|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт кибернетики | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

|  |  |
| --- | --- |
|  | |
| **ЛАБОРАТОРНАЯ РАБОТА № 2** | |
| Вариант № 12 | |
|  | |
| Студент группы: КМБО-02-19   Курса: 3 | *С. А. Минеев* |
| Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_  (Должность) | *А. А. Липатов* |

|  |  |  |
| --- | --- | --- |
| «Отчет представлен к рассмотрению» | «\_\_\_» \_\_\_\_\_\_\_\_\_\_2021 г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *(подпись студента)* |
|  |  |  |
| «Отчет утвержден.  Допущен к защите» | «\_\_\_» \_\_\_\_\_\_\_\_\_\_2021 г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *(подпись руководителя)* |

Москва 2021

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc90141157)

[1 Текст задания 3](#_Toc90141158)

[1.1 Задание № 1 1](#_Toc90141159)

[1.2 Задание № 2 1](#_Toc90141160)

[1.3 Задание № 3 1](#_Toc90141161)

[2 Теоретическая часть 1](#_Toc90141162)

[2.1 Задание № 1 1](#_Toc90141163)

[2.2 Задание № 2 1](#_Toc90141164)

[2.3 Задание № 3 1](#_Toc90141165)

[3 Практическая часть 1](#_Toc90141166)

[3.1 Задание № 1 1](#_Toc90141167)

[3.2 Задание № 2 1](#_Toc90141168)

[3.3 Задание № 3 1](#_Toc90141169)

[4 Приложения № 1 – Примеры работы программ. 1](#_Toc90141170)

[4.1 Задание № 1 1](#_Toc90141171)

[4.2 Задание № 2 1](#_Toc90141172)

[4.3 Задание № 3 1](#_Toc90141173)

[5 Приложения № 2 – Листинг программ. 1](#_Toc90141174)

[4.1 Задание № 1 1](#_Toc90141175)

[4.2 Задание № 2 1](#_Toc90141176)

[4.3 Задание № 3 1](#_Toc90141177)

[Список использованных источников 1](#_Toc90141178)

[Как раскрыть все заголовки в документе: 1](#_Toc90141179)

# Текст задания

* 1. Задание № 1

1. Разработать программу решения системы линейных уравнений (СЛАУ)

точным методом Холецкого.

* 1. Задание № 2

1) Разработать программу решения СЛУ приближенным методом простой итерации.

* 1. Задание № 3

1. Исследовать сходимость итерационных процессов для конкретной

СЛАУ и решить ее точным и приближенным методом, разработанным в заданиях

1 и 2.

1. Сопоставить время расчетов и точность результатов.

А = B =

# Теоретическая часть

* 1. Задание № 1

### 2.1.1] Пункт 1 (QProject\_2)

#### Тема: Метод Холецкого (метод квадратного корня).

Разложение Холецкого впервые предложено французским офицером и математиком польского происхождения Андре-Луи Холецким в конце Первой Мировой войны, незадолго до его гибели в бою в августе 1918 г. Идея этого разложения была опубликована в 1924 г. его сослуживцем.

(1.1)

*AX = B; (1.2)*

*Метод Холецкого* (известный также как *метод квадратного корня* )

предназначен для решения систем уравнений вида (1.2) с действительной

симметричной положительно определенной матрицей.

Напомним, что матрица называется *положительно определенной ес*ли для любого вектора выполнено неравенство:

*(1.3)*

Матрицы с такими свойствами возникают, например, при использовании метода наименьших квадратов и численном решении дифференциальных уравнений.

Метод Холецкого основан на разложении матрицы *A* в произведение:

(1.4)

где *L* – нижняя треугольная матрица с положительными элементами на главной диагонали. Известно, что разложение (1.4) можно получить для любой матрицы, удовлетворяющей указанным свойствам.

Существует также обобщение этого разложения на случай комплекснозначных матриц.

Если разложение (1.4) получено, то решение системы (1.2) сводится к последовательному решению двух систем уравнений с треугольными матрицами:

(1.5)

Следует отметить, что на практике часто необходимо решить последовательность систем вида (1.2), отличающихся лишь правой частью *B*. Учитывая этот факт и соотношения (1.5), однократная факторизация матрицы *A* значительно упрощает решение оставшихся систем.

Получим расчетные формулы метода. Из условия (1.4) следует, что:

Так как матрица *A*  симметричная, можно рассмотреть лишь случай .

Т. к. , данные условия можно записать в виде:

В частности, при *i = j*, получаем:

*Откуда:*

*Далее, при i < j получим:*

Итак, если предположить, что метод квадратного корня применяется к системе уравнений с действительной симметричной положительно определенной матрицей, то элементы матрицы *L* можно вычислить, начиная с ее левого угла, по следующим расчетным формулам:

1. ;

Подсчитаем число операций, требующихся для выполнения разложения.

Вычисления по формуле (1.7) требуют:

операций. Вычисления по формуле (1.8) при каждом фиксированном *j*

требуют:

операций, а всего потребуется:

операций. Следует отметить, что в дополнение к указанным действиям для расчета по формулам (1.7), (1.8) потребуется *n* операций извлечения корня.

Если матрица *A* факторизована в виде (1.4) , то обратный ход метода квадратного корня состоит в последовательном решении двух систем уравнений:

(1.5)

Решения этих систем находятся по формулам, аналогичным формулам об-

ратного хода метода Гаусса:

Вычисления по формулам (1.9), (1.10) потребуют 2*n*(*n*+1) операций, что

не оказывает существенного влияния на кубическую трудоемкость метода.

Таким образом, общее время работы метода можно оценить как:

где *τ*  время выполнения одной операции.

При больших *n* время работы метода Холецкого будет примерно в два раза меньше, чем метода Гаусса, это сокращение объясняется тем, что *A*  симметричная матрица, и метод Холецкого учитывает данную особенность задачи.

* 1. Задание № 2

### 2.2.1] Пункт 1 (QProject\_3)

#### Тема: Приближенный Метод простой итераций.

Пусть дана система линейных уравнений, имеющая вид:

(2.1)

Который можно привести к виду:

*AX = B; (2.2)*

Если система имеет большую размерность ( – уравнений) или матрица системы ***разрежена***, более эффективны для решения непрямые итерационные методы. Разреженная матрица возникают в системе, где многие коэффициенты при неизвестных равны нулю.

Итерационные методы (методы последовательных приближений) состоят в том, что решение системы (2.1) находится как предел последовательных приближений решение системы (1) находится как предел последовательных приближений при *n*   , где *n* номер итерации. При использовании методов итерации обычно задается некоторое малое число *ε*  0 и вычисления проводятся до тех пор, пока не будет выполнена оценка:

(2.3)

Одним из таких методов является приближенный метод простых итераций.

В методе простых итераций первым делом систему (2.2) стандартного вида переводят к виду:

*;*

*При этом должно выполняться сходимости итерационного процесса. Итерационный процесс сходится если:*

*Для нахождения нормы можно использовать любой из следующих трех способов поиска нормы:*

1. *m* – норма или *j* – норма:
2. *i* – норма:
3. *k* – норма:

*Если хотя бы для одной из нормы выполняется условие (2.4) , то итерационный процесс сходится.*

Добиться того, чтобы выполнялось условие (2.4) Можно путем *элементарный преобразований* над матрицей А таким образом, чтобы:

(2.5)

А для получения матрицы после элементарных преобразований матрицы А и приведения к виду в котором выполняется условие (2.4) выражают систему через элементы, стоящие на главной диагонали и каждую строку системы делят на соответствующий коэффициент, стоящий на главной диагонали.

Стоит заметить в полученной системе коэффициенты на главной диагонали будут занулены, а сумма элементов по строке и/или по столбцу будет меньше 1, поэтому будет выполняться условие (2.4).(В результатах кода такая матрица обозначается R = и *r =*  для аналогичных преобразований матрицы B).

После того как получена система:

Составляют итерационный процесс:

В качестве берут r = и проводят необходимое количество итераций, пока не достигнем выполнения условия:

(2.3)

Корнем будет являться значение .

* 1. Задание № 3

### 2.3.1] Пункт 1

#### Тема: Сходимость итерационных процессов.

1. Необходимое условие сходимости.

Для того, чтобы итерационный процесс сходился необходимо чтобы выполнялось условие:

*=>*

и для одной из строк условие имело строгое неравенство:

1. Достаточное условие сходимости итерационных процессов.

Для того чтобы итерационный процесс сходился достаточно, чтобы выполнялось условие:

# Практическая часть

* 1. Задание № 1

### 3.1.1] Пункт 1

---

* 1. Задание № 2

### 3.2.1] Пункт 1

---

* 1. Задание № 3

### 3.3.1] Пункт 1

#### Дано:

А = B =

#### Задание:

• Исследовать сходимость итерационных процессов для конкретной

СЛАУ и решить ее точным и приближенным методом, разработанным в заданиях.

#### Решение:

##### Исследование сходимости итерационных процессов данной СЛАУ:

СЛАУ:

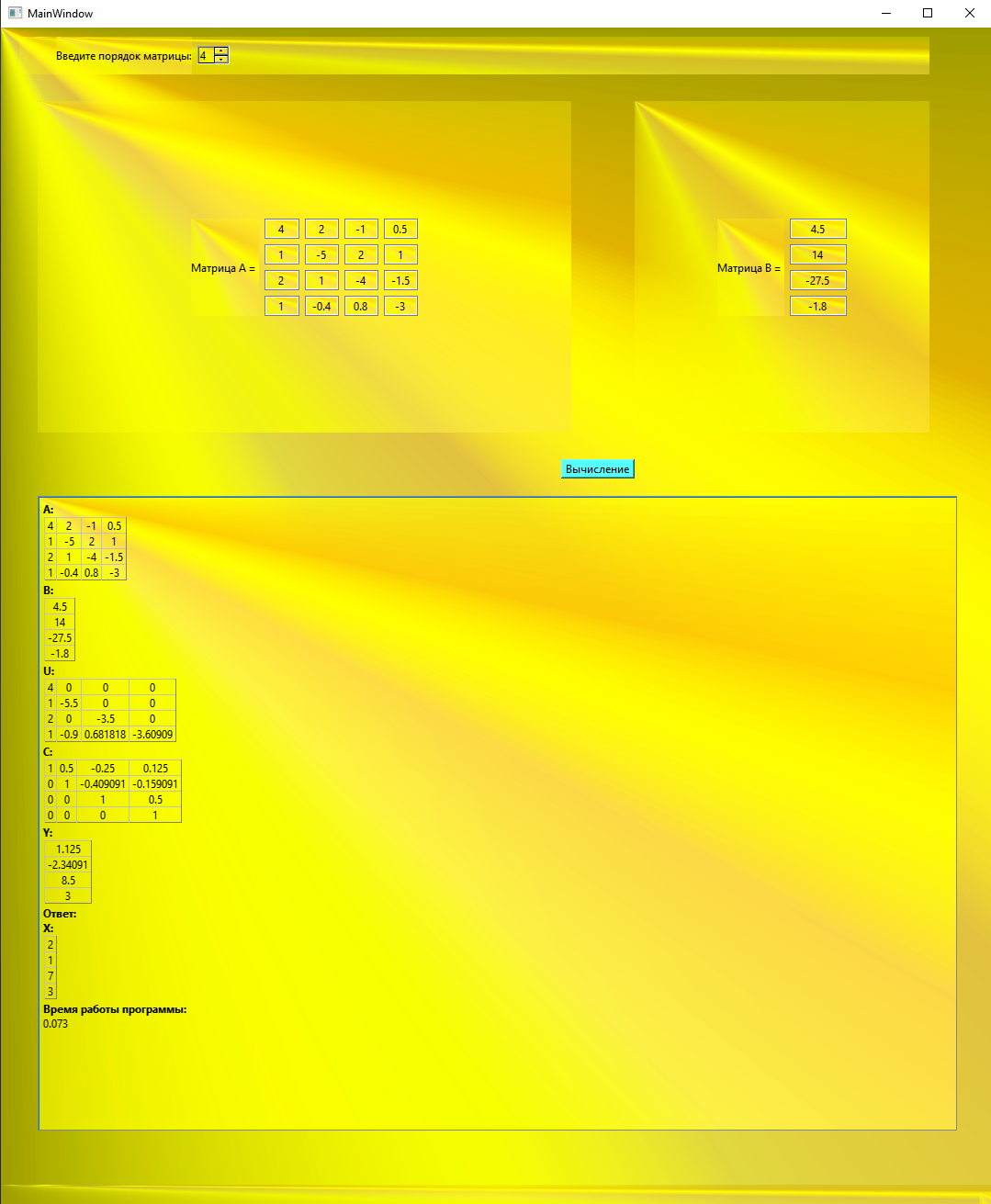
А = B =

Т. к. для элементов матрицы А выполняется условие:

А =

##### Решение с помощью точного метода Холецкого и приближенного метода простых итераций для СЛАУ с точностью 0.01.

###### 2.1) Точный метод:



###### 2.2) Приближенный метод:



### 3.3.2] Пункт 2

#### Дано:

А = B =

#### Задание:

• Сопоставить время расчетов и точность результатов.

#### Ответ:

Согласно п. 3.3.1 имеем:

При точном методе Холецкого Время расчетов составило: 0.073 сек. В результате получены точные значения.

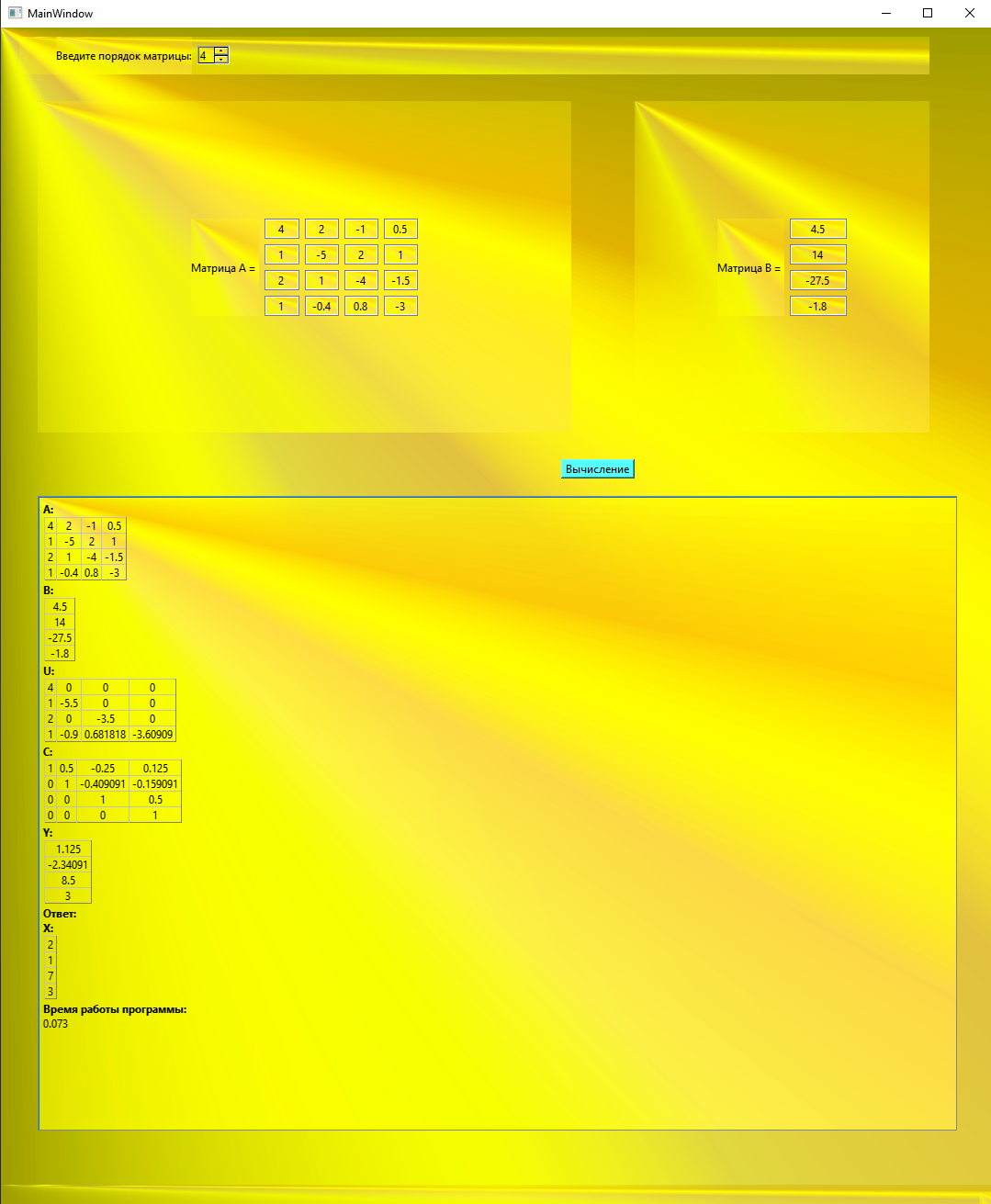
При приближенном методе простых итераций время расчетов составило: 0.405 сек. В результате получены приближенные значения.

# Приложения № 1 – Примеры работы программ.

4.1 Задание № 1

### Пункт № 1: (QProject\_2)

Ниже представлен пример работы программы.



*Рис.1 – Пример работы программы.*

4.2 Задание № 2

### Пункт № 1: (QProject\_3)

Ниже представлен пример работы программы.



*Рис.1 – Пример работы программы.*

4.3 Задание № 3

### Пункт № 1:

# Приложения № 2 – Листинг программ.

4.1 Задание № 1

### Пункт № 1: (QProject\_2)

Ниже представлен листинг основного кода, а также ссылка на GitHub.

*this*->ORD = ui->spinBox\_1->value();

QVector<QVector<double>> u(ORD, QVector<double>(ORD, 0));

QVector<QVector<double>> c(ORD, QVector<double>(ORD, 0));

QVector<bool> fLAG\_s\_Ethread(2, *true*);

U = u;

C = c;

FLAG\_s\_Ethread = fLAG\_s\_Ethread;

thread\_1 = *new* EThread("thread 1", 1, *\*this*);

thread\_2 = *new* EThread("thread 2", 2, *\*this*);

thread\_1->start(QThread::*TimeCriticalPriority*);

thread\_2->start(QThread::*TimeCriticalPriority*);

*while*(thread\_1->isRunning() || thread\_2->isRunning()){

SuspendThread(MainWindow::thread());

}

*if*(FLAG\_s\_Ethread[0] && FLAG\_s\_Ethread[1]){

qDebug() << "Матрицы U и C посчитаны успешно !";

}

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>U:</b>");

PrintMatrix\_2(U);

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>C:</b>");

PrintMatrix\_2(C);

QVector<double> y(ORD, 0);

QVector<double> x(ORD, 0);

Y = y;

X = x;

*for*(size\_t i = 0; i< ORD; i++){

*if*(i == 0){

Y[i] = setNewValue(qElem\_s\_2[i].qLineEdit->text()) / (U[i][i]);

}*else*{

*auto* SUMM = [*this*, i](*auto*& summ,int start, int end) -> double{

*if*(start > end) *return* 0;

*return* U[i][start]\*Y[start] + summ(summ,start + 1, end);

};

Y[i] = (setNewValue(qElem\_s\_2[i].qLineEdit->text()) - SUMM(SUMM,0, i - 1))/(U[i][i]);

}

}

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>Y:</b>");

PrintMatrix\_1(Y);

*for*(size\_t i = ORD-1; i < ORD; i--){

*if*(i == (ORD-1)){

X[i] = Y[i];

}*else*{

*auto* SUMM = [*this*, i](*auto*& summ,int start, int end) -> double{

*if*(start > end) *return* 0;

*return* C[i][start]\*X[start] + summ(summ,start + 1, end);

};

X[i] = Y[i] - SUMM(SUMM,i, ORD-1);

}

}

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>Ответ: </b>");

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>X: </b>");

PrintMatrix\_1(X);

QTime finish = QTime::currentTime();

double time = start.msecsTo(finish) / 1000.0;

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>Время работы программы: </b>");

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + QString::number(time));

Реализация в потоках:

void EThread::***run***()

{

*switch*(index)

{

*case*(1):

*for*(size\_t j = 0; j < mw.GetORD(); j++)

{

Resume();// Возобновление работы потока

*for*(size\_t i = 0; i < mw.GetORD(); i++)

{

*if*(j == 0){

mw.GU()[i][j] = mw.setNewValue(mw.GetQElem\_s\_1()[i][0].qLineEdit->text());

}*else* *if*(i>=j){

*auto* SUMM = [*this*, i, j](*auto*& summ,int start, int end) -> double{

*if*(start > end) *return* 0;

*return* mw.GU()[i][start]\* mw.GC()[start][j] + summ(summ,start + 1, end);

};

mw.GU()[i][j] = mw.setNewValue(mw.GetQElem\_s\_1()[i][j].qLineEdit->text()) - SUMM(SUMM,0, j - 1);

}

}

Pause();// Постановка потока на паузу.

}

*break*;

*case*(2):

*for*(size\_t i = 0; i < mw.GetORD(); i++) {

Resume();

*for*(size\_t j = 0; j < mw.GetORD(); j++){

*if*(i == j){

mw.GC()[i][j] = 1;

}*else* *if*(j>i){

*auto* SUMM = [*this*, i, j](*auto*& summ,int start, int end) -> double{

*if*(start > end) *return* 0;

*return* mw.GU()[i][start]\* mw.GC()[start][j] + summ(summ,start + 1, end);

};

mw.GC()[i][j] = ((mw.setNewValue(mw.GetQElem\_s\_1()[i][j].qLineEdit->text()) - SUMM(SUMM,0, j - 1))/(mw.GU()[i][i]));

}

*if*((i == 0) && (j >0)){

mw.GC()[i][j] = mw.setNewValue(mw.GetQElem\_s\_1()[i][j].qLineEdit->text()) / mw.GU()[i][i];

}

}

Pause();

}

*break*;

*default*:

*break*;

};

}

Листинг кода № 2 – Основной код.

Ссылка на gitHab: (QTProject\_2)// <https://github.com/MineevS/CHM_3_5.git>

4.2 Задание № 2

### Пункт № 1: (QProject\_3)

Ниже представлен листинг основного кода, а также ссылка на GitHub.

*Основной Код:*

*//* *Метод* *прибовления* *к* *матрице* *константы.*

*auto* F\_2\_1 = [](QVector<QVector<double>>& M,double value) -> QVector<QVector<double>> {

QVector<QVector<double>> m;

foreach(*auto* row, M){

QVector<double> m\_i(row.size()/ 8);

foreach(*auto* elem, row){

m\_i.push\_back(elem + value);

};

m.push\_back(m\_i);

};

*return* m;

};

*//* *Метод* *умножения* *матрице* *на* *константу.*

*auto* F\_1\_2 = [*this*](QVector<double>& M, double value) -> QVector<double> {

QVector<double> m;

foreach(*auto* elem, M){

m.push\_back(value \* elem);

};

*return* m;

};

*//* *Метод* *сложения* *двух* *матриц.*

*auto* F\_1\_5 = [*this*](QVector<double>& M,QVector<double>& N) -> QVector<double> {

QVector<double> m;

*for*(size\_t i = 0; i < Count.count(); i++){

m.push\_back(M[i] + N[i]);

};

*return* m;

};

Листинг кода № 2 – Основной код.

Ссылка на gitHab: QProject\_4// <https://github.com/MineevS/CHM_3_5.git>

*auto* norm\_m\_1 = [](QVector<QVector<double>>& M) -> double {

*auto* V = [&](QVector<QVector<double>>& M) -> QVector<double> {

*auto* summ\_b = [](*auto* summ\_b, QVector<double> M, size\_t k) -> double {

*if*(k == M.count()) *return* 0;

*return* qAbs(M[k]) + summ\_b(summ\_b, M, k + 1);

};

QVector<double> v(M.count() , 0);

*for*(size\_t i = 0; i < M.count(); i++){

v[i] = summ\_b(summ\_b, M[i], 0);

}

*return* v;

};

***////////////////////////////////////////////////////////////***

*auto* max = [](QVector<double> vec) -> double {

*return* \*std::max\_element(vec.begin(),vec.end());

};

*return* max(V(M));

};

*auto* norm\_i\_1 = [](QVector<QVector<double>>& M) -> double {

*auto* V = [&](QVector<QVector<double>>& M) -> QVector<double> {

*auto* T = [](QVector<QVector<double>>& M) -> QVector<QVector<double>> {

QVector<QVector<double>> tM(M.count(), QVector<double>(M.count(),0));

*for*(size\_t i = 0; i < M.count(); i++){

*for*(size\_t j = 0; j < M.count(); j++){

tM[i][j] = M[j][i];

}

}

*return* tM;

};

QVector<QVector<double>> Tm = T(M);

*auto* summ\_b = [](*auto* summ\_b, QVector<double> M, size\_t k) -> double {

*if*(k == M.count()) *return* 0;

*return* qAbs(M[k]) + summ\_b(summ\_b, M, k + 1);

};

QVector<double> v(M.count() , 0);

*for*(size\_t i = 0; i < M.count(); i++){

v[i] = summ\_b(summ\_b, Tm[i], 0);

}

*return* v;

};

***////////////////////////////////////////////////////////////***

*auto* max = [](QVector<double> vec) -> double {

*return* \*std::max\_element(vec.begin(), vec.end());

};

*return* max(V(M));

};

*auto* norm\_k\_1 = [](QVector<QVector<double>>& M) -> double {

*auto* SUMM = [](QVector<QVector<double>>& M) -> double {

double x = 0;

*for*(size\_t i = 0; i < M.count(); i++){

*for*(size\_t j = 0; j < M[i].count(); j++){

x += qPow(qAbs(M[i][j]), 2);

}

}

*return* x;

};

*return* qSqrt(SUMM(M));

};

double norm\_m(1), norm\_i(1), norm\_k(1);

QVector<QVector<double>> R;

QVector<double> r;

double ip = 0;

QVector<bool> count(A.count(), *false*);

Count = count;

*//* *Метод* *сортировки* *строк* *матрицы* *в* *зависимости* *от* *того* *чтобы*

*//* *на* *главной* *диагонали* *по* *возможности* *был* *максимальный* *элемент* *по* *отношению* *к* *элементам* *соответствующей* *строи/столбца.*

*auto* F\_1\_4 = [](QVector<QVector<double>>& M, QVector<double>& N,QVector<bool>& count) -> void {

*auto* SummAbs = [](QVector<double>& M, qint16 k) -> double {

qint16 x(0);

*for*(size\_t i = 0; i < M.count(); i++){

*if*(i != k){

x += qAbs(M[i]);

}

}

*return* x;

};

*for*(size\_t i = 0; i < M.count(); i++){

*for*(size\_t j = 0; j < M[i].count(); j++){

*if*(qAbs(M[j][i]) > SummAbs(M[j], i)){

qDebug() << j << " : " << i<< " : " << M[j][i] << " : " << SummAbs(M[j], i) << " : " << "\033[32m" << *true* << "\033[0m";

count[i] = *true*;

std::swap(*M[i]*, *M[j]*);

std::swap(*N[i]*, *N[j]*);

}*else*{

qDebug() << j << " : " << i << " : " << M[j][i] << " : " << SummAbs(M[j], i) << " : " << *false*;

}

}

}

*return*;

};

F\_1\_4(A, *B*, count);

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>A:</b>");

PrintMatrix\_2(A);

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>B:</b>");

PrintMatrix\_1(B);

F\_1\_4(A, *B*, count);

*//* *Метод* *проверки* *наличия* *строк* *для* *которых* *невыполняется* *условие* *(2.4)*

*auto* Cou = [&]()-> bool { bool x = *false*;

foreach(*auto* elem, Count){

*if*(!elem) x =!elem;

};

*return* x;

};

Count = count;

*//* *Метод* *сортировки* *строк* *в* *зависимости* *от* *наличия* *на* *главной* *диагонали*

*//* *максимального* *по* *модулю* *элемента* *по* *стравнению* *с* *элементами* *в* *строке.*

*auto* F\_1\_7 = [](QVector<QVector<double>>& M, QVector<double>& N, QVector<bool>& count) -> void {

*for*(size\_t i = 0; i < M.count(); i++){

*if*(!count[i]){

*for*(size\_t j = 0; j < M.count(); j++){

*if*(!count[j]){

*if*(qAbs(M[i][j]) < qAbs(M[j][i])){

std::swap(*M[i]*, *M[j]*);

std::swap(*N[i]*, *N[j]*);}}}}}

};

*//* *Метод* *сортировки* *строк* *в* *зависимости* *от* *наличия* *на* *главной* *диагонали*

*//* *максимального* *по* *модулю* *элемента* *по* *стравнению* *с* *элементами* *в* *строке.*

*auto* F\_1\_7 = [](QVector<QVector<double>>& M, QVector<double>& N, QVector<bool>& count) -> void {

*for*(size\_t i = 0; i < M.count(); i++){

*if*(!count[i]){

*for*(size\_t j = 0; j < M.count(); j++){

*if*(!count[j]){

*if*(qAbs(M[i][j]) < qAbs(M[j][i])){

std::swap(*M[i]*, *M[j]*);

std::swap(*N[i]*, *N[j]*);

}

}

}

}

}

};

F\_1\_7(A, *B*, count); *//* *Перестановка* *строк* *по* *возрастанию* *для* *недиагонолизироанных* *строк.*

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>A:</b>");

PrintMatrix\_2(A);

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>B:</b>");

PrintMatrix\_1(B);

F\_1\_4(A, *B*, count);

*//* *Цикл* *элементарных* *преобразований(диагонализации)* *по* *строке* *для* *строк,* *которые* *не* *выполняется* *условие* *(2.4).*

*while*(Cou()){

*for*(size\_t i = 0; i < Count.count(); i++){

*if*(!Count[i]){

QVector<qint16> H(A[i].count(),0);

QVector<quint16> h(A[i].count(), 0);h[i] = 1;quint16 MAX = 10;

QVector<size\_t> ITER\_BUFF\_F(A[i].count(), 0);

*auto* ITER\_F = [&](*auto* iter\_f, QVector<size\_t>& iter\_buff\_f, size\_t f) -> void {

*if*(f == 0){

QVector<int> ITER\_BUFF\_D(A[i].count(), 0);

QVector<double> C(A[i].count(),0);

*auto* ITER\_D = [&](*auto* iter\_d, QVector<size\_t>& iter\_buff\_f,QVector<int>& iter\_buff\_d, int di) -> void {

*if*(di == 0){

*for*(size\_t z = 0; z < H.count(); z++)

{

H[z] = iter\_buff\_d[z] \* iter\_buff\_f[z];

}

*auto* iterat\_A = [&](*auto* it, QVector<qint16> H, int i) -> QVector<double> {

*if*(i >= H.count()) *return* QVector<double>(A[i-1].count(), 0);

QVector<double> F12 = F\_1\_2(A[i], H[i]);

QVector<double> It = it(it, H, i + 1);

*return* F\_1\_5( F12 , It);

};

C = iterat\_A(iterat\_A, H, 0);

*auto* SummAbs = [](QVector<double>& M, qint16 k) -> double {

qint16 x(0);

*for*(size\_t i = 0; i < M.count(); i++){

*if*(i != k){ x += qAbs(M[i]);}

}

*return* x;

};

*if*(C.count() > 0){

*if*(qAbs(C[i]) > SummAbs(C, i)){

*auto* F\_1\_6 = [&](QVector<double>& M, QVector<double>& N) -> void {

*for*(size\_t u = 0; u < C.count() ; u++){

M[u] = N[u];

};

};

*auto* iterat\_B = [&](*auto* it, QVector<qint16> H, qint16 j, size\_t i) -> double {

*if*(i >= B.count()) *return* 0;

B[j] = B[i] \* H[i] + it(it,H,j, i + 1);

*return* B[j];

};

iterat\_B(iterat\_B, H, i, 0);

*//* *Получение* *в* *консоли* *информации* *к* *какой* *строке* *какие* *коэффициенты*

*//* *были* *применены* *для* *достижения* *в* *строке* *изсеняемой* *диагонализированного* *вида.*

qDebug() << "H[0] : " << H[0]; qDebug() << "H[1] : " << H[1];

qDebug() << "H[2] : " << H[2]; qDebug() << "H[3] : " << H[3];

F\_1\_6(A[i] , C); Count[i] = *true*;

}}

}*else* {

*for*(int s = -1; (!Count[i]) && (s < 2); s+=2){

iter\_buff\_d[di - 1] = s;

iter\_d(iter\_d, iter\_buff\_f, iter\_buff\_d, di - 1);

}}};

ITER\_D(ITER\_D, iter\_buff\_f, ITER\_BUFF\_D, A.count());

}*else* {

*for*(size\_t t = h[f - 1];(!Count[i]) && (t < MAX); t++){iter\_buff\_f[f - 1] = t;iter\_f(iter\_f, iter\_buff\_f, f - 1);}}};

ITER\_F(ITER\_F, ITER\_BUFF\_F, A.count());};};

*if*(Cou()){

qDebug() << "Программа не смогла деагонализировать матрицу. "

"Попробуйте увеличить глубину ассоциативности, указав значение выше "

"установленного в переменной MAX, строки кода: [673].";

exit(0);

};

};

///////////////////////////////////////

*//* *Метод* *выражения* *элементов* *матрицы* *в* *зависимости* *от* *элементов,* *стоящих* *на* *главной* *диагонали.*

*auto* F\_2\_2 = [](QVector<QVector<double>>& M) -> QVector<QVector<double>> {

QVector<QVector<double>> m;

*auto* F\_2\_4 = [](QVector<double> M, int value)-> QVector<double> {

QVector<double> m;

*for*(size\_t i = 0; i < M.count(); i++){

m.push\_back((M[i] / value));

}

*return* m;

};

*for*(size\_t i = 0; i < M.count(); i++){

m.push\_back(F\_2\_4(M[i], -M[i][i]));

m[i][i] = 0;

}

*return* m;

};

R = F\_2\_2(A);

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>R:</b>");

PrintMatrix\_2(R);

*//* *Метод* *деления* *элементов* *матрицы* *по* *строке* *на* *соответствующий* *элемент,* *стоящий* *на* *главной* *диагонали.*

*auto* F\_2\_7 = [&](QVector<double> M) -> QVector<double>{

QVector<double> m(M.count(),0);

*for*(size\_t n = 0; n < M.count(); n++){

m[n] = M[n] / A[n][n];

}

*return* m;

};

r = F\_2\_7(B);

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>r:</b>");

PrintMatrix\_1(r);

***////////////////////////////////////***

norm\_m = norm\_m\_1(R);norm\_i = norm\_i\_1(R);norm\_k = norm\_k\_1(R);

qDebug() << norm\_m; qDebug() << norm\_i;qDebug() << norm\_k;

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>Норма m: </b>" + QString::number(norm\_m));

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>Норма i: </b>" + QString::number(norm\_i) + "< 1;");

ui->textBrowser\_1->setHtml(ui->textBrowser\_1->toHtml() + "<b>Норма k: </b>" + QString::number(norm\_k) + "< 1;");

***///////////////////////////////////***

QVector<double> x\_0 = r;QVector<double> x\_i(B.count(), 0);

QVector<double> x\_j(B.count(), 0);

double eps = Epsilon.qLineEdit->text().toDouble();

*//* *Метод* *проверки* *выхода* *из* *цикла* *итерации* *по* *достижению* *точности.*

*auto* FAb = [&](QVector<double>& M, QVector<double>& N, double& eps) -> bool {

bool flg = *false*;

*for*(size\_t i = 0; i < M.count(); i++){

*if*(fabs(M[i] - N[i]) >= eps){

flg = *true*;

}

}

*return* flg;

};

*do*{

*if*(counter == 0){

x\_i = x\_0;

}*else*{

x\_i = x\_j;

}

counter++;

*for*(size\_t k = 0; k < x\_j.count(); k++)

{

*auto* SUMM = [&](*auto* summ,QVector<QVector<double>>& M,QVector<double>& N, size\_t start) -> double {

*if*(start == M.count()){

*return* 0;

}*else* *if*(k == start){

*return* 0 + summ(summ, M , N, start + 1);

}*else*{

*return* M[k][start]\*N[start] + summ(summ, M , N, start + 1);

}

};

x\_j[k] = R[k][k] - SUMM(SUMM, *R*, x\_i, 0);

}

} *while*(FAb(x\_i, *x\_j*, eps));

4.3 Задание № 3

### Пункт № 1.

---

# Список использованных источников

1. <https://vega.fcyb.mirea.ru/disc/disc.php?id=163>
2. <https://online-edu.mirea.ru/course/view.php?id=2154>

# Как раскрыть все заголовки в документе:

