Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

Институт компьютерных наук и кибербезопастности

Высшая школа программной инженерии

**Лабораторная работа №2**

**по дисциплине «Вычислительная математика»**

Выполнил

Студент группы 5130904/20004 Машкин А.А.

Преподаватель Устинов С.М.

Оглавление

[Задание 2](#_Toc163212483)

[Результаты 3](#_Toc163212484)

[Вывод 4](#_Toc163212485)

[Код программы 5](#_Toc163212486)

[<DIR>/computational\_mathematics/lab\_2/main.cpp 5](#_Toc163212487)

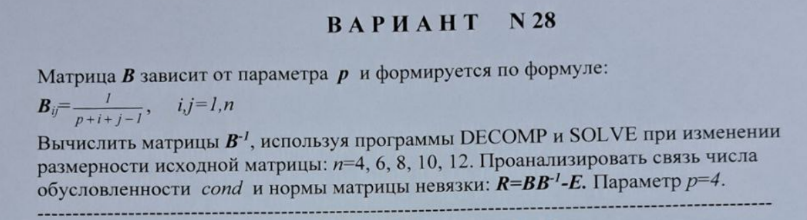
[<DIR>/computational\_mathematics/lab\_2/solve.h 6](#_Toc163212488)

[<DIR>/computational\_mathematics/lab\_2/solve.cpp 6](#_Toc163212489)

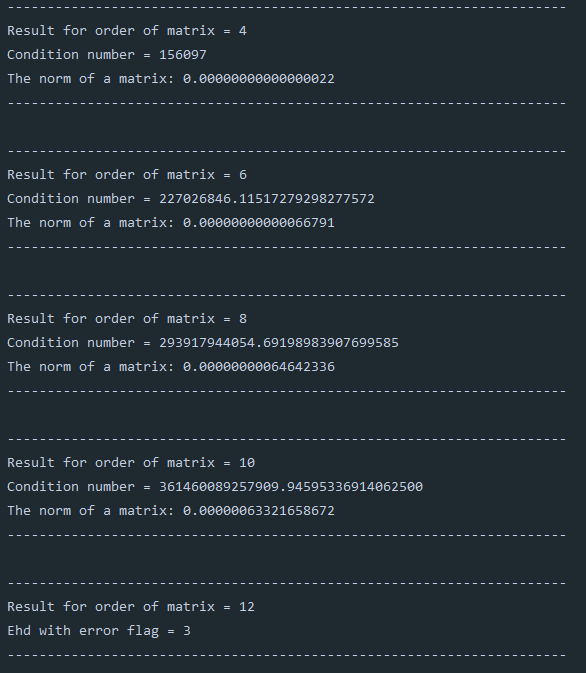
[<DIR>/computational\_mathematics/lab\_2/matrix\_function.h 7](#_Toc163212490)

[<DIR>/computational\_mathematics/lab\_2/matrix\_function.cpp 7](#_Toc163212491)

# Задание



# Результаты



# Вывод

В лабораторной работе видно, что при росте числа обусловленности растет и норма матрицы невязки. Но на матрице порядка 12 программа выдает ошибку с флагом равным 3, это означает, что определитель в этом случае у нас равен нулю.

# Код программы

## <DIR>/computational\_mathematics/lab\_2/main.cpp

#include <iostream>

#include <iomanip>

#include <algorithm>

#include "decomp.h"

#include "solve.h"

#include "matrix\_function.h"

int main() {

const auto default\_precision{std::cout.precision()};

static long double cond = 0;

double det = 0;

const int ndim = 12;

int n = 2;

int p = 4;

int pivot[ndim], i, flag;

long double B[ndim \* ndim] = {0};

long double E[ndim \* ndim] = {0};

long double result\_for\_multiplication[ndim \* ndim] = {0};

long double result\_for\_subtraction[ndim \* ndim] = {0};

long double copy\_of\_b[ndim \* ndim] = {0};

long double copy\_of\_e[ndim \* ndim] = {0};

for (int k = 0; k < 5; k++)

{

n += 2;

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

{

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

{

B[i \* ndim + j] = 1.0 / (p + i + j - 1);

if (i == j)

{

E[i \* ndim + j] = 1;

}

else

{

E[i \* ndim + j] = 0;

}

}

}

if (k == 0)

{

std::copy(B, B + ndim \* ndim, copy\_of\_b);

std::copy(E, E + ndim \* ndim, copy\_of\_e);

}

decomp(n, ndim, B, &cond, pivot, &flag);

std::cout << "----------------------------------------------------------------------\n";

std::cout << "Result for order of matrix = " << n << "\n";

if (flag == 0)

{

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

{

solve(n, ndim, B, (E + i \* ndim), pivot);

}

std::cout << "Condition number = " << cond << "\n" << std::fixed << std::setprecision(17);// << std::scientific << std::setprecision(1);

// std::cout << "R = [\n";

mashkin::multiply(n, ndim, copy\_of\_b, E, result\_for\_multiplication);

mashkin::subtract(n, ndim, result\_for\_multiplication, copy\_of\_e, result\_for\_subtraction);

// mashkin::print\_matrix(n, ndim, result\_for\_subtraction);

// std::cout << "]\n";

std::cout << "The norm of a matrix: ";

std::cout << mashkin::calculate\_the\_norm\_of\_a\_matrix(result\_for\_subtraction, ndim);

std::cout << "\n";

det = pivot[n-1];

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

det = det \* B[i \* ndim + i];

}

else

{

std::cout << "Ehd with error flag = " << flag << "\n";

}

std::cout << "----------------------------------------------------------------------\n\n";

}

return 0;

}

## <DIR>/computational\_mathematics/lab\_2/solve.h

#ifndef LAB2\_SOLVE\_H

#define LAB2\_SOLVE\_H

int solve (int n, int ndim, long double \*a, long double b[], int pivot[]);

#endif

## <DIR>/computational\_mathematics/lab\_2/solve.cpp

#include "solve.h"

#define AINDEX(i,j) (i \* ndim + j)

int solve (int n, int ndim, long double \*a, long double b[], int pivot[])

{ /\* --- begin function solve() --- \*/

int i, j, k, m;

long double t;

if (n == 1)

{

/\* trivial \*/

b[0] /= a[0];

}

else

{

/\* Forward elimination: apply multipliers. \*/

for (k = 0; k < n-1; k ++)

{

m = pivot[k];

t = b[m]; b[m] = b[k]; b[k] = t;

for (i = k+1; i < n; ++i) b[i] += a[AINDEX(i,k)] \* t;

}

/\* Back substitution. \*/

for (k = n-1; k >= 0; --k)

{

t = b[k];

for (j = k+1; j < n; ++j) t -= a[AINDEX(k,j)] \* b[j];

b[k] = t / a[AINDEX(k,k)];

}

}

return(0);

} /\* --- end function solve() --- \*/

## <DIR>/computational\_mathematics/lab\_2/matrix\_function.h

#ifndef LAB2\_MATRIX\_FUNCTION\_H

#define LAB2\_MATRIX\_FUNCTION\_H

namespace mashkin

{

void multiply(int n, int ndim, long double \*a, long double \*b, long double \*result);

void subtract(int n, int ndim, long double \*a, long double \*b, long double \*result);

double calculate\_the\_norm\_of\_a\_matrix(long double \*matrix, int ndim);

}

#endif

## <DIR>/computational\_mathematics/lab\_2/matrix\_function.cpp

#include "matrix\_function.h"

#include <iomanip>

#include <iostream>

namespace mashkin {

void multiply(int n, int ndim, long double \*a, long double \*b, long double \*result) {

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

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

long double temp = 0.0;

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

temp += a[i \* ndim + k] \* b[k \* ndim + j];

}

result[i \* ndim + j] = temp;

}

}

}

void subtract(int n, int ndim, long double \*a, long double \*b, long double \*result)

{

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

{

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

{

result[i \* ndim + j] = a[i \* ndim + j] - b[i \* ndim + j];

}

}

}

double calculate\_the\_norm\_of\_a\_matrix(long double \*matrix, int ndim)

{

double result = 0.0;

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

{

double var = 0.0;

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

{

var += matrix[i \* ndim + j];

}

if (var > result)

{

result = var;

}

}

return result;

}

}