МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

**«Вятский государственный университет»**

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Методы решения систем уравнений

Отчет по лабораторной работе №2 дисциплины

«Вычислительная математика»

Выполнил студент группы ИВТб-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Седов М.Д./

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Исупов К.С./

Киров 2018

1 Постановка задачи

Решить систему линейных уравнений 4-го порядка методом Гаусса с точностью е=0,001.

Уравнения системы:

0,73\*x1+1,24\*x2-0,38\*x3-1,43\*x4=0,58

1,07\*x1-0,77\*x2+1,25\*x3+0,66\*x4=-0,66

1,56\*x1+0,66\*x2+1,44\*x3-0,87\*x4=1,24

0,75\*x1+1,22\*x2-0,83\*x3+0,87\*x4=0,92

Решить систему линейных уравнений 4-го порядка с точностью е=0,0001: - методом Зейделя.

Уравнения системы:

x1=0,17\*x1+0,31\*x2-0,18\*x3+0,22\*x4-1,71

x2=-0,21\*x1+0,33\*x3+0,22\*x4+0,62

x3=0,32\*x1-0,18\*x2+0,05\*x3-0,19\*x4-0,89

x4=0,12\*x1+0,28\*x2-0,14\*x3+0,94

Решить систему линейных уравнений 3-го порядка методом обратной матрицы с точностью e=0,001.

Уравнение системы:

3\*x1+2\*x2+7\*x3=0

-5\*x1+4\*x2+x3=2

x1-3\*x2+2\*x3=-7

Решить систему нелинейных уравнений 2-го порядка методом Ньютона с точностью е=0,001.

Уравнения системы:

sin(x)-y-1,32=0

cos(y)-x+0,85=0

Проверить результаты с помощью системы Mathcad.

2 Краткие теоретические сведения

2.1 Метод Гаусса для систем линейных уравнений

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) (СЛАУ). Назван в честь немецкого математика [Карла Фридриха Гаусса](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D1%83%D1%81%D1%81,_%D0%9A%D0%B0%D1%80%D0%BB_%D0%A4%D1%80%D0%B8%D0%B4%D1%80%D0%B8%D1%85). Это 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 (по номеру), находятся все переменные системы.

Пусть исходная система имеет следующий вид:

Ее можно записать в матричной форме: *Ax=b*, где

, , .

Матрица А называется основной матрицей системы, b – столбцом свободных членов.

Тогда, согласно свойству элементарных преобразований над строками, основную матрицу этой системы можно привести к ступенчатому виду (эти же преобразования нужно применять к столбцу свободных членов):

Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа:

* На первом этапе осуществляется так называемый прямой ход, когда путём [элементарных преобразований](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B) над строками систему приводят к ступенчатой или [треугольной форме](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0), либо устанавливают, что система несовместна. А именно, среди элементов первого столбца матрицы выбирают ненулевой, перемещают его на крайнее верхнее положение перестановкой строк и вычитают получившуюся после перестановки первую строку из остальных строк, домножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.
* На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить [фундаментальную систем у решений](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%B4%D0%B0%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9), либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.

Метод Гаусса требует O(n3) арифметических операций.

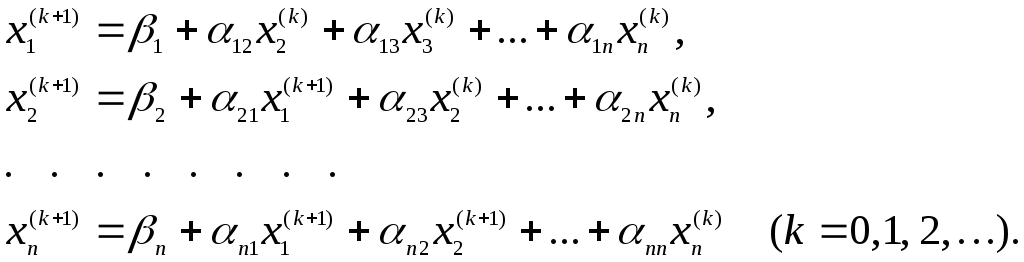
Достоинства метода Гаусса:

* Для матриц ограниченного размера менее трудоёмкий по сравнению с другими методами.
* Позволяет однозначно установить, совместна система или нет, и если совместна, найти ее решение.
* Позволяет найти максимальное число линейно независимых уравнений – ранг матрицы системы.

2.2 Метод Зейделя

Метод Зейделя представляет собой некоторую модификацию метода итераций. Основная идея его заключается в том, что при вычислении (k+1)-го приближения неизвестной xi учитываются уже вычисленные ранее (k+1)-е приближения неизвестных x1, x2, …, xi-1.

Пусть получена система. Выберем начальные приближения корней. Далее, предполагая, что k-ые приближения корней известны, согласно Зейделю будем строить (k+1)-е приближения корней по формулам:



Обычно метод Зейделя даёт лучшую сходимость, чем метод простой итерации, но приводит к более громоздким вычислениям.

Условия сходимости для метода Зейделя:

Чтобы последовательность приближений сходилось, необходимо и достаточно, чтобы любая из норм матрицы α была меньше 1.

||α||1 = , ||α||2 = , ||α||3 =

Оценка погрешности метода Зейделя:

||xт – x(k)|| <=

2.3 Метод обратной матрицы

Метод обратной матрицы используется при решении систем линейных алгебраических уравнений, если число неизвестных равно числу уравнений.

Суть метода обратной матрицы:

Пусть задана система *n* линейных уравнений с *n* неизвестными:

Эту систему можно записать в виде матричного уравнения AX=B, где

*,*

*,*

*.*

Из полученного матричного уравнения необходимо выразить X. Для этого умножим обе части матричного уравнения слева на A-1, получим:

A-1 \*A\*X = A-1 \*B

Так как A-1 \* A = E, то E \* X = A-1 \* B или X = A-1 \* B. Далее находится обратная матрица A-1 и умножается на столбец свободных членов B.

Обратная матрица к матрице А существует только при условии, что detA ≠ 0. Поэтому при решении системы линейных уравнений методом обратной матрицы в первую очередь вычисляется detA. Если detA ≠ 0, то система имеет единственное решение, которое можно получить методом обратной матрицы, если же detA = 0, то методом обратной матрицы решить систему нельзя.

2.4 Метод Ньютона

Метод Ньютона решения систем нелинейных уравнений является обобщением метода Ньютона решения нелинейных уравнений, который основан на идее линеаризации.

Пусть дана система из ***n*** нелинейных уравнений c ***n*** неизвестными.

, где – нелинейные функции, определенные и непрерывно дифференцируемые в некоторой области G.

Запишем её в векторном виде:

Требуется найти такой вектор , который, при подстановке в исходную систему, превращает каждое уравнение в верное числовое равенство.

При таком подходе формула для нахождения решения является естественным обобщением формулы одномерного итеративного процесса:

, k = 0, 1, 2…, где

В качестве критерия окончания процесса итераций обычно берут условие ||x(k+1) – x(k)|| < ε, где ε – требуемая точность решения.

Расчет погрешности: S = ||x(k) – x(k-1)||.

3 Выполнение задания

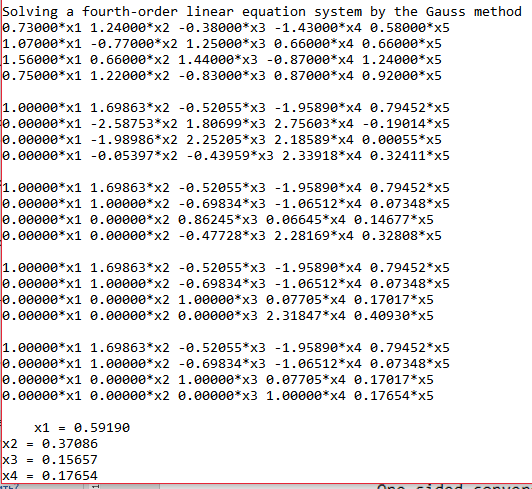


Рисунок 1 – метод Гаусса

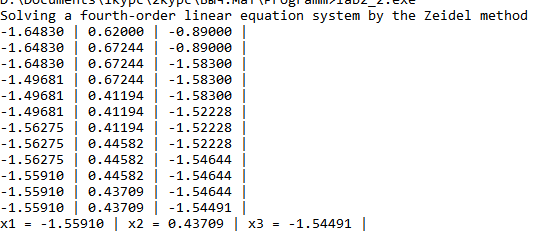


Рисунок 2 –метод Зейделя

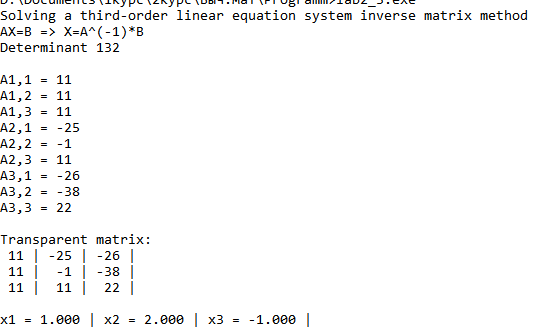


Рисунок 3 –метод обратной матрицы

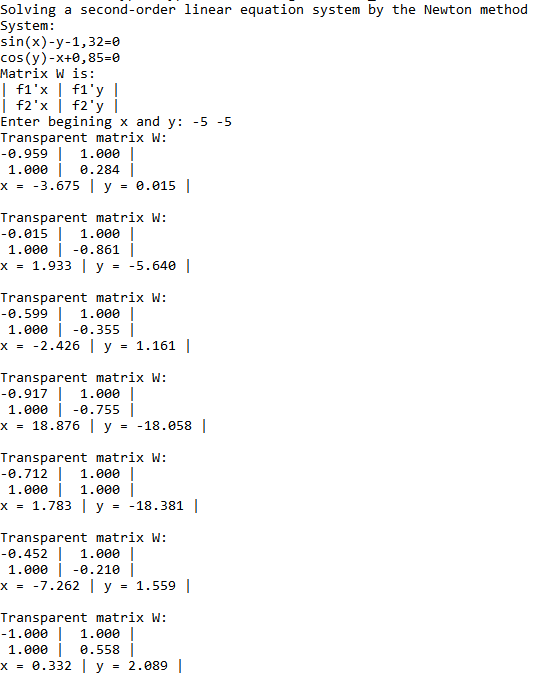


Рисунок 4 –метод Ньютона

Приложение А

Листинг программы #1

#include <iostream>

#include <iomanip>

using namespace std;

double matrix [4][5] = {{0.73, 1.24, -0.38,-1.43, 0.58},

{1.07, -0.77, 1.25, 0.66, 0.66},

{1.56, 0.66, 1.44, -0.87, 1.24},

{0.75, 1.22, -0.83, 0.87, 0.92} };

const int n = 4, m = 5;

void show\_matrix() {

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) cout << fixed << setprecision(5) << matrix[i][j] << "\*x" << j + 1 << " " << setw(5);

cout << endl;

}

cout << endl;

}

int main() {

cout << "Solving a fourth-order linear equation system by the Gauss method" << endl;

double e = 0.001;

show\_matrix();

double tmp, \*xx = new double[m];

for (int i = 0; i < n; i++) {

tmp = matrix[i][i];

for (int j = n; j >= i; j--) matrix[i][j] /= tmp;

for (int j = i + 1; j < n; j++) {

tmp = matrix[j][i];

for (int k = n; k >= i; k--) matrix[j][k] -= tmp \* matrix[i][k];

}

show\_matrix();

}

xx[n - 1] = matrix[n - 1][n];

for (int i = n - 2; i >= 0; i--) {

xx[i] = matrix[i][n];

for (int j = i + 1; j < n; j++) xx[i] -= matrix[i][j] \* xx[j];

}

for (int i = 0; i < n; i++) cout << "x" << i + 1 << " = " << xx[i] << endl;

delete [] xx;

}

Листинг программы #2

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

const int n = 3;

const double e = 0.01;

bool converge(double \*xk, double \*xkp) {

double norm = 0;

for (int i = 0; i < n; i++) norm += (xk[i] - xkp[i]) \* (xk[i] - xkp[i]);

return (sqrt(norm) < e);

}

int main() {

cout << "Solving a fourth-order linear equation system by the Zeidel method" << endl;

double A [4][4] = { {0.17, 0.31, -0.18, 0.22},

{-0.21, 0, 0.33, 0.22},

{0.32, -0.18, 0.05, -0.19},

{0.12, 0.28, -0.14, 0}

};

double B[4] = {-1.71, 0.62, -0.89, 0.94};

double x[4], p[4];

for (int i = 0; i < n; i++) x[i] = B[i];

do {

for (int i = 0; i < n; i++) p[i] = x[i];

for (int i = 0; i < n; i++) {

double var = 0;

for (int j = 0; j < n; j++) var += (A[i][j] \* x[j]);

x[i] = B[i] + var;

for (int i = 0; i < n; i++) cout << setw(5) << fixed << setprecision(5) << x[i] << " | ";

cout << endl;

}

}

while (!converge(x, p));

for (int i = 0; i < n; i++) cout << "x" << i + 1 << " = " << x[i] << " | ";

}

Листинг программы #3

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int det\_matrix(int (&a)[3][3]) {

return a[0][0] \* (a[1][1] \* a[2][2] - a[2][1] \* a[1][2]) - a[0][1] \* (a[1][0] \* a[2][2] - a[2][0] \* a[1][2]) + a[0][2] \* (a[1][0] \* a[2][1] - a[2][0] \* a[1][1]);

}

int \_A(int i, int j, int (&a)[3][3]) {

int a2[2][2];

int count = 0;

for (int k = 0; k < 3; k++)

for (int l = 0; l < 3; l++)

if (k != i && l != j) {

a2[count / 2][count % 2] = a[k][l];

count++;}

return pow(-1, i + j) \* (a2[0][0] \* a2[1][1] - a2[0][1] \* a2[1][0]);

}

int main() {

cout << "Solving a third-order linear equation system inverse matrix method" << endl << "AX=B => X=A^(-1)\*B" << endl;

int n = 3;

int A[3][3] = { {3, 2, 7},

{-5, 4, 1},

{1, -3, 2}

};

int B[3] = {0, 2, -7};

double X[3];

int det;

int A\_T[3][3];

if (det\_matrix(A) != 0) {

det = det\_matrix(A);

cout << "Determinant " << det << endl << endl;

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++) {

A\_T[j][i] = \_A(i, j, A);

cout << "A" << i + 1 << "," << j + 1 << " = " << A\_T[j][i] << endl;

}

cout << endl << "Transparent matrix: " << endl;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++)

cout << setw(3) << A\_T[i][j] << " | ";

cout << endl;

}

for (int i = 0; i < 3; i++) {

int val = 0;

for (int j = 0; j < 3; j++)

val += A\_T[i][j] \* B[j];

X[i] = 1 / double(det) \* val;

}

cout << endl;

for (int i = 0; i < 3; i++) cout << "x" << i + 1 << " = " << fixed << setprecision(3) << X[i] << " | ";

} else cout << "The system has no solutions.";

}

Листинг программы #4

#include <iostream>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

double f1(double x, double y) {

return sin(x) - y - 1.32;}

double f2(double x, double y) {

return cos(y) - x + 0.85;}

double f1\_x(double x, double y) {

return cos(x);}

double f1\_y(double x, double y) {

return -1;}

double f2\_x(double x, double y) {

return -1;}

double f2\_y(double x, double y) {

return -sin(y);}

int main() {

cout << "Solving a second-order linear equation system by the Newton method" << endl;

double e = 0.001;

cout << "System: " << endl

<< "sin(x)-y-1,32=0" << endl

<< "cos(y)-x+0,85=0" << endl;

cout << "Matrix W is: " << endl

<< "| f1'x | f1'y |" << endl

<< "| f2'x | f2'y |" << endl;

double x, y, W[2][2], dx, dy, norm;

cout << "Enter begining x and y: ";

cin >> x >> y;

do {

W[0][0] = f2\_y(x, y);

W[0][1] = -f1\_y(x, y);

W[1][0] = -f2\_x(x, y);

W[1][1] = f1\_x(x, y);

cout << "Transparent matrix W: " << endl;

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++)

cout << setw(6) << fixed << setprecision(3) << W[i][j] << " | ";

cout << endl;

}

double det = W[0][0] \* W[1][1] - W[0][1] \* W[1][0];

dx = (W[0][0] \* f1(x, y) + W[0][1] \* f2(x, y)) / det;

dy = (W[1][0] \* f1(x, y) + W[1][1] \* f2(x, y)) / det;

double px = x, py = y;

x -= dx;

y -= dy;

cout << "x = " << x << " | y = " << y << " |" << "\n\n";

norm = max((px - x) \* (px - x), (py - y) \* (py - y));

} while (norm >= e);

cout << "x = " << x << " | y = " << y << " |";

}