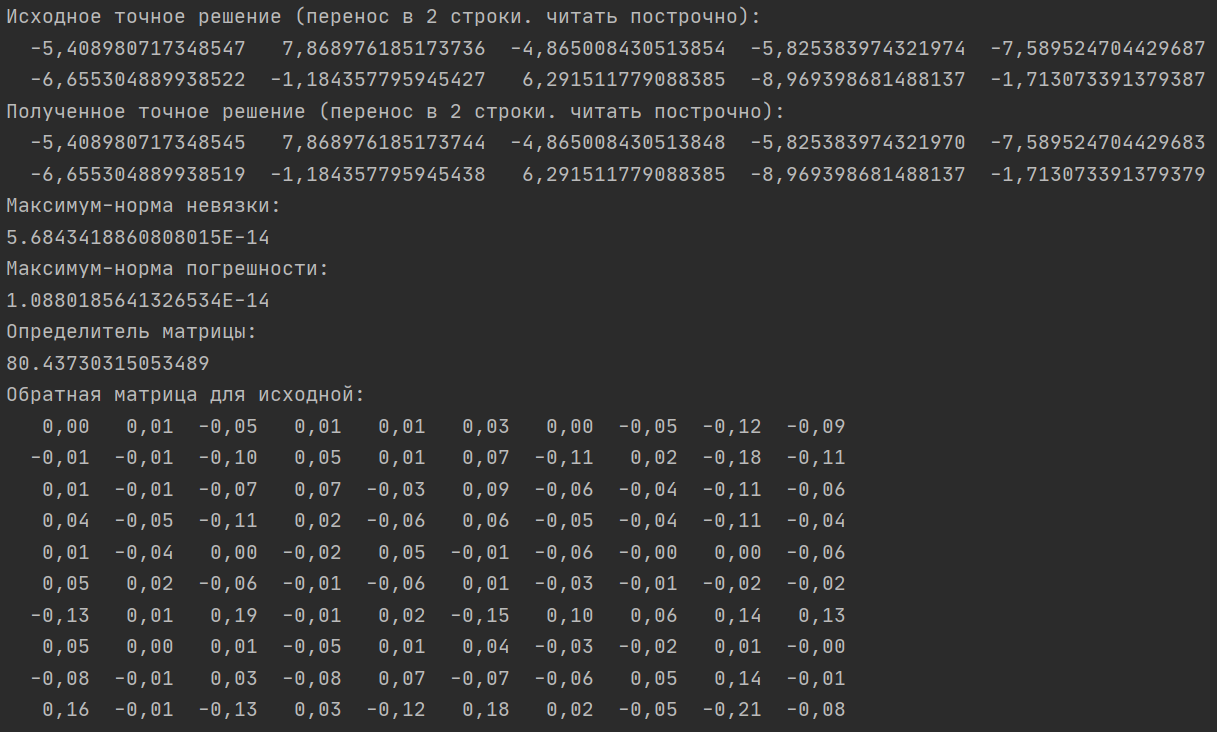
**Листинг**

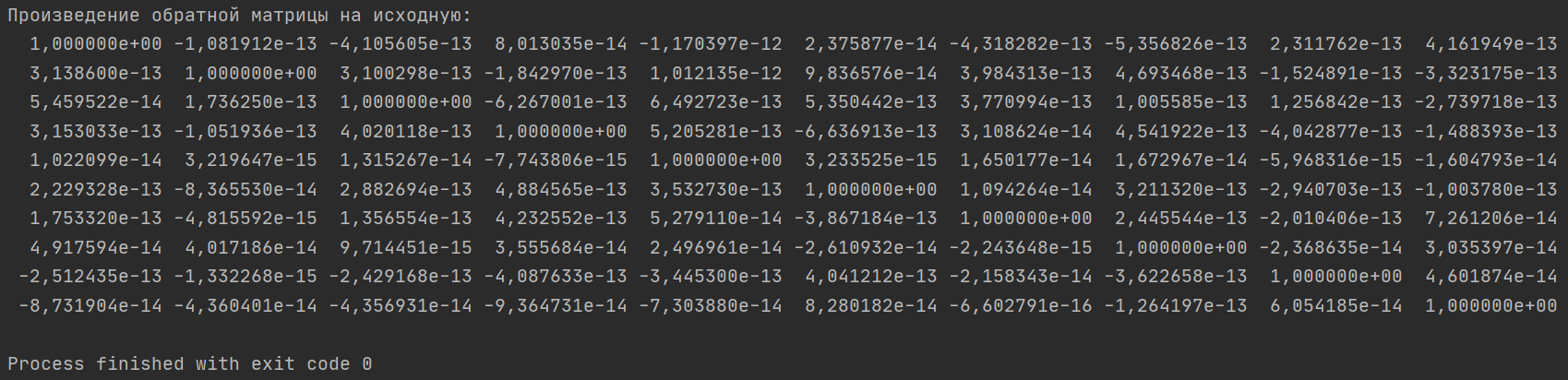
import java.util.Random;  
import java.util.Date;  
  
public class L1 {  
 public static int *n* = 10, *l* = 0;  
 public static double[] *detDiag* = new double[*n*];  
 //Метод для генерации А  
 public static double[][] MatrixGenerationA() {  
 double[][] A = new double[*n*][*n*];  
 Random myRand = new Random((new Date()).getTime());  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 A[i][j] = (myRand.nextDouble() \* (20) - 10);  
 }  
 }  
 return A;  
 }  
 //Метод для генерации X  
 public static double[] MatrixGenerationX() {  
 double[] X = new double[*n*];  
 Random myRand = new Random((new Date()).getTime());  
 for (int i = 0; i < *n*; i++) {  
 X[i] = (myRand.nextDouble() \* (20) - 10);  
 }  
 return X;  
 }  
 //Печать матрицы  
 public static void MatrixPrinting(double[][] M, int m, int p) {  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < p; j++) {  
 System.*out*.printf("%7.2f", M[i][j]);  
 }  
 System.*out*.print("\n");  
 }  
 }  
 //Печать матрицы с экспонентой  
 public static void MatrixPrintingPr(double[][] M, int m, int p) {  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < p; j++) {  
 System.*out*.printf("%20.10e", M[i][j]);  
 }  
 System.*out*.print("\n");  
 }  
 }  
 //Получение F  
 public static double[] MatrixGettingF(double[][] A, double[] X) {  
 double[] F = new double[*n*];  
 for (int i = 0; i < *n*; i++) {  
 F[i] = 0;  
 }  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 F[i] += A[i][j] \* X[j];  
 }  
 }  
 return F;  
 }  
 //Печать системы с решением  
 public static void PrintingSoLAE(double[][] A, double[] X, double[] F) {  
 System.*out*.print("Условие AX = F: \n");  
 for (int i = 0; i < *n*; i++) {  
 System.*out*.print("|");  
 for (int j = 0; j < *n*; j++) {  
 System.*out*.printf("%10.2f", A[i][j]);  
 }  
 System.*out*.printf("|%10.2f|", X[i]);  
 System.*out*.printf("%15.2f\n", F[i]);  
 }  
 }  
 //Непосредственно метод Гаусса  
 public static double[] GaussMainElementByCol(double[][] A, double[] F) {  
 double[][] SoLAE = new double[*n*][*n* + 1];  
 double[] desX = new double[*n*];  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 SoLAE[i][j] = A[i][j];  
 }  
 SoLAE[i][*n*] = F[i];  
 }  
 for (int i = 0; i < *n*; i++) {  
 double maxInCol = SoLAE[i][i];  
 int k = i;  
 //Выбираем главный элемент по столбцу  
 for (int j = i + 1; j < *n*; j++) {  
 if (Math.*abs*(SoLAE[j][i]) > Math.*abs*(maxInCol)) {  
 maxInCol = SoLAE[j][i];  
 k = j;  
 }  
 }  
 //Меняем ряды  
 if (k != i) {  
 *l*++;  
 SoLAE = *SwapRows*(SoLAE, i, k);  
 }  
 //Отнимаем от остальных выбранный  
 double mainElem = SoLAE[i][i];  
 for (int j = i + 1; j < *n*; j++) {  
 double koefRow = SoLAE[j][i] / mainElem;  
 for (int t = i; t < *n* + 1; t++) {  
 SoLAE[j][t] = SoLAE[j][t] - koefRow \* SoLAE[i][t];  
 }  
 }  
 }  
 for (int i = 0; i < *n*; i++){  
 *detDiag*[i] = SoLAE[i][i];  
 }  
 //Обратный ход  
 double s = 0;  
 desX[*n* - 1] = SoLAE[*n* - 1][*n*] / SoLAE[*n* - 1][*n* - 1];  
 for (int j = *n* - 2; j >= 0; j--) {  
 s = 0;  
 for (int t = *n* - 1; t >= j + 1; t--) {  
 s += desX[t] \* SoLAE[j][t];  
 }  
 desX[j] = (SoLAE[j][*n*] - s) / SoLAE[j][j];  
 }  
 return desX;  
 }  
 //Менять стобцы местами  
 public static double[][] SwapRows(double[][] M, int fCol, int kCol) {  
 for (int i = fCol; i < *n* + 1; i++) {  
 double temp = M[fCol][i];  
 M[fCol][i] = M[kCol][i];  
 M[kCol][i] = temp;  
 }  
 return M;  
 }  
 //Печать вектора  
 public static void ArrayPrinting(double[] M, int m) {  
 for (int i = 0; i < m; i++) {  
 System.*out*.printf("%9.2f", M[i]);  
 }  
 System.*out*.print("\n");  
 }  
 //Печать вектора с экспонентой  
 public static void ArrayPrintingPr(double[] M, int m) {  
 for (int i = 0; i < m; i++) {  
 System.*out*.printf("%20.15f", M[i]);  
 }  
 System.*out*.print("\n");  
 }  
 //Нахождение максимум-нормы невязки  
 public static double MaxNormDiscrepancy(double[][] A, double[] desX, double[] F){  
 double[]desF = new double [*n*];  
 for (int i = 0; i < *n*; i++) {  
 desF[i] = 0;  
 }  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 desF[i] += A[i][j] \* desX[j];  
 }  
 desF[i] = Math.*abs*(desF[i] - F[i]);  
 }  
 double maxND = desF[0];  
 for (int i = 0; i < *n*; i++)  
 {  
 if (desF[i] > maxND)  
 maxND = desF[i];  
 }  
 return maxND;  
 }  
 //Нахождение максимум-нормы погрешности  
 public static double MaxNormError(double[] X, double[] desX){  
 double[]errorX = new double [*n*];  
 for (int i = 0; i < *n*; i++) {  
 errorX[i] = 0;  
 }  
 for (int i = 0; i < *n*; i++) {  
 errorX[i] = Math.*abs*(X[i] - desX[i]);  
 }  
 double maxNE = errorX[0];  
 for (int i = 0; i < *n*; i++)  
 {  
 if (errorX[i] > maxNE)  
 maxNE = errorX[i];  
 }  
 return maxNE;  
 }  
 //Вычисление опредилителя  
 public static double Determinant(double[][]A, int m){  
 double det = 0;  
 for (int i = 0; i < *n*; i++){  
 det += *detDiag*[i];  
 }  
 return det \* Math.*pow*(-1, *l*);  
 }  
 //Главная функция  
 public static void main(String[] args) throws Exception {  
 double[][] A = *MatrixGenerationA*();  
 System.*out*.print("Исходная матрица: \n");  
 *MatrixPrinting*(A, *n*, *n*);  
 double[] X = *MatrixGenerationX*();  
 double[] F = *MatrixGettingF*(A, X);  
 System.*out*.println("Исходная правая часть:");  
 *ArrayPrinting*(F, *n*);  
 *PrintingSoLAE*(A, X, F);  
 double[] desX = *GaussMainElementByCol*(A, F);  
 if (*Determinant*(A, *n*) != 0) {  
 System.*out*.println("Исходное точное решение:");  
 *ArrayPrintingPr*(X, *n*);  
 System.*out*.println("Полученное точное решение:");  
 *ArrayPrintingPr*(desX, *n*);  
 System.*out*.println("Максимум-норма невязки: \n" + *MaxNormDiscrepancy*(A, desX, F));  
 System.*out*.println("Максимум-норма погрешности: \n" + *MaxNormError*(X, desX));  
 System.*out*.println("Определитель матрицы: \n" + *Determinant*(A, *n*));  
 double[][] \_A = *ReverseGaussMainElementByCol*(A);  
 System.*out*.print("Обратная матрица для исходной: \n");  
 *MatrixPrinting*(\_A, *n*, *n*);  
 double[][] fin = new double[*n*][*n*];  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 fin[i][j] = 0;  
 }  
 }  
 //Умножение обратрной мтарицы на исходную  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 for (int m = 0; m < *n*; m++)  
 fin[i][j] += \_A[i][m] \* A[m][j];  
 }  
 }  
 System.*out*.print("Произведение обратной матрицы на исходную: \n");  
 *MatrixPrintingPr*(fin, *n*, *n*);  
 }  
 else  
 System.*err*.println("Матрица вырожденная.");  
 }  
 //Нахождение обратной матрицы  
 public static double[][] ReverseGaussMainElementByCol(double[][] A) {  
 double[][] SoLAE = new double[*n*][2 \* *n*];  
 for (int i = 0; i < *n*; i++) {  
 for (int j = 0; j < *n*; j++) {  
 SoLAE[i][j] = A[i][j];  
 }  
 for (int j = *n*; j < 2 \* *n*; j++) {  
 SoLAE[i][j] = (i == j - *n* ? 1 : 0);  
 }  
 }  
 //Решение системы  
 for (int i = 0; i < *n*; i++) {  
 double mainElem = SoLAE[i][i];  
 for (int j = i + 1; j < *n*; j++) {  
 double koefRow = SoLAE[j][i] / mainElem;  
 for (int t = i; t < 2 \* *n*; t++) {  
 SoLAE[j][t] = SoLAE[j][t] - koefRow \* SoLAE[i][t];  
 }  
 }  
 }  
 for (int i = *n* - 1; i >= 0; i--) {  
 double mainElem = SoLAE[i][i];  
 for (int j = i - 1; j >= 0; j--) {  
 double koefRow = SoLAE[j][i] / mainElem;  
 for (int t = i; t < 2 \* *n*; t++) {  
 SoLAE[j][t] = SoLAE[j][t] - koefRow \* SoLAE[i][t];  
 }  
 }  
 }

//Получение решения  
 for (int i = 0; i < *n*; i++) {  
 double elem = SoLAE[i][i];  
 for (int j = 0; j < 2 \* *n*; j++) {  
 SoLAE[i][j] = SoLAE[i][j] / elem;  
 }  
 }  
 double[][] \_A = new double[*n*][*n*];  
 for (int i = 0; i < *n*; i++) {  
 for (int j = *n*; j < *n* \* 2; j++) {  
 \_A[i][j – *n*] = SoLAE[i][j];  
 }  
 }  
 return \_A;  
 }  
}

**Результаты**

****

****

****

**Вывод**

Таким образом, в результате поставленного эксперимента удалось выяснить, что использованный метод Гаусса с выбором главного элемента по столбцу позволяет минимизировать погрешность при решении систем линейных алгебраических уравнений, что доказывают полученные значение максимум-норм невязки и погрешности. Найден определитель случайной матрицы, который позволяет сделать вывод, что матрица имеет единственное решение при ненулевом определителе, который можно вычислить благодаря перемножению диагональных элементов полученной в результате прямого хода метода Гаусса треугольной матрицы. Найдена обратная матрица, правильность и точность вычисления которой доказывает результат перемножения обратной матрицы на исходную.