|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| МИНОБРНАУКИ РОССИИ | | | |
| 1. Федеральное государственное бюджетное образовательное учреждение 2. высшего профессионального образования 3. **"Московский технологический университет"** 4. МИРЭА | | | |
| Институт Кибернетики | | |  |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры при АО «Концерн радиостроения «ВЕГА» | | |  |

|  |  |
| --- | --- |
| **Отчет по лабораторной работе №2** | |
| **по дисциплине** | |
| **«** Численные методы **»** | |
| **Вариант 7** | |
| Студент 3-го курса  группы КМБО-2-16 | Миронов Д.А. |
| Преподаватель | Даева С.Г. |
| Рецензент |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2018 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2018 г. |  |

Москва 2018

Содержание

[Задание №1 3](#__RefHeading___Toc7811_1268055582)

[Теоретическая часть 3](#__RefHeading___Toc7813_1268055582)

[Практическая часть](#__RefHeading___Toc7815_1268055582) 3

[Задание №2](#__RefHeading___Toc7817_1268055582) 3

[Теоретическая часть](#__RefHeading___Toc696_1272466978) 3

[Практическая часть](#__RefHeading___Toc698_1272466978) 4

[Задание №3](#__RefHeading___Toc2885_1914997794) 4

[Практическая часть](#__RefHeading___Toc2889_1914997794) 4

[Приложения](#__RefHeading___Toc2891_1914997794) 6

**Задание № 1.**

Разработать программу решения системы линейных уравнений (СЛУ) методом Гаусса.

**Теоретическая часть.**

Метод Гаусса включает в себя 2 шага:

**1. Прямой ход.**

На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, 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.

**2. Обратный ход.**

На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, 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.

**Практическая часть.**

Была написана программа «lab2.1» на языке «C++» в операционной системе «ubuntu» с использованием компилятора «GNU GCC».

**Задание № 2.**

Разработать программу решения СЛУ методом Зейделя.

**Теоретическая часть.**

Итерационный метод решения систем линейных уравнений, основанный на преобразовании системы **Ax = b** к виду **x = β + αx**, где **β** – новый столбец свободных членов, а **α** – новая матрица коэффициентов. После приведения системы к такому виду можно организовать итерационный процесс **x(k+1) = β + αx(k)**. Такой метод называется *методом итераций*.

Особенность метода Зейделя заключается в использовании при вычислении **xi** приближенных значений **x1**, **x2**, …, **xi-1**, полученных на этой же итерации.

x = β + αx = φ(x)

xi(k) = φ(x1(k-1), x2(k-1), …, xn(k-1)) – метод итераций.

xi(k) = φ(x1(k), x2(k), …, xi-1(k), xi+1(k-1), …, xn(k-1)) – метод Зейделя.

xi(k+1) = βi + ∑(j = 1, …, i-1)αijxj(k+1) + ∑(j= i, …, n)αijxj(k)

Сходимость итерационных методов зависит от матрицы **α** коэффициентов и определяется следующей теоремой: ***x(n)*** *имеет предел и сходится к корню уравнения, если какая-нибудь каноническая форма матрицы α меньше 1*.

Под *нормой* матрицы **A** понимается такое действительное число **||A||**, что:

1. ||A|| ≥ 0,
2. ||αA|| = |α| ∙ ||A||,
3. ||A + B|| ≤ ||A|| + ||B||,
4. ||AB|| = ||A|| ∙ ||B||.

Если при этом:

1. |aij| ≤ ||A||,
2. |A| ≤ |B| => ||A|| ≤ ||B||,

то такая норма называется *канонической*.

Примеры канонических норм: ||A||m = maxi∑j|aij|, ||A||i = maxj∑i|aij|, ||A||k = √∑i, j|aij|2

**Практическая часть.**

Была написана программа «lab2.2» на языке «C++» в операционной системе «ubuntu» с использованием компилятора «GNU GCC».

Решение СЛУ начинается с приведения системы к виду x = β + αx. Для этого на диагонали матрицы α ставятся 0, а все остальные элементы вычисляются как отношение соответствующего элемента матрицы A к диагональному элементу из той же строки. Столбец свободных членов β вычисляется аналогично делением на диагональный элемент из соответствующей строки.

После получения α и β, начинается итерационный процесс вычисления вектора X по формулам, описанным в теоретической части. Используется всего один массив xi, так как в вычислениях используются ранее полученные на этой итерации значения.

После каждой итерации производится подстановка полученного X в систему и оценивается точность результата. Как только для каждого уравнения системы точность корня становится не больше ε = 0.001, итерационный процесс прекращается.

**Задание № 3.**

3.1. Исследовать сходимость итерационных процессов для конкретной СЛУ и решить ее точным и приближенным методом, разработанным в заданиях 1 и 2.

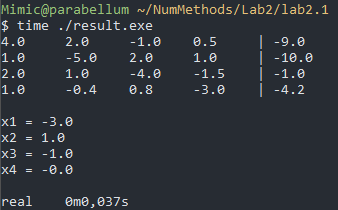
3.2. Сопоставить время расчетов и точность результатов.

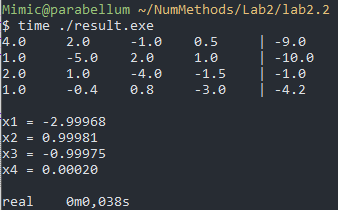
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 2 | -1 | 0.5 | -9 |
| 1 | -5 | 2 | 1 | -10 |
| 2 | 1 | -4 | -1.5 | -1 |
| 1 | -0.4 | 0.8 | -3 | -4.2 |

**Практическая часть**

Время измерялось с помощью утилиты time.

Решая систему методом Гаусса, программа нашла ответ – вектор .

Время выполнения программы: 0.037s

Решая систему методом Зейделя, программа выдала ответ – вектор Х = (-2.99968, 0.99981, -0.99975, 0.00020). Время выполнения: 0.038

**Приложения.**

**Исходный код**

*Код программы «lab2.1»*

#include <iostream>

#include <math.h>

#include <stdlib.h>

using namespace std;

int m = 5;

int i, j, n;

float matrix[4][5] = {

{ 4, 2, -1, 0.5, -9},

{ 1, -5, 2, 1, -10},

{ 2, 1, -4, -1.5, -1},

{ 1, -0.4, 0.8, -3, -4.2}

};

void cat\_mat()

{

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

{

for (j=0; j<m; j++)

printf("%s%.1f\t", (j==4)?"| ":"", matrix[i][j]);

cout << endl;

}

cout << endl;

}

int main()

{

n = m-1;

cat\_mat();

// forward move

float tmp, res[m];

int k;

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

{

tmp=matrix[i][i];

for (j=n;j>=i;j--)

matrix[i][j]/=tmp;

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

{

tmp=matrix[j][i];

for (k=n;k>=i;k--)

matrix[j][k]-=tmp\*matrix[i][k];

}

}

// reverse move

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

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

{

res[i] = matrix[i][n];

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

res[i]-=matrix[i][j]\*res[j];

}

// result out

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

printf("x%d = %.1f\n",i+1, res[i]);

return 0;

}

*Код программы «lab2.2»*

#include <iostream>

using namespace std;

int N = 4;

double A[4][5] = {

{ 4, 2, -1, 0.5, -9},

{ 1, -5, 2, 1, -10},

{ 2, 1, -4, -1.5, -1},

{ 1, -0.4, 0.8, -3, -4.2}

};

double B[4] = {-9, -10, -1, -4.2};

double res[4];

double eps = 10;

void cat\_mat()

{

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

{

for (int j=0; j<N+1; j++)

printf("%s%.1f\t", (j==4)?"| ":"", A[i][j]);

cout << endl;

}

cout << endl;

}

void zeidel()

{

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

{

res[i] = B[i];

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

{

res[i] += A[i][j] \* res[j];

}

}

return;

}

void change\_matr()

{

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

{

double del = A[i][i];

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

{

if (i == j)

{

A[i][j] = 0;

}

else

{

A[i][j] /= del;

A[i][j] \*= -1;

}

}

B[i] /= del;

}

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

{

res[i] = B[i];

}

return;

}

void check()

{

double chk;

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

{

chk = B[i];

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

{

chk += A[i][j] \* res[j];

}

chk -= res[i];

if (chk < 0) chk \*= -1;

if (i == 0)

{

eps = chk;

}

else

{

if (chk > eps) eps = chk;

}

}

return;

}

void output()

{

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

{

printf("x%d = %.5f\n", i+1, res[i]);

}

}

int main()

{

cat\_mat();

change\_matr();

while (eps >= 0.001)

{

zeidel();

check();

}

output();

return 0;

}