**МИНОБРНАУКИ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

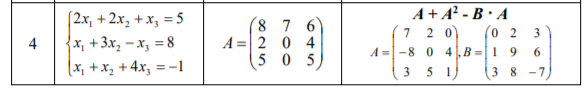
Лабораторная работа №1  
дисциплина: Вычислительная математика  
тема: «Метод Гаусса»

Выполнил: ст. группы ВТ-22  
Воскобойников Илья Проверил: Бондаренко Т.В.

Белгород 2019

**Цель работы:** изучить прямой и обратный ход метода Гаусса для решения систем линейных алгебраических уравнений; изучить схему единственного деления с выбором максимального по модулю элемента; изучить применение метода Гаусса для вычисления определителя матрицы и обратной матрицы; получить практические навыки программной реализации метода Гаусса и решения поставленных задач методом Гаусса с помощью ЭВМ.

**Вариант 4**



**Задания к работе**

1. Выполнить вручную действия над матрицами А и В из пункта 3 задания соответствующего варианта.

2. Выполнить следующие действия, не используя метод Гаусса:

− решить вручную систему линейных алгебраических уравнений с тремя неизвестными из пункта 1 задания соответствующего варианта;

− найти вручную определитель матрицы А из пункта 2 задания соответствующего варианта;

− найти вручную матрицу А -1 обратную матрице А из пункта 2 задания соответствующего варианта.

3. Решить вручную методом Гаусса систему линейных алгебраических уравнений с тремя неизвестными из пункта 1 задания соответствующего варианта.

4. Найти вручную с помощью метода Гаусса определитель матрицы А из пункта 2 задания соответствующего варианта.

5. Найти вручную с помощью метода Гаусса матрицу А -1 обратную матрице А из пункта 2 задания соответствующего варианта. Выполнить проверку полученной матрицы на соответствие условию: А·А -1 = Е, где Е ― единичная матрица.

6. Создать модуль для работы с матрицами произвольного порядка, содержащий подпрограммы для умножения двух матриц, умножения числа на матрицу, сложения матриц, вычитания матриц, транспонирования матрицы, умножения матрицы на вектор, ввода и вывода матрицы.

7. Создать модуль, содержащий подпрограммы, реализующие прямой и обратный ход метода Гаусса для схемы единственного деления с выбором максимального по модулю элемента.

8. Создать программу для решения следующих задач: − нахождение методом Гаусса решения системы линейных алгебраических уравнений с одной и той же матрицей коэффициентов при неизвестных и произвольным числом столбцах свободных членов.

Замечание. В прямом ходе метода Гаусса выполняется приведение расширенной матрицы (коэффициенты при неизвестных и свободные члены) к треугольному виду, и одновременно изменяются все столбцы свободных членов. На этапе обратного хода выполняется вычисление решения системы для каждого столбца свободных членов, составляется матрица решений. − вычисление определителя заданной матрицы методом Гаусса; − нахождение для заданной матрицы обратной матрицы методом Гаусса.

9. Решить все задания соответствующего варианта с помощью составленной программы. Выполнить проверку правильности найденного решения системы линейных уравнений и матрицы обратной к заданной матрице с помощью составленной программы, сравнить значения, полученные при решении заданий с помощью метода Гаусса и с использованием произвольного метода решения.

П6.

#include **<stdio.h>  
void** input(**float** \*\*A,**int** N, **int** M){  
 **for** (**int** i = 0; i <N ; ++i) {  
 **for** (**int** j = 0; j <M ; ++j) {  
 scanf(**"%f"**, &A[i][j]);  
 }  
 }  
}  
**void** output(**float** \*\*A,**int** N, **int** M){  
 **for** (**int** i = 0; i <N ; ++i) {  
 **for** (**int** j = 0; j <M ; ++j) {  
 printf(**"%f "**, A[i][j]);  
 }  
 printf(**"\n"**);  
 }  
}  
**void** mult\_A\_B(**float** \*\*A, **int** N, **int** M,**float** \*\*B, **float** \*\*C){  
 **float** temp ;  
 **for** (**int** i = 0; i < N; i++){  
 **for** (**int** j = 0; j < M; j++){  
 temp=0;  
 **for**(**int** k = 0; k < N; k++){  
 temp += A[i][k] \* B[k][j];  
 }  
 C[i][j] = temp;  
 }  
 }  
}  
**void** mult\_A\_X(**float** \*\*A, **int** N, **int** M, **float** X, **float** \*\*C){  
 **for** (**int** i = 0; i < N; ++i) {  
 **for** (**int** j = 0; j < M; ++j) {  
 C[i][j]=A[i][j]\*X;  
 }  
 }  
}  
**void** sum\_A\_B(**float** \*\*A, **int** N, **int** M, **float** \*\*B, **float** \*\*C){  
 **for** (**int** i = 0; i < N; ++i) {  
 **for** (**int** j = 0; j < M; ++j) {  
 C[i][j]=A[i][j]+B[i][j];  
 }  
 }  
}  
  
**void** dif\_A\_B(**float** \*\*A, **int** N, **int** M, **float** \*\*B, **float** \*\*C){  
 **for** (**int** i = 0; i < N; ++i) {  
 **for** (**int** j = 0; j < M; ++j) {  
 C[i][j]=A[i][j]-B[i][j];  
 }  
 }  
}  
**void** transpose(**float** \*\*A, **int** N, **int** M,**float** \*\*C){  
 **for** (**int** i = 0; i < N; ++i) {  
 **for** (**int** j = 0; j < M; ++j) {  
 C[i][j]=A[j][i];  
 }  
 }  
}  
**void** mult\_with\_vect(**float** \*\*A, **int** N, **int** M, **float** \*vect, **float** \*rez){  
 **for** (**int** i = 0; i < N; ++i) {  
 rez[i]=0;  
 **for** (**int** j = 0; j < M; ++j) {  
 rez[i]+=A[i][j]\*vect[j];  
 }  
 }  
}

П7

#include **<math.h>**#include **<stdio.h>***//находит макс элемент и отправляет на обмен строки в swap\_str***int** max\_el\_str(**float** \*\*A, **int** y, **int** i){  
 **float** maximym;  
 **int** max;  
 max = y;  
 maximym = abs(A[y][y]);  
 **for**(**int** n = y+1; n < i; n++)  
 {  
 **if**(maximym < abs(A[n][y]))  
 {  
 maximym = A[n][y];  
 max = n;  
 }  
 }  
 **return** max;  
}  
  
*//обмен строк***void** swap\_str(**float** \*\*A, **float** \*B, **int** y, **int** max, **int** i){  
 **float** a;  
 a = B[y];  
 B[y] = B[max];  
 B[max] = a;  
  
 **for**(**int** z = 0; z < i; z++){  
 a = A[y][z];  
 A[y][z] = A[max][z];  
 A[max][z] = a;  
 }  
}  
  
*//обратный ход***void** reverse\_move(**float** \*\*A, **float** \*B, **float** \*X, **int** n){  
 X[n - 1] = B[n - 1] / A[n - 1][n - 1];  
 printf(**"\nx%d = %f\n"**, n, X[n-1]);  
 **for** (**int** i = n - 2; i >= 0; i--){  
 X[i] = B[i];  
 **for** (**int** j = i + 1; j < n; j++) {  
 X[i] -= A[i][j] \* X[j];  
 }  
 X[i] /= A[i][i];  
 printf(**"x%d = %f\n"**, i+1, X[i]);  
 }  
}  
  
*//прямой ход***void** first\_move(**float** \*\*A, **int** i, **float** \*B){  
 **int** max;  
 **float** m;  
 **for**(**int** y = 0; y < i; y++){  
 max = max\_el\_str(A, y, i);  
 swap\_str(A, B, y, max, i);  
  
 **for**(**int** z = y+1; z < i; z++){  
 m = -(A[z][y]/A[y][y]);  
  
 **for**(**int** t = 0; t < i; t++) { *// работа со свободными членами* A[z][t] = A[z][t] + m \* A[y][t];  
 }  
 B[z] = B[z] + m \* B[y];  
 }  
 }  
}

П8

#include **<stdio.h>**#include **<stdlib.h>***//ввод матрицы размером i***float** \*\*read\_matr(**float** \*\*set, **int** i)  
{  
 **int** y;  
 **float** x;  
  
 printf(**"\nRead matr:\n"**);  
 set = (**float**\*\*) malloc(i \* **sizeof**(**float**\*));  
 **for**(y = 0; y < i; y++)  
 set[y] = (**float**\*) malloc(i \* **sizeof**(**float**));  
  
 **for**( y = 0; y < i; y++)  
 **for**(**int** t = 0; t < i; t++)  
 {  
 scanf(**"%f"**, &x);  
 set[y][t] = x;  
 }  
 **return** set;  
}  
  
*//вывод матрицы***void** write\_matr(**float** \*\*set, **int** i)  
{  
 printf(**"\nWrite matr:\n"**);  
 **for**(**int** y = 0; y != i; y++)  
 {  
 **for**(**int** t = 0; t != i; t++)  
 printf(**"%f "**, set[y][t]);  
 printf(**"\n"**);  
 }  
}

**float** \*\*read\_free\_term(**float** \*\*B, **int** i, **int** \*g)  
{  
 **int** h = 1;  
 **int** y;  
 **int** z = 0;  
 B = (**float**\*\*) malloc(i \* **sizeof**(**float**\*));  
 **for**(y = 0; y < i; y++)  
 B[y] = (**float**\*) malloc(1 \* **sizeof**(**float**));  
 **while**(h != 0)  
 {  
 **for**(y = 0; y < i; y++)  
 B[y] = (**float**\*) realloc(B[y], (z+1) \* **sizeof**(**float**));  
  
 printf(**"\nВведите свободные члены : "**);  
 **for**(y = 0; y < i; y++)  
 scanf(**"%f"**, &B[y][z]);  
 z++;  
  
 printf(**"Нужно ввести еще? 1 - да, 0 - нет "**);  
 scanf(**"%d"**, &h);  
 }  
  
  
 \*(g) = z;  
  
 **return** B;  
}  
  
**int** main()  
{  
 **int** y, i, g;  
 **float** r = 1;  
 **float** \*\*A = **NULL**;  
 **float** \*\*B = **NULL**;  
 **float** \*X = **NULL**;  
  
 printf(**"i = "**);  
 scanf(**"%d"**, &i);  
  
 A = read\_matr(A, i);  
  
 write\_matr(A, i);  
  
 B = read\_free\_term(B, i, &g);  
  
 first\_move(A, i, B, g);  
  
  
  
 X = (**float**\*)malloc(i \* **sizeof**(**float**));  
  
 **for**(y = 0; y < g; y++)  
 reverse\_move(A, B, X, i, y);  
  
  
  
 **for** (y = 0; y != i; y++)  
 free(A[y]);  
 free(A);  
  
 **for** (y = 0; y != g; y++)  
 free(B[y]);  
 free(B);  
  
 free(X);  
  
 **return** 0;  
}

