Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Кафедра телекоммуникационных технологий и сетей

И.В. Семушин, Ю.В. Цыганова

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
для семинарских (практических) занятий, лабораторного практикума
и самостоятельной работы
по дисциплинам

«Численные методы» и «Вычислительная математика»

для студентов направлений
09.03.02 «Информационные системы и технологии»,
11.03.02 «Инфокоммуникационные технологии и системы связи»

Ульяновск
2019

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплинам «Численные методы» и «Вычислительная математика» / составители: И.В. Семушин, Ю.В. Цыганова - Ульяновск: УлГУ, 2019 – 108 с.

Настоящие методические рекомендации предназначены для студентов направлений обучения 09.03.02 «Информационные системы и технологии», 11.03.02 «Инфокоммуникационные технологии и системы связи». В работе приведены литература по дисциплине, темы дисциплины и задания в рамках каждой темы, рекомендации по изучению теоретического материала, задания для самостоятельной работы, задания для лабораторного практикума и рекомендации по их выполнению.

Студентам всех форм обучения следует использовать данные методические рекомендации при подготовке к семинарам, самостоятельной подготовке, а также промежуточной аттестации по дисциплинам «Численные методы» и «Вычислительная математика».

Рекомендованы к введению в образовательный процесс

Учёным советом факультета математики, информационных и авиационных технологий УлГУ

протокол № 2/19 от «19» марта 2019 г.

Оглавление

Введение

Кафедра «Информационные технологии» УлГУ рекомендует данное учебно-методическое пособие для студентов высших учебных заведений, обучающихся по направлениям:

- 09.03.02 «Информационные системы и технологии»;
- 11.03.02 «Инфокоммуникационные технологии и системы»

при изучении дисциплин «Численные методы» и «Вычислительная математика».

Базовый курс «Численные методы» по многим специальностям и направлениям подготовки в университетах преследует следующие общие цели:

- заложить базовые умения и навыки в области разработки вычислительных алгоритмов решения задач, возникающих в процессе математического моделирования законов реального мира;
- обеспечить понимание основных идей вычислительных методов, особенностей и условий их применения;
- подготовить студентов к применению этих знаний в дальнейшей учебе и практической деятельности.

При изучении базового курса «Численные методы» значительное время отводят на «Вычислительные методы алгебры», или «Численные методы I». Согласно требованиям Государственного образовательного стандарта, к фундаментальной части численных методов относят следующие темы из раздела «Вычислительная линейная алгебра» (ВЛА):

- Тема 1. Методы исключения в решении систем;
- Тема 2. Разложения Холесского положительно определенных матриц;
- Тема 3. Методы ортогональных преобразований;
- Тема 4. Итерационные методы решения систем;
- Тема 5. Методы решения проблемы собственных значений матриц.

Содержание данного учебно-методического пособия включают в себя первые три темы из этого списка.

Базовым учебником, к которому данное пособие служит дополнением, является книга И. В. Семушина «Вычислительные методы алгебры и

оценивания». Ульяновск: Изд-во УлГТУ, 2011. 366 с. ISBN 978-5-9795-0902-0.

Значение «Численных методов» во многих областях науки и техники трудно переоценить, и оно постоянно растет. В связи с этим очень важно, чтобы студенты, готовящиеся стать специалистами в области математического моделирования, численных методов и комплексов программ, обладали подлинно глубокими знаниями, т. е. знаниями, имеющими для них практическую ценность в их будущей деятельности. Такое знание достигается не схоластическим изучением теории и не решением элементарных задач в классе, а реальной проектной работой по созданию серьезных программных продуктов высокого профессионального уровня, воплощающих эти численные методы. В связи с этим данное пособие, как и указанный выше базовый учебник, использует так называемый *проектно-ориентированный подход*[1], при котором студенты получают необходимый теоретический материал и закрепляют эти знания в практических лабораторных проектах. Надеемся, что при таком подходе к преподаванию и изучению студент лучше поймет и оценит этот важный предмет.

Почему же для реализации лабораторных проектов мы выбрали язык MATLAB? Система математических расчетов MATLAB (сокращение от MATrix LABoratory) разработана фирмой The Math Works, Inc. (США, г. Нэтик, шт. Массачусетс) и является интерактивной системой для выполнения инженерных и научных расчетов, которая ориентирована на работу с матричным представлением данных.

Сначала пакет MATLAB предназначался для матричных вычислений и был только удобной оболочкой для имеющихся библиотек программ, но впоследствии его возможности существенно выросли.

MATLAB применяется для решения задач, возникающих в различных прикладных областях. Обработка сигналов и изображений, исследование и расчет физических процессов, визуализация данных, статистический анализ, матричный анализ, оптимизация, нейронные сети, нечеткая логика, моделирование нелинейных динамических систем – вот далеко не полный перечень задач, которые могут быть эффективно решены в системе MATLAB.

MATLAB – это одновременно и операционная среда, и язык программирования. На языке MATLAB могут быть написаны программы для

[1] I. V. Semushin, Ju.V. Tsyganova, V. V. Ugarov, “Computational and Soft Skills Development Through the Project Based Learning,” *Lecture Notes in Computer Science*. – 2003. – Vol. 2658, Pt. 2. – Pp. 1098-1106.

многократного использования. Пользователь может сам написать специализированные функции и программы, которые оформляются в виде m-файлов. Наличие в MATLAB обширной библиотеки графических, вычислительных и сервисных функций способствует быстрому созданию приложений для исследования и решения поставленной задачи. Визуальная среда программирования позволяет при наличии некоторого опыта программировать приложения с графическим интерфейсом пользователя.

Программы, написанные на современных языках программирования высокого уровня, могут прекрасно взаимодействовать с приложениями, созданными в системе MATLAB.

Таким образом, MATLAB является наиболее подходящим программным инструментом для овладения студентами практическими навыками реализации вычислительных алгоритмов и проведения численных экспериментов.

Авторы искренне надеются, что данное учебно-методическое пособие поможет студентам овладеть практическими навыками реализации алгоритмов вычислительной линейной алгебры, а также навыками создания программных проектов, предназначенных для научно-исследовательских целей.

1 Основы работы в MATLAB

1.1 Командная строка MATLAB

При запуске программы MATLAB на экране появляется рабочая среда, изображенная на рис. 1.

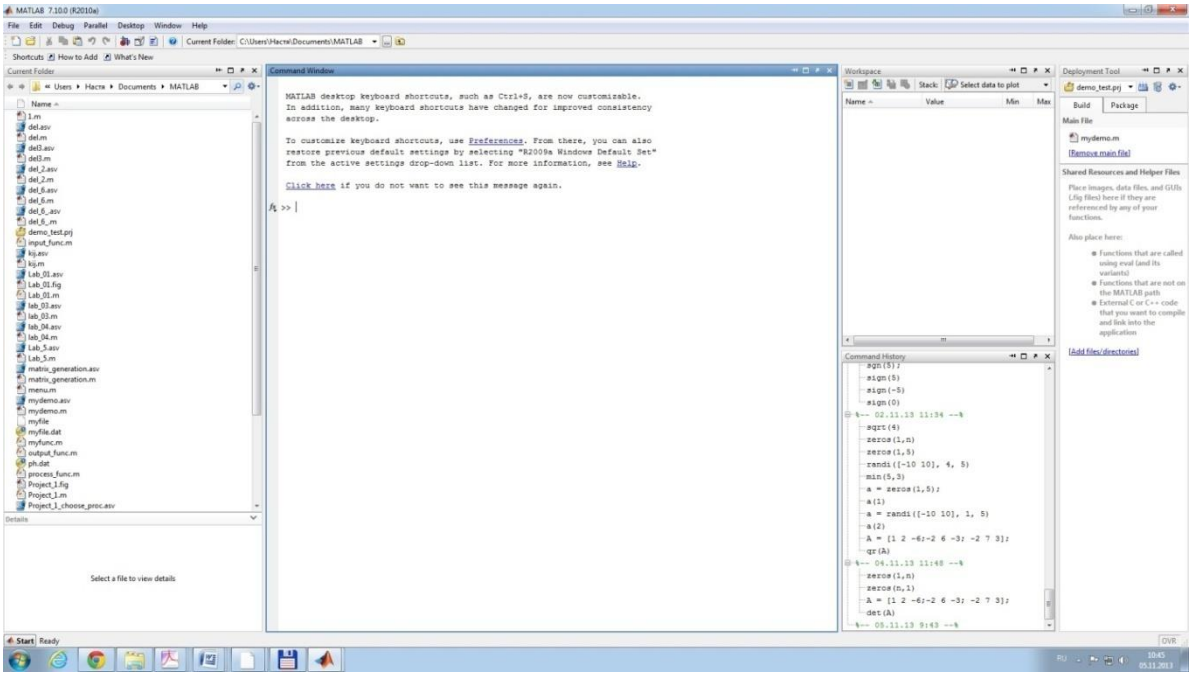

Рис. 1. Рабочая среда MATLAB

Элемент *Command Window* является командной строкой MATLAB. Команды MATLAB можно вводить непосредственно в командную строку либо выполнять их с помощью созданных пользователем программ. В обоих случаях результат выполнения можно будет увидеть в окне *Command Window*. Получить помощь по доступным функциям можно, набрав в командной строке слово help.

1.2 Создание программы

Создание m-файлов

Для создания собственных программ в MATLAB существуют специальные m-файлы, которые бывают двух типов: файлы-программы (файлы-скрипты) (Script m-Files), содержащие последовательность команд, и файлы-функции (Function m-Files), в которых описываются функции, определяемые пользователем.

Для создания файла-программы необходимо в меню выбрать последовательность *File→New→Script* (см. рис. 2).

Файлы-программы являются последовательностью команд MATLAB, они не имеют входных и выходных аргументов.

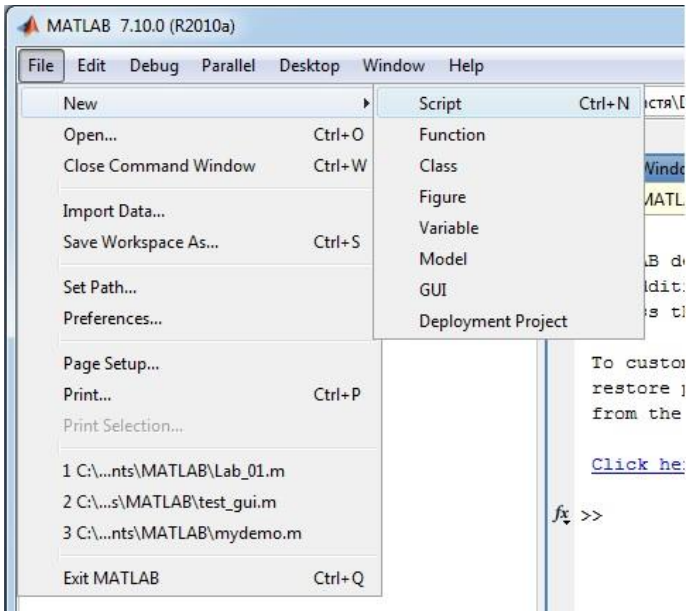

Рис. 2. Создание файла с программой

При программировании собственных приложений в MATLAB необходимо уметь составлять файлы-функции, которые производят необходимые действия с входными аргументами и возвращают результат в выходных аргументах.

Для создания файла-функции необходимо в меню выбрать последовательность *File→New→Function* (см. рис. 2).

Текст m-функции должен начинаться с заголовка, после которого следует тело функции. Заголовок определяет «интерфейс» функции (т. е. способ взаимодействия с ней) и устроен следующим образом:

```
function [output_args]=function_name(input_args)
```

Здесь function_name – имя функции; input_args – входные параметры (аргументы функции); output_args – выходные параметры (значения функции).

Указанное в заголовке имя функции (function_name) должно совпадать с именем файла, в который будет записан текст функции. Для данного примера это будет файл function_name.m.

Для вызова функции в главной программе необходимо указать:

```
>>function_name(input_args);
```

Каталог, в котором содержатся файлы-функции, должен быть текущим, или путь к нему должен быть добавлен в пути поиска, иначе MATLAB не найдет функцию или вызовет вместо нее другую с тем же именем (если она находится в каталогах, доступных для поиска).

Установка путей

Для работы в MATLAB необходимо определить текущий каталог и пути поиска. Это можно сделать либо при помощи соответствующих диалоговых окон, либо с помощью команд из командной строки. Текущий каталог определяется в диалоговом окне *Current Folder* рабочей среды. Содержимое текущего каталога отображается в таблице файлов.

Определение путей поиска производится в диалоговом окне *Set Path* навигатора путей, доступ к которому осуществляется из пункта *Set Path* меню *File* рабочей среды. Для добавления каталога нажмите кнопку *Add Folder* и в появившемся диалоговом окне выберите требуемый каталог. Добавление каталога со всеми его подкаталогами осуществляется при нажатии на кнопку *Add with Subfolders*. Порядок поиска соответствует расположению путей в этом поле, первым просматривается каталог, путь к которому размещен вверху списка. Порядок поиска можно изменить или вообще удалить путь к какому-либо каталогу, для чего выделите каталог в поле *MATLAB search path* и определите его положение при помощи следующих кнопок: Move to Top – поместить вверх списка; *Move Up* – переместить вверх на одну позицию; *Remove* – удалить из списка; *Move Down* – переместить вниз на одну позицию; *Move to Bottom* – поместить вниз списка. После внесения изменений следует сохранить информацию о путях поиска, нажав кнопку *Save*. При помощи кнопки *Default* можно восстановить стандартные установки. Кнопка *Revert* предназначена для возврата к сохраненным данным.

Сохранение и запуск программы

Сохранить файл-программу или файл-функцию можно либо из меню *File→Save as...*, либо с помощью кнопки *Save* (см. рис. 3) или нажав кнопку *Save and run* (см. рис. 4).

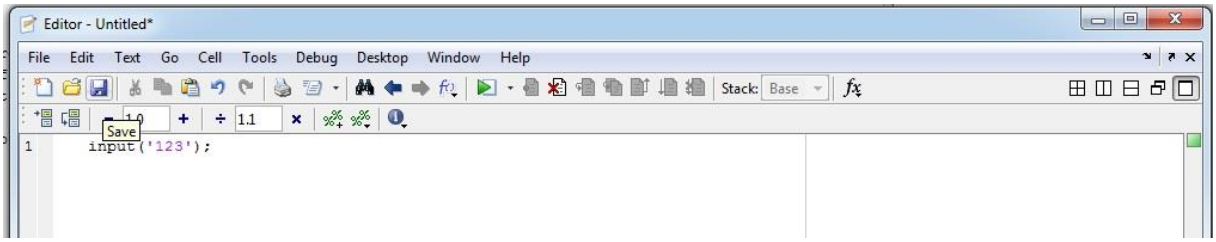

Рис. 3. Сохранение файла с программой

Во втором случае одновременно с сохранением программы произойдет ее запуск.

Запустить программу или функцию можно также из командной строки, набрав в ней имя m-файла.

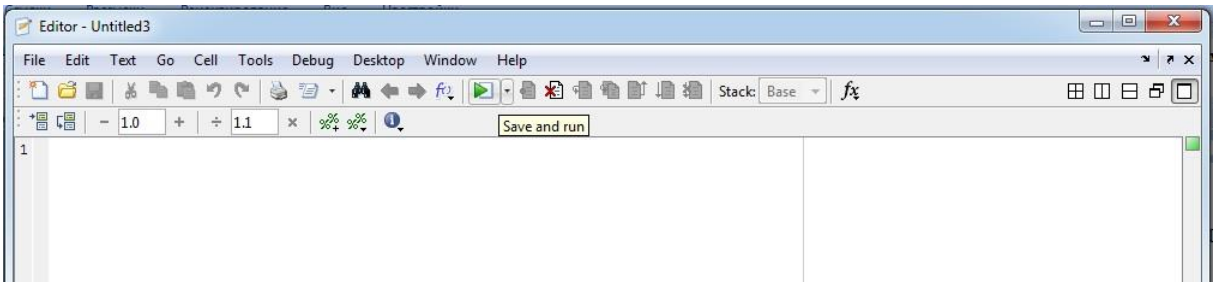

Рис. 4. Сохранение и запуск файла с программой

1.3 Создание GUI-приложения

Создать GUI-приложение в MATLAB можно, выбрав в меню пункт *File→New→GUI*, либо набрав в командной строке команду guide.

MATLAB предложит Вам на выбор несколько шаблонов графического интерфейса (см. рис. 5).

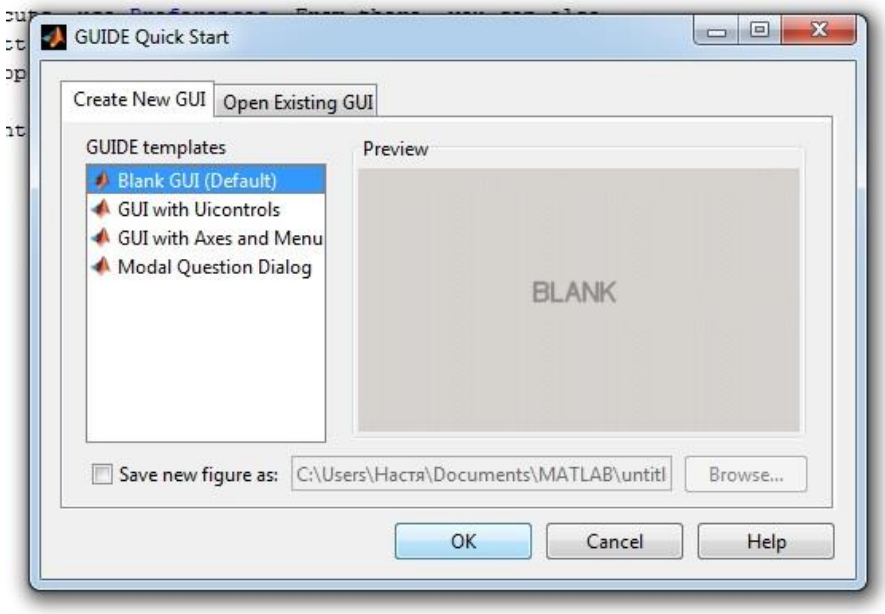

Рис. 5. Шаблоны графического интерфейса

Выбрав шаблон *Blank* – пустой шаблон, можно самостоятельно заполнить интерфейс нужными компонентами (см. рис. 6). Файл шаблона графического интерфейса имеет расширение `*.fig`.

При нажатии на кнопку *Run Figure* (см. рис. 7) MATLAB предложит сохранить файлы программы (два файла с расширением *.fig и *.m). После сохранения MATLAB запустит GUI-приложение и откроет в окне редактора файл-программу данного приложения.

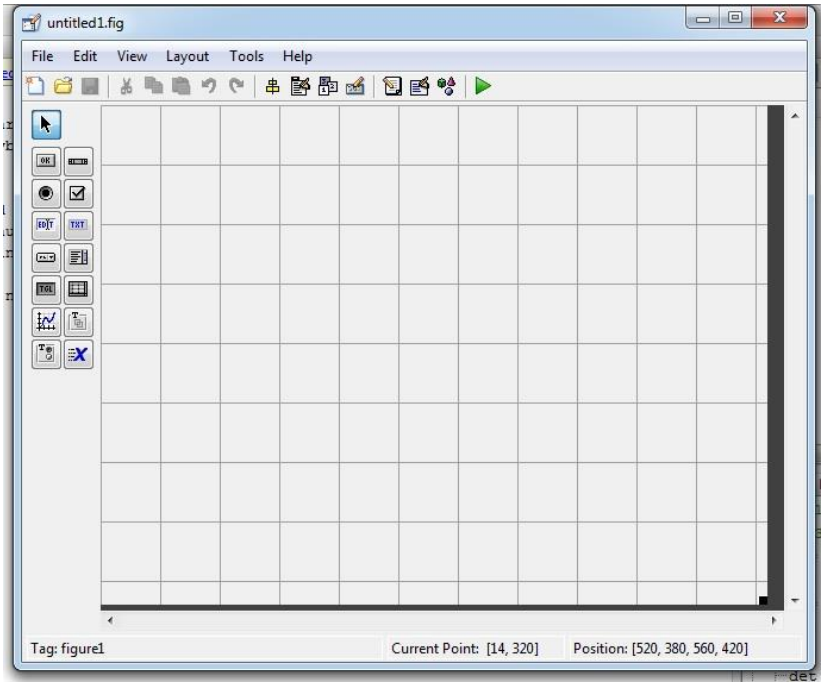

Рис. 6. Доступные компоненты шаблона

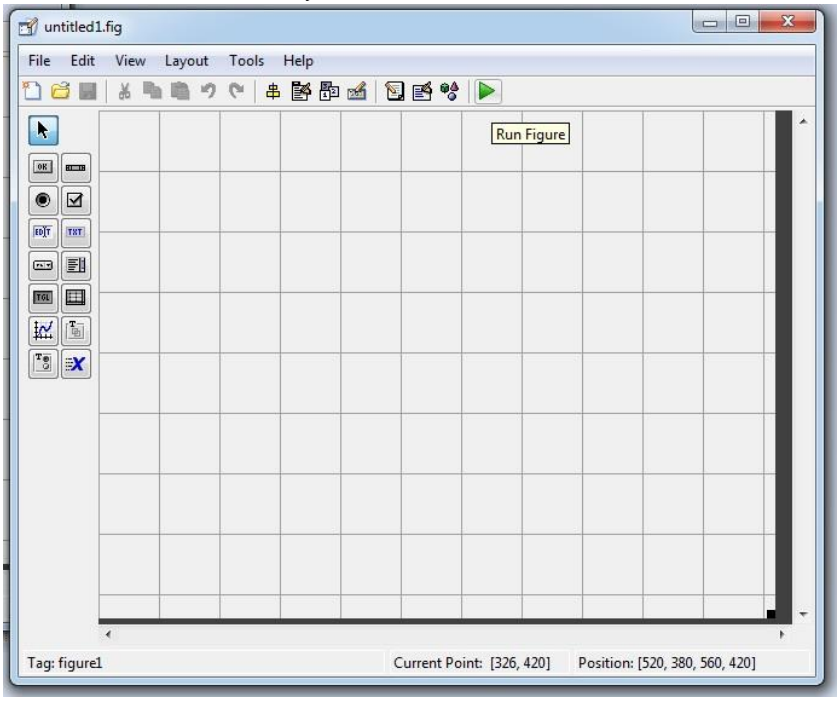

Рис. 7. Запуск GUI-приложения

Более подробную информацию о возможностях MATLAB можно найти в дополнительной литературе, см. с. 71.

2 Общие требования к выполнению и сдаче лабораторных проектов

Лабораторный практикум рассчитан на один семестр обучения и включает в себя три лабораторных проекта. Баллы за отдельные проекты складываются и тем самым образуют общую оценку за этот вид Вашей учебной работы.

Установлены следующие сроки сдачи лабораторных проектов.

1. Весенний семестр обучения:
   - проект № 1 – с 20 по 31 марта;
   - проект № 2 – с 20 по 30 апреля;
   - проект № 3 – с 20 по 31 мая.
2. Осенний семестр обучения:
   - проект № 1 – с 20 по 31 октября;
   - проект № 2 – с 20 по 30 ноября;
   - проект № 3 – с 20 по 30 декабря.

Любая сдача лабораторного задания позже установленного срока влечет уменьшение Вашей общей оценки на 10 баллов.

За каждое невыполненное задание начисляется 0 баллов.

За семестр Вам предлагается выполнить три лабораторных проекта. Максимальное количество баллов, которое можно заработать за все лабораторные проекты, составляет 100. Эти 100 баллов распределяются определенным образом между общим числом выданных лабораторных заданий и существенным образом влияют на экзаменационную оценку. Более подробно об этом см. базовый учебник, с. 16-23.

Предлагаемые каждому студенту три лабораторных проекта соответствуют первым трем темам из списка на с. 4, 5. За выполненное безупречно и в полном объеме задание по теме № 1 (лабораторный проект № 1) студент заработает 50 баллов. Это задание является базовым и поэтому должно предшествовать всем остальным. Далее, за выполненное безупречно и в полном объеме задание по теме № 2 (лабораторный проект № 2) студент заработает 25 баллов, а за выполненное безупречно и в полном объеме задание по теме № 3 (лабораторный проект № 3) – также 25 баллов. Для каждого лабораторного проекта необходимо:

1. Написать и отладить программу, реализующую ваш вариант задания.

2. Предусмотреть сообщения, предупреждающие о невозможности решения указанных задач с заданной матрицей.
3. Уделить особое внимание эффективности программы (в смысле экономии оперативной памяти).
4. Предусмотреть пошаговое выполнение алгоритмов с выводом результата на экран.
5. Написать пояснительную записку о проделанной работе (см. Приложение).
6. Представить преподавателю печатный текст пояснительной записки и CD диск, содержащий в отдельных директориях весь проект: Программный код проекта (готовый к запуску) + Инструкция пользователя + пояснительная записка в виде *.doc файла.

Заработанное число баллов за каждое задание будет уменьшено, если письменный отчет и устная защита работы не отвечает всем указанным требованиям или не демонстрирует самостоятельное исполнение.

Важно! Индивидуальный вариант для каждого студента назначается равным N+1, где N – порядковый номер в списке группы (в алфавитном порядке).

Далее следуют описания лабораторных проектов.

3 Лабораторный проект № 1
«Стандартные алгоритмы LU-разложения»

3.1 Теория

Перед выполнением лабораторного проекта рекомендуем подробно ознакомиться с материалом базового учебника, с. 27-52.

Алгоритмы метода Гаусса

Стандартный метод Гаусса, осуществляющий *LU*-разложение матрицы *A*, заключается в последовательном исключении переменных из уравнений системы

$$Ax=f \qquad (1)$$

Любой полный шаг алгоритма метода Гаусса состоит из двух действий: нормировка ведущей строки матрицы и обновление (серия вычитаний) в активной подматрице.

Нормировка – это деление ведущего уравнения системы на ведущий элемент.

Обновление в активной подматрице – это серия вычитаний ведущего уравнения из всех нижележащих, чтобы исключить из них неизвестную x_i (i – номер ведущего уравнения).

В результате работы алгоритмом факторизации Гаусса получим разложение матрицы *A* на верхнюю треугольную матрицу $\overline{U}$ и нижнюю треугольную матрицу *L*. Матрица $\overline{U}$ представляет собой эквивалентный вид системы (1), который она приобретет по завершении этого процесса. Матрица *L* представляет собой историю нормировок и вычитаний в процессе исключения неизвестных.

Алгоритм 1. $L\overline{U}$ -разложение матрицы A
(на основе гауссова исключения по столбцам):

Для k=1 до n
Выбираем главный элемент в $A^{(k-1)}$.
Нормируем первую строку матрицы $A^{(k-1)}$.
Для i=k+1 до n

Вычитаем первую строку матрицы $A^{(k-1)}$, умноженную на $a_{ik}^{(k-1)}$, из i-й строки.

Здесь $A^{(k-1)}$ – активная подматрица матрицы A после (k–1)-го шага метода Гаусса, k=1,2,…,n, причем $A^{(0)}=A$.

Алгоритм 2. $U\overline{L}$ -разложение матрицы A (на основе гауссова исключения по столбцам):

Процесс гауссова исключения ведется от последнего неизвестного x_n и от последнего уравнения системы, двигаясь снизу вверх по нумерации уравнений и справа налево по нумерации исключаемых неизвестных. Нормировка происходит по строкам.

Алгоритм 3. $\overline{L}U$ -разложение матрицы A (на основе гауссова исключения по столбцам):

Для k=1 до n

Выбираем главный элемент в $A^{(k-1)}$.

Нормируем первый столбец матрицы $A^{(k-1)}$.

Для i=k+1 до n

Вычитаем первую строку матрицы $A^{(k-1)}$, умноженную на $a_{ik}^{(k-1)}$, из i-й строки.

Здесь $A^{(k-1)}$ – активная подматрица матрицы A после (k–1)-го шага метода Гаусса, k=1,2,…,n, причем $A^{(0)}=A$.

Алгоритм 3 отличается от алгоритма 1 только нормировкой элементов активной подматрицы.

Алгоритм 4. $\overline{U}L$ -разложение матрицы A (на основе гауссова исключения по столбцам):

Действия аналогичные, как и в алгоритме 3, но процесс исключения ведется от последнего неизвестного x_n и от последнего уравнения системы, двигаясь снизу вверх по нумерации уравнений и справа налево по нумерации исключаемых неизвестных. Нормировка происходит по столбцам.

Алгоритм 5. *$L\overline{U}$* ***-разложение матрицы A (на основе гауссова исключения по строкам):***

Для k=1 до n
Для i=1 до k-1
Вычитаем i-ю строку матрицы $A^{(k-1)}$, умноженную на $a_{ki}^{(k-1)}$, из k-й строки.
Выбираем главный элемент в $A^{(k-1)}$.
Нормируем k-ю строку матрицы $A^{(k-1)}$.
Здесь $A^{(k-1)}$ – активная подматрица матрицы A после (k–1)-го шага метода Гаусса, k=1,2,…,n, причем $A^{(0)}=A$.

Отличие гауссова исключения по строкам от гауссова исключения по столбцам сводится в алгоритме 5 к изменению порядка действий: сначала серия вычитаний, а затем нормировка.

Алгоритм 6. *$U\overline{L}$* *-разложение матрицы A (на основе гауссова исключения по строкам):*

Процесс гауссова исключения по строкам выполняется как в алгоритме 5, но от последнего неизвестного x_n и от последнего уравнения системы, двигаясь снизу вверх по нумерации уравнений и справа налево по нумерации исключаемых неизвестных.

Компактные схемы

Компактные схемы являются разновидностью метода Гаусса. Первая называется компактной схемой Краута, а вторая – компактной схемой «строка за строкой». В схеме Краута на каждом шаге исключения изменяются только первый столбец и первая строка активной подматрицы. В схеме «строка за строкой» на k-м шаге изменяется только k-ая строка матрицы A.

Алгоритм 7. *$L\overline{U}$* *-разложение по компактной схеме Краута:*

Для k=1 до n
Вычисляем k-й столбец матрицы L по формуле:

$$l_{ik} = a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}, \quad i \geq k.$$

Выбираем среди элементов k-го столбца главный элемент.

Вычисляем k-ю строку матрицы $\overline{U}$ по формуле:

$$u_{kj} = \left(a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj} \right) \Big/ l_{kk}, \quad j > k.$$

Алгоритм 8. $U\overline{L}$-разложение по компактной схеме Краута:

Вычисления столбца и строки выполняются как в алгоритме 7, но в обратном порядке, т. е. двигаясь снизу вверх по нумерации уравнений и справа налево по нумерации исключаемых неизвестных.

Алгоритм 9. $\overline{L}U$-разложение по компактной схеме Краута:

Для k=1 до n

Вычисляем k-ю строку матрицы U по формуле:

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}, \quad j \geq k.$$

Выбираем среди элементов k-ой строки главный элемент.

Вычисляем k-й столбец матрицы $\overline{L}$ по формуле:

$$l_{ik} = \left(a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk} \right) \Big/ u_{kk}, \quad i > k.$$

Алгоритм 10. $\overline{U}L$-разложение по компактной схеме Краута:

Вычисления столбца и строки выполняются как в алгоритме 9, но в обратном порядке, т. е. двигаясь снизу вверх по нумерации уравнений и справа налево по нумерации исключаемых неизвестных.

Алгоритм 11. $L\overline{U}$-разложение по компактной схеме «строка за строкой»:

Для k=1 до n

Вычисляем элемент k-й строки матрицы L по формуле:

$$l_{ki} = a_{ki} - \sum_{p=1}^{i-1} l_{kp} u_{pi}, \quad i \le k.$$

Вычисляем k-ю строку матрицы $\overline{U}$ по формуле:

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}, \quad j > k.$$

Выбираем среди элементов k-ой строки главный элемент.

Делим на главный элемент k-ю строку матрицы $\overline{U}$

Алгоритм 12. $U\overline{L}$ -разложение по компактной схеме «строка за строкой»:

Вычисления столбца и строки выполняются как в алгоритме 11, но в обратном порядке, т. е. двигаясь снизу вверх по нумерации уравнений и справа налево по нумерации исключаемых неизвестных.

Алгоритмы метода Жордана

К последней группе методов исключения относятся алгоритмы метода Жордана. Эти алгоритмы используют те же самые формулы, что и обычный метод Гаусса, но в отличие от него на k-м шаге метода Жордана пересчитывают все строки матрицы A, а не только строки, находящиеся ниже ведущей строки. Это означает полное исключение i-ой переменной из всех уравнений, кроме i-го. Таким образом, метод Жордана формально дает решение системы линейных алгебраических уравнений за один проход.

Алгоритм «$L\overline{U}^{-1}$-разложение» отыскивает $L\overline{U}$ -разложение матрицы A, но при его выполнении в одном и том же массиве дает вместо матрицы $\overline{U}$ обратную матрицу $\overline{U}^{-1}$.

Алгоритм 13. «$L\overline{U}^{-1}$-разложение» $A = L\overline{U}$ по методу Жордана:

Для k=1 до n

Выбираем главный элемент в $A^{(k-1)}$.

Нормируем первую строку матрицы $A^{(k-1)}$.

Для i=1 до k–1

Вычитаем первую строку матрицы $A^{(k-1)}$, умноженную на $a_{ik}^{(k-1)}$, из i-й строки.

Для i=k+1 до n
Вычитаем первую строку матрицы $A^{(k-1)}$, умноженную на $a_{ik}^{(k-1)}$, из i-й строки.
Для i=1 до n
Для j=i+1 до n
$a_{ij} = -a_{ij}$

Алгоритм 14. «$\bar{L}^{-1}U$ -разложение» $A = U\bar{L}$ по методу Жордана:

Алгоритм «$\bar{L}^{-1}U$ -разложение» находит $U\bar{L}$ -разложение матрицы A, но при его выполнении в одном и том же массиве дает вместо матрицы $\bar{L}$ обратную матрицу $\bar{L}^{-1}$. Действия здесь такие же, как в алгоритме 13, но выполняются в обратном порядке.

Выбор ведущего элемента

Для предотвращения появления нулевых элементов на диагонали матрицы в процессе гауссова исключения применяют выбор главного элемента.

Существуют три стратегии выбора главного (ведущего) элемента:

1) *Стратегия выбора главного элемента по столбцу* заключается в том, что на каждом шаге k работы алгоритма факторизации выбирается максимальный по модулю элемент в k-ом столбце среди элементов от k-го до n-го (для прямых алгоритмов) или среди элементов от 1-го до k-го (для обратных алгоритмов).

2) *Стратегия выбора главного элемента по строке* заключается в том, что на каждом шаге k работы алгоритма факторизации выбирается максимальный по модулю элемент в k-ой строке среди элементов от k-го до n-го (для прямых алгоритмов) или среди элементов от 1-го до k-го (для обратных алгоритмов).

3) *Стратегия выбора главного элемента по активной подматрице* заключается в том, что на каждом шаге k работы алгоритма факторизации выбирается максимальный по модулю элемент среди элементов, лежащих в столбцах от k-го до n-го и в строках от k-ой до n-ой (для прямых алгоритмов) или лежащих в столбцах от 1-го до k-го и в строках от 1-ой до k-ой (для обратных алгоритмов).

3.2 Задание на лабораторный проект

Для выполнения лабораторного проекта необходимо запрограммировать, отладить и протестировать следующие функции:

1. Система меню для взаимодействия пользователя с программой и генерация исходных данных задачи.
2. Функция факторизации матрицы, отвечающая Вашему варианту исключения.
3. Функция решения системы линейных алгебраических уравнений.
4. Функция вычисления определителя матрицы.
5. Функция обращения матрицы через решение системы $AX=E$.
6. Функция обращения матрицы через элементарные преобразования разложения.
7. Эксперимент 1 «Решение СЛАУ для случайных матриц».
8. Эксперимент 2 «Решение СЛАУ с плохо обусловленными матрицами».
9. Эксперимент 3 «Обращение случайных матриц».

Варианты заданий на лабораторный проект № 1

1. $L\overline{U}$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по столбцу.
2. $L\overline{U}$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по строке.
3. $L\overline{U}$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по активной подматрице.
4. $\overline{L}U$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по столбцу.
5. $\overline{L}U$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по строке.
6. $\overline{L}U$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по активной подматрице.
7. $L\overline{U}$ – разложение на основе гауссова исключения по строкам с выбором главного элемента по строке.
8. $L\overline{U}$ – разложение по компактной схеме Краута с выбором главного элемента по столбцу.

9. $\overline{L}U$ – разложение по компактной схеме Краута с выбором главного элемента по строке.
10. $L\overline{U}$ – разложение по компактной схеме «строка за строкой» с выбором главного элемента по строке.
11. $L\overline{U}^{-1}$ – разложение $A = L\overline{U}$ на основе жорданова исключения с выбором главного элемента по столбцу.
12. $L\overline{U}^{-1}$ – разложение $A = L\overline{U}$ на основе жорданова исключения с выбором главного элемента по строке.
13. $L\overline{U}^{-1}$ – разложение $A = L\overline{U}$ на основе жорданова исключения с выбором главного элемента по активной подматрице.
14. $\overline{U}L$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по столбцу.
15. $\overline{U}L$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по строке.
16. $\overline{U}L$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по активной подматрице.
17. $U\overline{L}$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по столбцу.
18. $U\overline{L}$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по строке.
19. $U\overline{L}$ – разложение на основе гауссова исключения по столбцам с выбором главного элемента по активной подматрице.
20. $U\overline{L}$ – разложение на основе гауссова исключения по строкам с выбором главного элемента по строке.
21. $U\overline{L}$ – разложение по компактной схеме Краута с выбором главного элемента по столбцу.
22. $\overline{U}L$ – разложение по компактной схеме Краута с выбором главного элемента по строке.
23. $U\overline{L}$ – разложение по компактной схеме «строка за строкой» с выбором главного элемента по строке.
24. $\overline{L}^{-1}U$ – разложение $A = U\overline{L}$ на основе жорданова исключения с выбором главного элемента по столбцу.
25. $\overline{L}^{-1}U$ – разложение $A = U\overline{L}$ на основе жорданова исключения с выбором главного элемента по строке.

26. $\bar{L}^{-1}U$ – разложение $A=U\bar{L}$ на основе жорданова исключения с выбором главного элемента по активной подматрице.

Обозначения:

L – нижняя треугольная матрица;

U – верхняя треугольная матрица;

$\bar{L}$ – нижняя треугольная матрица с единичными элементами на диагонали;

D – диагональная матрица;

$\bar{U}$ – верхняя треугольная матрица с единичными элементами на диагонали.

3.3 Методические рекомендации и примеры

1. Система меню для взаимодействия пользователя с программой

Возможны различные варианты меню:

Меню, в котором взаимодействие с пользователем осуществляется через командную строку MATLAB.

Пример:

```
clc;
while(1)
   param = input('Ввод данных - a;\n  Обработка данных - 2;\n
      Вывод данных - 3\n  Выход - 4\n', 's');
   switch param
      case '1', input_func();
      case '2', process_func();
      case '3', output_func();
      case '4', return;
      otherwise, disp('Введен некорректный символ.
Введите один из предложенных.');
   end
end
```

Меню, в котором взаимодействие с пользователем осуществляется через графический интерфейс GUI.

Меню должно включать следующие возможности:

– Ввод с клавиатуры размера задачи n, т. е. количество уравнений в СЛАУ.

– Ввод с клавиатуры элементов квадратной матрицы A размера $n \times n$.

– Ввод с клавиатуры элементов вектора b размера $n \times 1$ – правой части СЛАУ.

– Заполнение матрицы A и вектора b случайными числами. Для заполнения матрицы A использовать случайные числа из диапазона от -100 до 100.

Выделение памяти под матрицу размера $n \times n$:

```
A=zeros(n);
```

Освобождение памяти:

```
clear A;
```

Пример заполнения матрицы размера $n \times n$ нецелыми случайными числами в диапазоне от -100 до 100:

```
a = -100;
b = 100;
n = 3;
for i=1:n
        for j=1:n
                A(i,j) = a + (b-a).*rand(1);
        end
end
```

Пример заполнения матрицы размера $n \times n$ целыми случайными числами в диапазоне от -100 до 100:

```
A=randi([-100,100],n,n);
```

2. Функция факторизации матрицы

Рассмотрим на примере алгоритма 1 реализацию $L\overline{U}$ -разложения.

Рассмотрим реализацию каждого действия отдельно:

1. *Нормируем первую строку матрицы* $A^{(k-1)}$.

```
        for j=k:n
                A(k,j)=A(k,j)/A(k,k);
        end
```

2. *Для i=k+1 до n*

Вычитаем первую строку матрицы $A^{(k-1)}$, *умноженную на* $a_{ik}^{(k-1)}$, *из i-ой строки.*

```
for i=k+1:n
    for j=k:n
        A(i,j)=A(i,j)-A(i,k)*A(k,j);
    end
end
```

Таким образом, программируя отдельные действия с помощью циклов, составляем код функции факторизации матрицы *A* по заданному алгоритму.

Стратегии выбора главного элемента

Стратегия выбора главного элемента по активной подматрице включает в себя стратегию выбора по столбцу и по строке. Поэтому будем рассматривать пример ее реализации.

Для максимального элемента нужно запомнить его индексы, например, imax и jmax. После этого элемент матрицы с индексами imax и jmax меняется местами с главным (диагональным) элементом с индексами (k,k). Для того чтобы не нарушить связи в матрице линейной системы, необходимо при обмене полностью менять местами строки и столбцы. Эффективным способом обмена является введение так называемых *векторов перестановок строк и столбцов*, которые содержат номера строк и столбцов матрицы *A*. Тогда для обмена достаточно поменять всего лишь два элемента в массиве перестановок.

При использовании массива перестановок всюду в программе обращаемся к элементам матрицы следующим образом: для первой стратегии A(p(i),j), для второй стратегии A(i,q(j)), для третьей стратегии A(p(i),q(j)), где p и q – вектора перестановок строк и столбцов, соответственно.

Рассмотрим реализацию третьей стратегии как более общей:

```
% Массивы перестановок: p – массив номеров строк, q – массив номеров столбцов
p = zeros(n,1);
q = zeros(n,1);
%вспомогательные переменные
max = 0;
```

```
imax = 0;
jmax = 0;
buf = 0;
% Заполняем массивы перестановок начальными значениями перед началом факторизации
for i=1:n
   p(i)=i;
   q(i)=i;
end
% n – размер задачи, k – номер текущего шага алгоритма
max=abs(A(p(k),q(k)));
imax=k;
jmax=k;
for i=k:n
   for j=k:n
      if(abs(A(p(i),q(j)))>max)
         max=abs(A(p(i),q(j)));
         imax=i;
         jmax=j;
      end
   end
end
% Обмен
if(imax~=k)
         buf=p(k);
         p(k)=p(imax);
         p(imax)=buf;
end
if(jmax~=k)
         buf=q(k);
         q(k)=q(jmax);
         q(jmax)=buf;
end
```

3. Решение системы линейных алгебраических уравнений

Рассмотрим детали реализации алгоритма нахождения решения СЛАУ по $L\overline{U}$ -разложению. Сначала рассмотрим нахождение решения без учета выбора главного элемента. Решение будем искать в два этапа: сначала находим решение для системы с нижней треугольной матрицей, затем находим решение с верхней треугольной матрицей. Приведем математическое обоснование:

$$Ax = b \Rightarrow L\overline{U}x = b \Rightarrow (\overline{U}x = y) \Rightarrow Ly = b \Rightarrow y \Rightarrow \overline{U}x = y \Rightarrow x\,.$$

Первый проход: известны матрица L (нижняя треугольная часть факторизованной матрицы A) и вектор b, находим неизвестный вектор y с помощью прямой подстановки:

```
% метод прямой подстановки (скалярных произведений)
    for i=1:n
        sum=0;
        for j=1:i
            sum=sum+A(i,j)*y(j);
        end
        y(i)=(b(i)-sum)/A(i,i);
    end
```

Второй проход: известны матрица $\overline{U}$ (верхняя треугольная часть факторизованной матрицы A) и вектор y, находим неизвестный вектор x с помощью обратной подстановки:

```
%метод обратной подстановки (скалярных произведений)
    for i=n:-1:1
        sum=0;
        for j=i+1:n
            sum=sum+A(i,j)*x(j);
        end
        x(i)=y(i)-sum;
    end
% Найденное решение находится в массиве x
```

Можно использовать и другие способы программной реализации методов прямой и обратной подстановки (см. базовый учебник, с. 67-69).

Аналогичным образом реализуются алгоритмы нахождения решения и для других вариантов факторизации. Теперь будем учитывать, что при факторизации была использована стратегия выбора главного элемента по активной подматрице. В этом случае при перестановке строк в СЛАУ необходимо переставлять соответствующие элементы в векторе b (чтобы не потерять связи между неизвестными и уравнениями в линейной системе). В программе нужно обращаться к массиву b также через вектор перестановок строк, т. е. b(p(i)).

Если в СЛАУ переставляют столбцы матрицы коэффициентов, то таким же образом должны быть переставлены и элементы вектора x.

Тогда один из вариантов программной реализации алгоритма нахождения СЛАУ при выборе главного элемента по активной подматрице можно записать так:

Первый проход: известны матрица L и вектор b, находим неизвестный вектор y с помощью прямой подстановки, учитываем перестановки в элементах матрицы и вектора:

```
% метод прямой подстановки (скалярных произведений)
    for i=1:n
        sum=0;
        for j=1:i
            sum=sum+A(p(i),q(j))*y(j);
        end
        y(i)=(b(p(i))-sum)/A(p(i),q(i));
    end
```

Второй проход: известны матрица $\overline{U}$ и вектор y, находим неизвестный вектор x с помощью обратной подстановки, учитываем перестановки в элементах матрицы и вектора:

```
%метод обратной подстановки (скалярных произведений)
    for i=n:-1:1
        sum=0;
        for j=i+1:n
            sum=sum+A(p(i),q(j))*x(q(j));
        end
        x(q(i))=y(i)-sum;
    end
%Найденное решение находится в массиве x
```

4. Определитель матрицы

Если разложение уже известно, определитель находится просто как произведение диагональных элементов в факторизованной матрице, так как

$$\det A = \det L\overline{U} = \det L \cdot \det \overline{U} = \prod_{i=1}^{n} l_{ii} \cdot 1 = \prod_{i=1}^{n} l_{ii} = \prod_{i=1}^{n} a_{ii}$$

(определитель верхней треугольной матрицы с единичной диагональю равен единице, а определитель нижней треугольной матрицы равен произведению диагональных элементов.)

Если мы используем любую из трех стратегий выбора главного элемента, тогда при работе алгоритма факторизации происходит обмен строк и/или столбцов. При этом каждый раз при обмене знак определителя меняется на противоположный. При написании программы можно объявить глобальный флаг znak, в котором будет сохраняться значение -1 или 1, в зависимости от того, четное или нечетное количество перестановок было сделано на данный момент. Изменять флаг следует в той части программы, где происходит обмен значений в векторах перестановок. Затем при вычислении определителя произведение диагональных элементов нужно домножить на флаг.

```
%перед началом алгоритма факторизации
znak=1;
%при обмене в векторах перестановок
if(imax~=k)
        znak=-znak;
        buf=p(k);
        p(k)=p(imax);
        p(imax)=buf;
end
if(jmax~=k)
        znak=-znak;
        buf=q(k);
        q(k)=q(jmax);
        q(jmax)=buf;
end
%теперь вычисляем определитель
```

```
det=1;
for i=1:n
        det=det*A(p(i),q(i));
end
det=det*znak;
```

5. Обращение матрицы через решение системы AX=E

Предположим, что мы уже нашли разложение $L\overline{U}$ и в данный момент в массиве A находится факторизованная матрица A. Более того, у нас уже реализован алгоритм нахождения решения СЛАУ.

Будем искать обратную матрицу из следующих соображений.

Обозначим обратную матрицу через X. Тогда верно тождество $AX=E$, где E – единичная матрица. Разобьем матрицы X и E на столбцы и запишем систему СЛАУ

$$\begin{cases} Ax_1 = e_1 \\ Ax_2 = e_2 \\ \dots \\ Ax_n = e_n \end{cases},$$

где x_k – k-ый столбец матрицы X, e_k – k-ый столбец матрицы E.

Каждую СЛАУ решаем уже имеющимся алгоритмом, т. е. по очереди находим столбцы обратной матрицы и собираем их вместе. Таким образом, находим обратную матрицу.

Для программной реализации необходим дополнительный массив A1, в который будем записывать найденные столбцы обратной матрицы.

Далее в цикле заполняем массив b элементами очередного столбца единичной матрицы, решаем СЛАУ и получаем в массиве x очередной столбец обратной матрицы, затем записываем его в матрицу A1.

6. Обращение матрицы через элементарные преобразования

Предположим, что мы уже нашли разложение $L\overline{U}$ и в данный момент в массиве A находится преобразованная матрица A. Необходимо на месте разложения в массиве A вычислить обратную матрицу A1.

Пример вычисления обратной матрицы данным способом подробно рассмотрен в базовом учебнике на с. 41-46. Смысл алгоритма заключается в следующем (пока не учитываем стратегию выбора главного элемента):

Пусть $A = L\overline{U}$. Тогда $A^{-1} = \overline{U}^{-1}L^{-1}$. Алгоритм вычисления обратной матрицы можно разделить на четыре этапа:

1) Выполняем подготовку элементов матрицы для вычисления обратной. Для этого вычисляем обратные к элементарным матрицам следующим образом: в матрице $\overline{U}$ меняем знаки на противоположные у всех наддиагональных элементов; в матрице L каждый диагональный элемент заменяем на обратный по величине, затем полученное число берем с противоположным знаком и умножаем на каждый поддиагональный элемент.

2) Вычисляем матрицу, обратную к $\overline{U}$, в верхней треугольной части массива A (т. к. у данной матрицы единичная диагональ, то у обратной матрицы также будет единичная диагональ, и ее можно не хранить в массиве).

3) Вычисляем матрицу, обратную к L, в нижней треугольной части (вместе с диагональю) массива A.

4) Перемножаем в одном массиве матрицы $\overline{U}^{-1}$ и L^{-1}. Получаем искомую матрицу A1. При вычислении нужно правильно задать границы изменения индексов для верхней треугольной и нижней треугольной частей массивов.

Рассмотрим *один из возможных вариантов* программной реализации пунктов 2), 3), 4).

Программная реализация пункта 1) является тривиальной.

2) Программная реализация вычисления матрицы $\overline{U}^{-1}$ в верхней треугольной части массива A (т. е. вместо матрицы $\overline{U}$ надо получить матрицу $\overline{U}^{-1}$, не используя дополнительные массивы).

```
for k=n:(-1):2
        for i=1:(k-2)
                for j=k:n
                        A(i,j)=A(i,j)+A(i,k-1)*A(k-1,j);
                end
        end
end
```

3) Программная реализация вычисления матрицы L^{-1} в нижней треугольной вместе с диагональю части массива A (т. е. вместо матрицы L надо получить матрицу L^{-1}, не используя дополнительные массивы).

```
for k=1:(n-1)
    for i=(k+2):n
        for j=1:k
            A(i,j)=A(i,j)+A(i,k+1)*A(k+1,j);
            for i=j:k
                A(k+1,j)=A(k+1,j)*A(k+1,k+1);
            end
        end
    end
end
```

4) Перемножение двух треугольных матриц $\overline{U}^{-1}$ и L^{-1} в одном массиве:

```
for i=1:n
    for j=1:n
        if(i<j)
            sum=0;
            for k=j:n
                sum=sum+A(i,k)*A(k,j);
            end
        end
        if(i>=j)
            sum=A(i,j);
            for k=i+1:n
                sum=sum+A(i,k)*A(k,j);
            end
        end
        A(i,j)=sum;
    end
end
```

Для того чтобы учесть стратегию выбора главного элемента, необходимо в программном коде обращаться к элементам матрицы *A* как A(p(i),q(j)). Поскольку при факторизации исходной матрицы *A* использовались перестановки строк и столбцов, в результате вычислений мы получим матрицу, обратную к матрице *PAQ*, где *P* и *Q* – матрицы перестановок строк и столбцов.

Следовательно, $PAQ = L\overline{U} \Rightarrow \overline{U}^{-1}L^{-1} = Q^{-1}A^{-1}P^{-1} \Rightarrow A^{-1} = Q\overline{U}^{-1}L^{-1}P$.

Тогда (*i*,*j*)-му элементу обратной матрицы A^{-1} соответствует элемент массива A(p(q1(i)),q(p1(j))), где *p*1, *q*1 – вектора обратных перестановок, элементы которых вычисляются как p1(p(i))=i и q1(q(j))=j.

Программная реализация остальных вариантов разложения может быть построена с помощью модификации приведенного программного кода.

Тестовые задачи для отладки программного кода можно найти в базовом учебнике на с. 160-177.

7. Эксперимент 1 «Решение СЛАУ для случайных матриц»

Требуется написать функцию для проведения первого эксперимента с целью исследования скорости и погрешности решения СЛАУ. Подробное описание эксперимента можно найти в базовом учебнике на с. 49.

Перед реализацией эксперимента необходимо написать процедуру, вычисляющую погрешность решения СЛАУ.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок	Время	Погрешность	Теоретическое число операций	Реальное число операций

Каждая строка в таблице будет зависеть от *n* – количества уравнений в системе. Число *n* должно изменяться от 5 до 100 через 5 порядков, т. е. в таблице будет 20 строк.

2) Присвоить *n* очередное значение.

3) Заполнить матрицу *A* случайными числами и сгенерировать вектор *b* в соответствии с **заданием 1**.

4) Выполнить факторизацию матрицы A в соответствии с **заданием 2**.

5) Найти решение СЛАУ x в соответствии с **заданием 3**.

6) Подсчитать теоретическое число операций умножения и деления по формуле $n^3/3$.

7) Подсчитать реальное число операций умножения и деления с помощью специальных счетчиков, которые добавляются в функции факторизации и решения.

8) Подсчитать скорость решения задачи как сумму времени, затраченного на разложение матрицы, и времени решения СЛАУ.

Сделать это можно с помощью функций `tic` и `toc`:

```
tic
<операция 1>
...
<операция n>
time = toc; %затраченное время
```

или воспользоваться другими стандартными функциями работы со временем, например, `clock` или `etime`.

9) Оценить погрешность решения СЛАУ:
   - Задать точное решение x^*. В качестве точного решения нужно взять вектор $x^*=(1, 2, \ldots, n)^T$, где n – размер очередной матрицы.
   - Образовать правые части $b=Ax^*$, где матрица A известна из п. 3.
   - Найти решение СЛАУ $Ax=b$.
   - Вычислить погрешность решения СЛАУ как максимальный модуль разности компонент вектора точного решения и вектора найденного решения.

10) Добавить в таблицу и файл для хранения экспериментальных данных полученные экспериментальные данные: порядок n, время выполнения, погрешность решения СЛАУ, теоретическое и реальное число операций.

    Для записи данных в файл можно воспользоваться функцией `dlmwrite('file.dat', A,';')`, где `'file.dat'` – имя файла; `A` – массив данных для записи; `';'` – разделитель.

11) Пункты 2-8 повторить для всех значений n.

12) Построить следующие графики решения СЛАУ:

- зависимость реального и оценочного числа операций от размера матрицы (для разных графиков использовать разные цвета);
- зависимость времени решения от размера матриц;
- зависимость погрешности решения от размера матриц.
- При построении графиков необходимо использовать данные из файла с экспериментальными данными.

  Для чтения экспериментальных данных из файла file.dat воспользуемся функцией dlmread('file.dat', ';'), где 'file.dat' – имя файла; ';' – разделитель.

  Для построения графиков воспользуемся функцией plot:

  plot(n, numb, 'g'); %выводит график функции numb(n)- зависимость числа операций от размера матрицы

  третий параметр функции устанавливает цвет и стиль линий и маркеров. В данном случае 'g' (green) задает зеленый цвет графика.

13) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

8. Эксперимент 2 «Решение СЛАУ с плохо обусловленными матрицами»

Написать реализацию второго эксперимента для исследования скорости и погрешности решения СЛАУ в случае, когда матрица *A* является плохо обусловленной. Подробное описание эксперимента можно найти в базовом учебнике на с. 49. Перед проведением эксперимента написать функцию, вычисляющую погрешность решения СЛАУ (см. эксперимент 1).

Методика проведения эксперимента:

Для каждой из 10 плохо обусловленных матриц на (см. ниже с. 37-38) выполнить следующее:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок	Время	Погрешность	Теоретическое число операций	Реальное число операций

Каждая строка в таблице будет зависеть от *n* – количества уравнений в системе. Если матрица *A* имеет фиксированный размер (матрицы с номе-

рами 2, 3, 6, 10), тогда n равно размеру матрицы, и в таблице будет только одна строка. Если порядок матрицы не фиксирован (матрицы с номерами 1, 4, 5, 7, 8, 9), тогда число n должно изменяться от 4 до 40 через 4 порядка, т. е. в таблице будет 10 строк.

2) Присвоить n очередное значение.
3) Заполнить матрицу A в соответствии с ее номером и сгенерировать вектор b в соответствии с **заданием 1** (для заданной матрицы!).
4) Выполнить факторизацию матрицы A в соответствии с **заданием 2**.
5) Найти решение СЛАУ x в соответствии с **заданием 3**.
6) Подсчитать теоретическое число операций умножения и деления по формуле $n^3/3$.
7) Подсчитать реальное число операций умножения и деления с помощью специальных счетчиков, которые добавляются в функции факторизации и решения.
8) Подсчитать скорость решения задачи как сумму времени, затраченного на разложение матрицы, и времени решения СЛАУ (см. эксперимент 1, п. 8).
9) Оценить погрешность решения СЛАУ так же, как в эксперименте 1.
10) Добавить в таблицу и файл для хранения экспериментальных данных полученные экспериментальные данные (см. эксперимент 1, п. 10): порядок n, время выполнения, погрешность решения СЛАУ, теоретическое и реальное число операций.
11) Пункты 2-8 повторить для всех значений n.
12) Построить следующие графики решения СЛАУ:
    - зависимость реального и оценочного числа операций от порядка матрицы (для разных графиков использовать разные цвета);
    - зависимость времени решения от порядка матриц;
    - зависимость погрешности решения от порядка матриц.

Для построения графиков необходимо использовать данные из текстового файла (см. эксперимент 1, п. 12).

13) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

Список плохо обусловленных матриц для эксперимента 2

1. Матрица Гильберта:

$a_{ij} = 1/(i + j - 1)$

2. Матрица A с элементами:

$a_{ii} = 1$ для $i = 1, 2, ..., 20$;
$a_{ii+1} = 1$ для $i = 1, 2, ..., 19$;
$a_{ij} = 0$ для остальных значений i и j.

3. $A = \begin{bmatrix} 5 & 4 & 7 & 5 & 6 & 7 & 5 \\ 4 & 12 & 8 & 7 & 8 & 8 & 6 \\ 7 & 8 & 10 & 9 & 8 & 7 & 7 \\ 5 & 7 & 9 & 11 & 9 & 7 & 5 \\ 6 & 8 & 8 & 9 & 10 & 8 & 9 \\ 7 & 8 & 7 & 7 & 8 & 10 & 10 \\ 5 & 6 & 7 & 5 & 9 & 10 & 10 \end{bmatrix}$

4. Матрица A с элементами:

$a_{ii} = 0.01/(n - i + 1)/(i + 1)$;
$a_{ij} = 0$ для $i < j$;
$a_{ij} = i\,(n - j)$ для $i > j$.

5. Матрица A с элементами:

$a_{ii} = 0.01/(n - i + 1)/(i + 1)$;
$a_{ij} = j\,(n - i)$ для $i < j$;
$a_{ij} = i\,(n - j)$ для $i > j$.

6. $A = \begin{bmatrix} R & S & T & T \\ S & R & S & T \\ T & S & R & S \\ T & T & S & R \end{bmatrix}$, $R = \begin{bmatrix} ctg\theta & \cos ec\theta \\ -\cos ec\theta & ctg\theta \end{bmatrix}$,

$S = \begin{bmatrix} 1 - ctg\theta & \cos ec\theta \\ -\cos ec\theta & 1 + ctg\theta \end{bmatrix}$, $T = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

Вычисления проводить при θ близких к нулю или π.

7. Матрица с параметром α:

$a_{ii} = \alpha^{|n-2i|/2}$;

$a_{1j} = a_{j1} = a_{11}/\alpha^{j}$;

$a_{nj} = a_{jn} = a_{nn}/\alpha^{j}$;

$a_{ij} = 0$ для остальных значений i и j.

8. $a_{ij} = e^{i \cdot j \cdot h}$.

Вычисления проводить при h близких к нулю или 1000.

9. $a_{ij} = c + \log_2(i\,j)$.

Вычисления проводить при больших c.

10.
$$A = \begin{bmatrix} 0.9143 \cdot 10^{-4} & 0 & 0 & 0 \\ 0.8762 & 0.7156 \cdot 10^{-4} & 0 & 0 \\ 0.7943 & 0.8143 & 0.9504 \cdot 10^{-4} & 0 \\ 0.8017 & 0.6123 & 0.7165 & 0.7123 \cdot 10^{-4} \end{bmatrix}$$

9. Эксперимент 3 «Обращение случайных матриц»

Реализовать третий эксперимент для исследования скорости и погрешности алгоритмов обращения матриц. Подробное описание эксперимента можно найти в базовом учебнике на с. 49-50.

Перед реализацией эксперимента написать процедуру, вычисляющую погрешность найденной обратной матрицы. Формулы (2.19), (2.20) на с. 49, 50 базового учебника позволяют вычислить погрешность обращения матрицы A. Для этого необходимо, в соответствии с формулами, сначала умножить исходную матрицу на найденную обратную и вычесть результат из единичной матрицы, затем найти норму найденной разности и поделить ее на норму исходной матрицы. Норма матрицы вычисляется по формуле (2.20) из базового учебника – это максимальная сумма модулей элементов строк.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок	Время I	Время II	Погрешность I	Погрешность II	Реальное число операций I	Реальное число операций II	Теоретическое число операций

Каждая строка в таблице будет зависеть от n – размера матрицы. Число n должно изменяться от 5 до 100 через 5 порядков, т. е. в таблице будет 20 строк.

2) Присвоить n очередное значение.
3) Заполнить матрицу A случайными числами в соответствии с **заданием 1**.
4) Сохранив предварительно копию исходной матрицы A, выполнить ее факторизацию в соответствии с **заданием 2**.
5) Найти обратную матрицу первым способом в соответствии с **заданием 5**.
6) Подсчитать скорость обращения матрицы (см. эксперимент 1) первым способом.
7) Найти погрешность вычисления обратной матрицы.
8) Найти обратную матрицу вторым способом в соответствии с **заданием 6**.
9) Подсчитать скорость обращения матрицы (см. эксперимент 1) вторым способом.
10) Найти погрешность вычисления обратной матрицы.
11) Подсчитать теоретическое число операций умножения и деления по формуле n^3.
12) Подсчитать реальное число операций умножения и деления с помощью специальных счетчиков, которые добавляются в функции факторизации и обращения первым и вторым способами.
13) Добавить в таблицу полученные экспериментальные данные: порядок n, время и погрешность обращения первым и вторым способами, теоретическое и реальное число операций при обращении первым и вторым способами.
14) Пункты 2-8 повторить для всех значений n.

15) Построить следующие графики для обращения матриц:
11. зависимость реального и оценочного числа операций от размера матрицы (для разных графиков использовать разные цвета);
12. зависимость времени обращения первым и вторым способом от размера матрицы;
13. зависимость погрешности обращения первым и вторым способом от размера матрицы (см. эксперимент 1).
16) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

4 Лабораторный проект № 2 «Разложения Холесского»

Перед выполнением лабораторного проекта рекомендуем подробно ознакомиться с материалом базового учебника, с. 90-106.

4.1 Теория

Разложением Холесского принято называть любое из следующих представлений положительно определенной матрицы P: $P = LL^T$, $P = \overline{L}D\overline{L}^T$, $P = UU^T$, $P = \overline{U}D\overline{U}^T$, где L – нижняя треугольная матрица с положительными элементами на диагонали, U – верхняя треугольная матрица с положительными элементами на диагонали, $\overline{L}$ – нижняя треугольная матрица с единичными элементами на диагонали, D – диагональная матрица с положительными элементами на диагонали, $\overline{U}$ – верхняя треугольная матрица с единичными элементами на диагонали.

Два варианта разложения Холесского $\boldsymbol{LL^T}$ *и* $\overline{L}D\overline{L}^T$ *с немедленными модификациями* (в каждом алгоритме объединены оба разложения, изменения, которые относятся к LL^T-разложению, заключены в скобки):

1) kij-алгоритм	***2) kji-алгоритм***
$(l_{11} = p_{11}^{1/2})$	$(l_{11} = p_{11}^{1/2})$
Для k=1 *до* n–1	*Для* k=1 *до* n–1
Для i=k+1 *до* n	*Для* s=k+1 *до* n
$l_{ik} = p_{ik}/p_{kk}$	$l_{sk} = p_{sk}/p_{kk}$ (p_{sk}/l_{kk})
$(l_{ik} = p_{ik}/l_{kk})$	*Для* j=k+1 *до* n
Для j=k+1 *до* i	*Для* i=j *до* n
$p_{ij} = p_{ij} - l_{ik}p_{jk}$	$p_{ij} = p_{ij} - l_{ik}p_{jk}$
$p_{ij} = p_{ij} - l_{ik}l_{jk}$	$(p_{ij} = p_{ij} - l_{ik}l_{jk})$
$(l_{k+1,k+1}=p_{k+1,k+1}^{1/2})$	$(l_{k+1,k+1} = p_{k+1,k+1}^{1/2})$

Четыре варианта разложения Холесского $\boldsymbol{LL^T}$ *и* $\overline{L}D\overline{L}^T$ *с отложенными модификациями* (в каждом алгоритме объединены оба разложения, изменения, которые относятся к LL^T-разложению, заключены в скобки):

3) jki-алгоритм

$(l_{11} = p_{11}^{1/2})$
Для j=2 *до* n
Для s=j *до* n
$l_{s,j-1} = p_{s,j-1}/p_{j-1,j-1}$
$(l_{s,j-1} = p_{s,j-1}/l_{j-1,j-1})$
Для k=1 *до* j–1
Для i=j *до* n
$p_{ij} = p_{ij} - l_{ik}p_{jk}$

4) jik-алгоритм

$(l_{11} = p_{11}^{1/2})$
Для j=2 *до* n
Для s=j *до* n
$l_{s,j-1} = p_{s,j-1}/p_{j-1,j-1}$
$(l_{s,j-1} = p_{s,j-1}/l_{j-1,j-1})$
Для i=j *до* n
Для k=1 *до* j–1
$p_{ij} = p_{ij} - l_{ik}p_{jk}$
$(p_{ij} = p_{ij} - l_{ik}l_{jk})$
$(l_{jj} = p_{jj}^{1/2})$

5) ikj-алгоритм

$(l_{11} = p_{11}^{1/2})$
Для i=2 *до* n
Для k=1 *до* i–1
$l_{ik} = p_{ik}/p_{kk}$
$(l_{ik} = p_{ik}/l_{kk})$
Для j=k+1 *до* i
$p_{ij} = p_{ij} - l_{ik}p_{jk}$
$(p_{ij} = p_{ij} - l_{ik}l_{jk})$
$(l_{ii} = p_{ii}^{1/2})$

6) ijk-алгоритм

$(l_{11} = p_{11}^{1/2})$
Для i=2 *до* n
Для j=2 *до* i
$l_{i,j-1} = p_{i,j-1}/p_{j-1,j-1}$
$(l_{i,j-1} = p_{i,j-1}/l_{j-1,j-1})$
Для k=1 *до* j–1
$p_{ij} = p_{ij} - l_{ik}p_{jk}$
$(p_{ij} = p_{ij} - l_{ik}l_{jk})$
$(l_{ii} = p_{ii}^{1/2})$

Приведенные алгоритмы LL^T и $\overline{L}D\overline{L}^T$-разложений Холесского получены из соответствующих *ijk*-алгоритмов $\overline{L}U$-разложения матрицы A (см. подраздел 3.5 базового учебника). Для получения UU^T и $\overline{U}D\overline{U}^T$-разложений Холесского матрицы A удобно исходить из $\overline{U}L$-разложения матрицы A, если для него предварительно построить *ijk*-алгоритмы. Необходимо учесть, что $\overline{U}L$-разложение соответствует обратному порядку исключения переменных и модификация системы уравнений начинается с последней переменной последнего уравнения.

Разложение Холесского: алгоритмы окаймления

Для разложения Холесского в любых его вариантах существует еще один класс алгоритмов – так называемые матрично-векторные алгоритмы,

объединенные идеей окаймления. Построение этих алгоритмов базируется на блочном перемножении матриц, участвующих в разложении.

Покажем, каким образом получаются такие матрично-векторные алгоритмы на примере LL^T-разложения Холесского. Аналогичным образом можно самостоятельно построить родственные алгоритмы для других трех вариантов разложения.

Поделим все матрицы на блоки, выделяя в каждой матрице j-ую строку и j-ый столбец. Тем самым разложение $P = LL^T$ будет представлено в блочной форме:

$$\begin{matrix} & j & & j & & j \\ & \Downarrow & & \Downarrow & & \Downarrow \end{matrix}$$

$$j \Rightarrow \begin{pmatrix} P_{11} & a & P_{13} \\ a^T & p_{jj} & b^T \\ P_{31} & b & P_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ c^T & l_{jj} & \\ L_{31} & d & L_{33} \end{pmatrix} \cdot \begin{pmatrix} L_{11}^T & c & L_{31}^T \\ & l_{jj} & d^T \\ & & L_{33}^T \end{pmatrix}$$

Фрагменты j-ой строки и j-го столбца обозначены как векторы-столбцы выделенными символами $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ и $\boldsymbol{d}$, а заглавные буквы обозначают матрицы. Нулевые элементы треугольных матриц не показаны.

Перемножение матриц, выполняемое поблочно, дает девять соотношений относительно блочных элементов матриц P и L. Пользуясь этим, рассмотрим два основных способа разложения матрицы P методом окаймления.

Окаймление известной части разложения

Из указанных девяти соотношений возьмем только те, которые окаймляют блок $P_{11} = L_{11}L_{11}^T$, считая, что в этой части разложение уже сделано, т. е. блок L_{11} уже вычислен.

В силу симметрии P из трех окаймляющих произведений имеем только два:

$$\boldsymbol{a} = L_{11}\boldsymbol{c} \text{ и } p_{jj} = \boldsymbol{c}^T\boldsymbol{c} + l_{jj}^2$$

Отсюда сначала находим $\boldsymbol{c}$ как решение нижнетреугольной системы уравнений $L_{11}\boldsymbol{c} = \boldsymbol{a}$; затем находим $l_{jj} = (p_{jj} - \boldsymbol{c}^T\boldsymbol{c})^{1/2}$.

Строчный алгоритм окаймления известной части LL^T-разложения:

$l_{11} = \sqrt{p_{11}}$
Для j=2 до n
    Для k=1 до j–1
        $l_{jk} = p_{jk} / l_{kk}$
        Для i=k+1 до j
            $p_{ji} = p_{ji} - l_{jk} l_{ik}$
$l_{jj} = \sqrt{p_{jj}}$

Алгоритм скалярных произведений окаймления известной части LL^T-разложения:

$l_{11} = \sqrt{p_{11}}$
Для j=2 до n
    Для i=2 до j
        $l_{j,i-1} = p_{j,i-1} / l_{i-1,i-1}$
        Для k=1 до i–1
            $p_{ji} = p_{ji} - l_{jk} l_{ik}$
$l_{jj} = \sqrt{p_{jj}}$

Окаймление неизвестной части разложения

Из указанных девяти соотношений возьмем те, которые окаймляют блок P_{33}, считая, что до этого блока разложение уже сделано, т. е. что блоки L_{11}, L_{31} и $\boldsymbol{c}$ уже найдены. В силу симметрии P из трех окаймляющих произведений имеем только два: $p_{jj} = \boldsymbol{c}^T\boldsymbol{c} + l_{jj}^2$ и $\boldsymbol{b} = L_{31}\boldsymbol{c} + \boldsymbol{d}\, l_{jj}$.

Отсюда сначала находим $l_{jj} = (p_{jj} - \boldsymbol{c}^T\boldsymbol{c})^{1/2}$; затем $\boldsymbol{d} = (\boldsymbol{b} - L_{31}\boldsymbol{c})/l_{jj}$.

Алгоритм линейных комбинаций окаймления неизвестной части LL^T-разложения:

Для j=1 до n
  Для k=1 до j–1
    $p_{jj} = p_{jj} - l_{jk} l_{jk}$
  $l_{jj} = \sqrt{p_{jj}}$
  Для k=1 до j–1
    Для i=j+1 до n
      $p_{ij} = p_{ij} - l_{ik} l_{jk}$
  Для s=j+1 до n
    $l_{sj} = p_{sj} / l_{jj}$

Алгоритм скалярных произведений окаймления неизвестной части LL^T-разложения:

Для j=1 до n
  Для i=j+1 до n
    Для k=1 до j–1
      $p_{jj} = p_{jj} - l_{jk} l_{jk}$
    $l_{jj} = \sqrt{p_{jj}}$
    Для s=j+1 до n
      $l_{sj} = p_{sj} / l_{jj}$

4.2 Задание на лабораторный проект

1. Система меню для взаимодействия пользователя с программой.
2. Функция генерации ПО-матрицы *P*.
3. Функция разложения Холесского.
4. Функция решения системы линейных алгебраических уравнений.
5. Функция решения системы линейных алгебраических уравнений с ленточной матрицей *P*.
6. Эксперимент 1 «Количество арифметических операций».
7. Эксперимент 2 «Решение СЛАУ с заполненной матрицей *P*».
8. Эксперимент 3 «Решение СЛАУ с разреженной матрицей *P*».

Варианты заданий на лабораторный проект № 2

Вид разложения	ijk-формы						Окаймление			
							известной части		неизвестной части	
	kij	kji	jki	jik	ikj	ijk	a	b	c	b
$P = \bar{L}D\bar{L}^T$	1	2	3	4	5	6	7	8	9	10
$P = LL^T$	11	12	13	14	15	16	17	18	19	20
$P = \bar{U}D\bar{U}^T$	21	22	23	24	25	26	27	28	29	30
$P = UU^T$	31	32	33	34	35	36	37	36	39	40

где a – строчный алгоритм, b – алгоритм скалярных произведений, c – алгоритм линейных комбинаций.

С учетом положительной определенности матрицы *P*, процедура выбора главного элемента, а также процедуры перестановки строк и столбцов матрицы *P* отсутствуют как для заполненных, так и для разреженных матриц.

4.3 Методические рекомендации и примеры

1. Система меню для взаимодействия пользователя с программой

См. лабораторный проект № 1, задание 1.

Необходимо также реализовать сервисные подпрограммы: демонстрация разложения на экране, подпрограмма контроля правильности разложения.

2. Функция генерации ПО-матрицы P

Для заполнения матрицы *P* нужно задать случайные числа из диапазона от –100 до 100. Поскольку матрица *P* симметрическая, достаточно хранить только нижнюю (или верхнюю) треугольную часть этой матрицы вместе с диагональю.

Заполненная матрица P

Для строчного хранения заполненной симметрической матрицы размера *n* требуется одномерный массив размера $n(n+1)/2$. Хранить элементы можно как по строкам, так и по столбцам. Положение (i, j)-го элемента матрицы *P* в массиве можно определить по формуле $k=(i-1)i/2+j$.

Для генерации ПО-матрицы *P* необходимо:

1) Заполнить треугольную часть матрицы P, т. е. элементы p_{ij} , где $i > j$:
2) Заполнить диагональ. В качестве диагонального элемента p_{ii}, $1 \le i \le n$, нужно выбрать случайное число из интервала

$$\left[\sum_{j \neq i} |p_{ij}| + 1, \quad \sum_{j \neq i} |p_{ij}| + 101\right],$$

Рассмотрим один из возможных вариантов генерации заполненной матрицы P (для хранения по строкам):

%генерируем внедиагональные элементы матрицы (n - размерность) вычисляем сумму элементов матрицы для j~=i:

```
%вычисляем сумму элементов матрицы для j?i:
sum=zeros(n);
for j=1:(n-1)
   for i=(j+1):n
      A(i,j)=randi([-100,100],1,1);
      A(j,i)=A(i,j);
   end
end

for i=1:n
   for j=1:n
      if i~=j
        sum(i)=sum(i)+abs(A(i,j));
      end
   end
end
%выделяем память под массив для хранения матрицы P
P=zeros(1,(n*(n+1)/2));
ch=1;%счетчик
%Заполняем массив для хранения нижней треугольной части матрицы P по строкам:
for i=1:n
   for j=1:i
      if (i==j) %если элемент диагональный, то
            P(ch)=randi([sum(i)+1,sum(i)+101],1,1);
%выбираем случайное число из интервала
```

```
      else
         P(ch)=A(i,j);
      end
      ch=ch+1;
    end
end
end
P
clear A; %очищаем память
```

Если матрица P разреженная

Для хранения разреженной матрицы P потребуется два одномерных массива. Хранить элементы можно как по строкам, так и по столбцам. При строчном варианте хранения в массиве **a** хранятся построчно элементы матрицы от первого ненулевого до диагонального включительно. В массиве **b** на i-ом месте стоит положение i-го диагонального элемента матрицы P в массиве **a**.

Для случайного заполнения нижней или верхней треугольно части разреженной ленточной матрицы P необходимо использовать следующий алгоритм:

а) в случае заполнения нижней треугольной части матрицы P в i-ой строке, $i = 2, 3, \ldots, n$, случайным образом определить количество ненулевых элементов (от 1 до 10), их местоположение (номер столбца от $max\{1,i–50\}$ до $i–1$) и значение (ненулевые целые числа, лежащие в интервале от –100 до 100);

б) при заполнении верхней треугольной части матрицы P применять тот же алгоритм, что и в п. а), с той лишь разницей, что номер столбца лежит в интервале от $i+1$ до $min\{i+50,n\}$;

в) диагональный элемент в i-ой строке, $i = 1, 2, \ldots, n$, определить случайным образом на интервале

$$\left[\sum_{j\neq i}\left|p_{ij}\right|+1,\quad \sum_{j\neq i}\left|p_{ij}\right|+101\right].$$

Положение (i, j)-го элемента матрицы P в массиве **a** можно определить по следующему алгоритму:

1) Сначала вычисляем $k=b(i)–(i–j)$.
2) Затем, если $k > b(i–1)$, то этот элемент стоит на k-ом месте.
3) В противном случае (i, j)-ый элемент стоит левее первого ненулевого элемента i-ой строки, поэтому он равен нулю и в массиве **a** не хранится.

Способ хранения по столбцам строится аналогичным образом, но в этом случае следует хранить все элементы от диагонального до последнего ненулевого элемента столбца включительно.

3. Функция разложения Холесского

Подробное описание алгоритмов Холесского можно найти в базовом учебнике на с. 96-101.

В лабораторном проекте все вычисления должны проводиться в одном и том же массиве. Это значит, что элементы матрицы P должны замещаться элементами матрицы L, U или D (в зависимости от варианта разложения) по мере вычисления последних. Замещения могут происходить сразу, а могут откладываться до того момента, когда элементы матрицы P станут ненужными для дальнейших вычислений.

Рассмотрим реализацию на примере kij-алгоритма разложения $\bar{L}D\bar{L}^T$ (базовый учебник, с. 97):

Для k=1 до n–1
  Для i=k+1 до n
    $l_{ik}=p_{ik}/p_{kk}$
    Для j=k+1 до i
      $p_{ij}=p_{ij}–l_{ik}p_{jk}$

Рассмотрим один из возможных способов реализации данного алгоритма. Пусть в одномерном массиве P хранится нижняя треугольная часть ПО-матрицы P. Замещение элементов матрицы P элементами матрицы L будет происходить сразу при их вычислении.

Заменим 3-ю строку $l_{ik}=p_{ik}/p_{kk}$ на $p_{ik}=p_{ik}/p_{kk}$. Соответственно, заменим 5-ую строку, подставив p_{ik} вместо l_{ik} и умножив p_{jk} на p_{kk}:

$$p_{ij}=p_{ij}–l_{ik}p_{jk}=p_{ij}–p_{ik}p_{jk}p_{kk}.$$

Для обращения к (i,j)-му элементу будем пользоваться формулой $k=(i–1)i/2+j$.

```
for k=1:(n-1)
  for i=(k+1):n
      ik=((i-1)*i/2)+k;
      kk=((k-1)*k/2)+k;
      P(ik)=P(ik)/P(kk);
     for j=(k+1):i
        ij=((i-1)*i/2)+j;
```

```
            jk=((j-1)*j/2)+k;
            P(ij)=P(ij)-P(ik)*P(jk)*P(kk);
        end
    end
end
```

В результате выполнения алгоритма получим нижнюю треугольную матрицу *P* с положительными элементами на диагонали.

Матрицу $\bar{L}$ найдем из получившейся матрицы *P*, заменив все диагональные элементы единицами, диагональную матрицу *D* – заполнив ее диагональ диагональными элементами матрицы *P*.

4. Функция решения системы линейных алгебраических уравнений

Рассмотрим детали реализации алгоритма нахождения решения СЛАУ по разложению Холесского. Решение будем искать в два этапа. Но сначала приведем математическое обоснование для варианта $\bar{L}D\bar{L}^T$-разложения:

$$Px = f \Rightarrow \bar{L}D\bar{L}^T x = f \Rightarrow (D\bar{L}^T x = y) \Rightarrow Ly = f \Rightarrow (z = \bar{L}^T x) \Rightarrow Dz = y \Rightarrow \bar{L}^T x = z \Rightarrow x.$$

Аналогичная цепочка вычислений получается и для других видов разложения. Рассмотрим реализацию алгоритма решения СЛАУ на примере $\bar{L}D\bar{L}^T$-разложения.

Первый проход: известны матрица $\bar{L}$ и вектор *f*, находим неизвестный вектор *y* с помощью прямой подстановки:

```
% метод прямой подстановки (скалярных произведений)
    for i=1:n
        sum=0;
      if (i>1)
          for j=1:(i-1)
            ij=((i-1)*i/2)+j;
            sum=sum+L(ij)*y(j);
          end
      end
        y(i)=f(i)-sum;
    end
```

Второй проход: находим произведение матриц *Dz=y*. Известен вектор *y* и матрица *D*. Находим неизвестный вектор *z*:

```
for k=1:n
   kk=((k-1)*k/2)+k;
   z(k)=y(k)/P(kk);
end
% Найденное решение находится в массиве z
```

Третий проход: известны матрица $\bar{L}$ и вектор z, находим искомый вектор x с помощью обратной подстановки:

```
%метод обратной подстановки (скалярных произведений)
   for i=n:-1:1
         sum=0;
      if (i<n)
            for j=i+1:n
               ij=((i-1)*i/2)+j;
               sum =sum+P(ij)*x(j);
            end
      end
         x(i)=z(i)-sum;
   end
% Найденное решение находится в массиве x
```

5. Функция решения системы линейных алгебраических уравнений с ленточной матрицей P

Для решения СЛАУ с ленточной матрицей можно воспользоваться алгоритмом, указанным в п. 4. Для определения положения элемента в одномерном массиве воспользоваться алгоритмом, описанным в п. 2.

Тестовые задачи для отладки программного кода можно найти в базовом учебнике на с. 163-185.

6. Эксперимент 1 «Количество арифметических операций»

Написать реализацию первого эксперимента для исследования количества арифметических операций. Описание эксперимента можно найти в базовом учебнике на с. 103.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Квадратные корни			Умножение и деление		
	а	б	в	а	б	в

где а, б, в означает способ вычисления количества арифметических операций: а – фактическое, б – теоретически точное, в – теоретически приближенное.

Каждая строка в таблице будет зависеть от n – порядка матрицы. Число n должно изменяться от 5 до 100 через 5 порядков, т. е. в таблице будет 20 строк.

2) Присвоить n очередное значение.
3) Сгенерировать матрицу P в соответствии с **заданием 2**.
4) Выполнить разложение матрицы P в соответствии с **заданием 3**.
5) Подсчитать фактическое число операций извлечения квадратного корня, умножения и деления с помощью специальных счетчиков, которые добавляются в функцию разложения.
6) Подсчитать теоретически точное число операций извлечения квадратного корня (равное n – размерности ПО-матрицы P), операций умножения и деления (равное, например, для $\overline{L}D\overline{L}^T$-разложения $\sum_{i=1}^{n}((n-i)(i-1)+(i-1)+1)$) в зависимости от размерности матрицы n.
7) Подсчитать теоретически приближенное число операций извлечения квадратного корня (равное n – размерности ПО-матрицы P), операций умножения и деления (равное $n^3/6$) в зависимости от n при $n \to \infty$.
8) Добавить в таблицу полученные экспериментальные данные: порядок n, фактическое, теоретически точное и теоретически приближенное число операций извлечения квадратного корня, умножения и деления.

7. Эксперимент 2 «Решение СЛАУ с заполненной матрицей P»

Реализовать второй эксперимент для исследования скорости и погрешности решения СЛАУ в случае, когда матрица P является заполненной. Описание эксперимента можно найти в базовом учебнике на с. 103-105.

Перед проведением эксперимента необходимо написать процедуру, вычисляющую погрешность решения СЛАУ.

В данный проект следует включить программный код, созданный ранее в проекте № 1, для того, чтобы иметь возможность сравнить метод

Холесского с методом Гаусса по времени выполнения и по погрешности вычислений.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Время		Погрешность	
	метод Гаусса	метод Холесского	метод Гаусса	метод Холесского

Каждая строка в таблице будет зависеть от n – порядка матрицы. Число n должно изменяться от 5 до 100 через 5 порядков, т. е. в таблице будет 20 строк.

2) Присвоить n очередное значение.
3) Заполнить матрицу P случайными числами в соответствии с **заданием 2** и сгенерировать вектор b.
4) Выполнить разложение матрицы P в соответствии с **заданием 3**.
5) Найти решение СЛАУ x в соответствии с **заданием 4**.
6) Подсчитать скорость решения задачи как сумму времени разложения матрицы и времени решения СЛАУ.

Сделать это можно с помощью функций tic и toc:

```
tic
    <операция 1>
    ...
    <операция n>
time = toc; %затраченное время
```

Можно также воспользоваться другими стандартными функциями работы со временем, например, `clock` или `etime`.

7) Оценить погрешность решения СЛАУ:
   - Задать точное решение x^*. В качестве точного решения взять вектор $x^*=(1, 2, \ldots, n)^T$, где n – порядок матрицы.
   - Образовать правые части $f = Px^*$, матрица P известна из п. 3.
   - Найти решение СЛАУ $Px=f$.
   - Вычислить погрешность решения СЛАУ как максимальный модуль разности компонента вектора точного решения и вектора найденного решения.

8) Добавить в таблицу полученные экспериментальные данные для разложения Холесского и для метода Гаусса (использовать данные лабораторного проекта № 1): порядок n, время выполнения, погрешность решения СЛАУ.
9) Записать в файл полученные экспериментальные данные для разложения Холесского: порядок n, время выполнения, погрешность решения СЛАУ.
10) Пункты 2-9 повторить для всех значений n.
11) Построить графики решения СЛАУ с заполненной матрицей P:
   - зависимость времени решения от порядка матриц для методов Гаусса и Холесского;
   - зависимость погрешности решения от порядка матриц для методов Гаусса и Холесского.
12) Сравнить полученные данные со своим вариантом из лабораторного проекта № 1.
13) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

8. Эксперимент 3 «Решением СЛАУ с разреженной матрицей P»

Написать реализацию третьего эксперимента для исследования скорости и погрешности решения СЛАУ в случае, когда матрица P является разреженной. Описание эксперимента можно найти в базовом учебнике на с. 104, 105.

Перед проведением эксперимента необходимо написать процедуру, вычисляющую погрешность решения СЛАУ (взять из лабораторного проекта № 1).

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Время		Погрешность	
	Заполненная матрица	Разреженная матрица	Заполненная матрица	Разреженная матрица

Для каждого текущего значения n порядка матрицы P необходимо решать две системы: одну – с заполненной матрицей P (см. п. 5 задания), другую – с разреженной матрицей P (см. п. 7 задания).

Каждая строка в таблице будет зависеть от n – количества уравнений в системе. Число n должно изменяться от 100 до 200 через 5 порядков, т. е. в таблице будет 21 строка.

2) Присвоить n очередное значение.
3) Заполнить случайными числами заполненную матрицу P в соответствии с **заданием 1**. Сгенерировать вектор f.
4) Выполнить разложение матрицы P в соответствии с **заданием 3**.
5) Найти решение СЛАУ x в соответствии с **заданием 4**.
6) Подсчитать скорость решения задачи так же, как и в **эксперименте 2** с заполненной матрицей.
7) Оценить погрешность решения СЛАУ так же, как и в **эксперименте 2**.
8) Добавить в таблицу полученные экспериментальные данные: порядок n, время выполнения, погрешность решения СЛАУ.
9) Записать в файл полученные экспериментальные данные: порядок n, время выполнения, погрешность решения СЛАУ.
10) Заполнить разреженную ленточную матрицу P случайными числами в соответствии с **заданием 2**.
11) Выполнить разложение матрицы P в соответствии с **заданием 3**.
12) Найти решение СЛАУ x в соответствии с **заданием 4**.
13) Подсчитать скорость решения задачи так же, как и в **эксперименте 2** с заполненной матрицей.
14) Оценить погрешность решения СЛАУ:
    - Задать точное решение x^*. В качестве точного решения взять вектор $x^*=(1, 2,. . . , n)^T$, где n – порядок матрицы.
    - Вычислить правые части $f = Px^*$.
    - Найти решение СЛАУ $Px=f$. В случае, если при решении системы $Px=f$ выяснится, что матрица P вырождена (плохо обусловлена), то сгенерировать новую матрицу того же порядка и решить систему линейных алгебраических уравнений с новой матрицей P и новой правой частью f.
    - Вычислить погрешность решения СЛАУ как максимальный модуль разности компонентов вектора точного решения и вектора найденного решения.
15) Добавить в таблицу полученные экспериментальные данные: порядок n, время выполнения, погрешность решения СЛАУ.
16) Записать в файл полученные экспериментальные данные: порядок n, время выполнения, погрешность решения СЛАУ.

17) Пункты 2-15 повторить для всех значений n.
18) Построить следующие графики решения СЛАУ с разреженной матрицей P:
   - зависимость времени решения от порядка матриц для обычного метода Холесского и для метода с учетом разреженности матрицы P;
   - зависимость погрешности решения от порядка матриц для обычного метода Холесского и для метода с учетом разреженности матрицы P.

   При построении графиков использовать данные из файла.
19) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

5 Лабораторный проект № 3
«Ортогональные преобразования»

Перед выполнением лабораторного проекта рекомендуем подробно ознакомиться с материалом базового учебника, с. 107-140.

5.1 Теория

Алгоритмы отражений Хаусхолдера

Рассмотрим разложение вида $TA=R$, где A – матрица размера $m \times n$, T – матрица отражений Хаусхолдера, R – верхняя треугольная матрица.

1. Столбцово-ориентированный алгоритм Хаусхолдера:	*2. Строчно-ориентированный алгоритм Хаусхолдера:*
Для k=1 до min(m–1,n) $s_k = -\mathrm{sgn}[A(k,k)]\left(\sum_{i=k}^{m}[A(i,k)]^2\right)^{1/2}$, $u_k(1) = A(k,k) - s_k$, $u_k(i) = A(k+i-1,k)$, $i = 2,\ldots,m-k+1$, $\alpha_k = 1/(s_k u_k(1))$ $(\alpha_k < 0)$. Для j=k+1,…,n $\lambda := \alpha_k \sum_{i=k}^{m} u_k(i-k+1)A(i,j)$, Для $i = k, k+1, \ldots, m$ $A(i,j) := A(i,j) + \lambda u_k(i-k+1)$.	Для k=1 до min(m–1,n) $s_k = -\mathrm{sgn}[A(k,k)]\left(\sum_{i=k}^{m}[A(i,k)]^2\right)^{1/2}$, $u_k(1) = A(k,k) - s_k$, $u_k(i) = A(k+i-1,k)$, $i = 2,\ldots,m-k+1$, $\alpha_k = 1/(s_k u_k(1))$ $(\alpha_k < 0)$. Для j=k+1,…,n $\lambda_k(j-k) := \alpha_k \sum_{i=k}^{m} u_k(i-k+1)A(i,j)$, Для i=$k$, k+1,…,m Для j=k+1,…,n $A(i,j) := A(i,j) + \lambda_k(j-k)u_k(i-k+1)$.

Алгоритмы вращений Гивенса

Рассмотрим разложение вида $PA=R$, где A – матрица размера $m\times n$, P – матрица вращений Гивенса, R – верхняя треугольная матрица.

1. Столбцово ориентированная схема Гивенса:	*2. Строчно ориентированная схема Гивенса:*
Для j=1 до min(m–1, n) r_j:=a_{jj} Для i=j+1 до m a:=r_j; b:=a_{ij} получить (r, c, s, ζ) из (a, b) ($a_{jj} \triangleq r_j$):=r ($a_{ij} \triangleq \zeta_{j,i}$):=$\zeta$ $c_{j,i}$:=c; $s_{j,i}$:=s Для s=j+1 до n Для i=j+1 до m α=$c_{j,i}a_{js}$+$s_{j,i}a_{is}$ a_{is}=–$s_{j,i}a_{js}$+$c_{j,i}a_{is}$ a_{js}:= α	Для j=1 до min(m–1,n) Для i=j+1 до m a:=a_{jj}; b:=a_{ij} получить (r, c, s, ζ) из (a, b) ($a_{jj} \triangleq r_j$):=r ($a_{ij} \triangleq \zeta_{j,i}$):=$\zeta$ Для s=j+1 до n α=ca_{js}+sa_{is} a_{is}=–sa_{js}+ca_{is} a_{js}:= α

Процедура вычисления (r, c, s, ζ) из (a, b):

$\sigma = \begin{cases} \operatorname{sgn}(a), & \lvert a\rvert > \lvert b\rvert \\ \operatorname{sgn}(b), & \lvert a\rvert \le \lvert b\rvert \end{cases}$	$r = \sigma\sqrt{a^2+b^2}$
$c = \begin{cases} \dfrac{a}{r}, & r \ne 0 \\ 1, & r = 0 \end{cases}$	$s = \begin{cases} \dfrac{b}{r}, & r \ne 0 \\ 0, & r = 0 \end{cases}$
$\zeta = \begin{cases} s, & \lvert a\rvert > \lvert b\rvert \\ \dfrac{1}{c}, & \lvert a\rvert \le \lvert b\rvert \,\&\, c \ne 0 \\ 1, & c = 0 \end{cases}$	

Процедура получения (c, s) из ζ (восстановление косинуса и синуса угла поворота по величине ζ)*:*

$\zeta = 1 \Rightarrow \begin{cases} c := 0 \\ s := 1 \end{cases}$,	$d := \lvert 1-\zeta^2\rvert^{1/2}$,	$\lvert\zeta\rvert < 1 \Rightarrow \begin{cases} c := d \\ s := \zeta \end{cases}$,	$\lvert\zeta\rvert > 1 \Rightarrow \begin{cases} c := 1/\zeta \\ s := d/\lvert\zeta\rvert \end{cases}$.

Процедура вычисления вектора y (вычисление вектора y теми преобразованиями $P_{j,i}$ произвольного вектора $z \in R^m$, которые сохранены в рабочих признаках $\zeta_{j,i}$ и восстанавливаются из них):

```
Для j=1 до min(m-1,n)
    Для i=j+1 до m
        получить (c, s) из ζ_{j,i}
        α=cy_{jj}+sy_{ij}
        y_{ij}=–sy_{jj}+cy_{ij}
        y_{jj}:= α
```

Ортогонализация Грама-Шмидта

Ортогонализацией Грама-Шмидта называют вычисление по матрице A такой ортогональной матрицы Q и верхней треугольной матрицы R, что $A=QR$.

Классическая Грама-Шмидта Ортогонализация (ГШО)

Этот вариант алгоритма предполагает вычисление ненулевых элементов матрицы R по столбцам, начиная с самого короткого (одноэлементного) столбца.

Для k=1 до n

$r_{ik}=a_k^{\mathrm{T}}q_i,\ \ i=1,2,\ldots,\ \mathrm{k}-1,$

$$v=a_k-\sum_{i=1}^{k-1} r_{ik}q_i\,,$$

$r_{kk}=(v^{\mathrm{T}}v)^{1/2},$

$q_k=v/r_{kk}$.

Модифицированная ГШО (МГШО)

В этом варианте ненулевые элементы матрицы R вычисляются по строкам, начиная с самой длинной (состоящей из n элементов) строки.

```
Для k=1 до n
    r_{kk}=||a_k||=(a_k^T a_k)^{1/2},
    a_k=a_k /r_{kk} ,
    Для j=k+1 до n
        r_{kj}=a_j^T a_k ,
        a_j=a_j–r_{kj}a_k .
```

МГШО с выбором ведущего вектора

Этот вариант МГШО-алгоритма использует стратегию выбора ведущего вектора. В качестве очередного, подлежащего ортогонализации вектора, выбирается тот из оставшихся столбцов матрицы A, который имеет наибольшую длину (евклидову норму). Хотя эта стратегия требует дополнительных вычислительных затрат, в некоторых плохо обусловленных задачах она так же полезна, как и выбор главного элемента в методе Гаусса.

Для k=1 до n
    $q(k)=k$,
Для k=1 до n
    Для $s=k$ до n
        Найти № l, для которого $\|a_{q(l)}\|=\max_s\|a_{q(l)}\|$,
    Переставить номера: $q(k)\leftrightarrow q(l)$,
    $r_{q(k),q(k)}=\|a_{q(k)}\|=(a_{q(k)}{}^{T}a_{q(k)})^{1/2}$,
    $a_{q(k)}=a_{q(k)}/r_{q(k),q(k)}$,
    Для $j=k$+1 до n
        $r_{q(k),q(j)}=a_{q(j)}{}^{T}a_{q(k)}$,
        $a_{q(j)}=a_{q(j)}-r_{q(k),q(j)}a_{q(k)}$.

5.2 Задание на лабораторный проект

1. Система меню для взаимодействия пользователя с программой.
2. Функция факторизации матрицы.
3. Функция решения системы линейных алгебраических уравнений.
4. Функция вычисления определителя матрицы.
5. Функция обращения матрицы через решение системы $AX=E$ на основе метода ортогонального преобразования (в соответствии со своим вариантом).
6. Эксперимент 1 «Подсчет количества арифметических операций».
7. Эксперимент 2 «Решение СЛАУ».
8. Эксперимент 3 «Решение СЛАУ для плохо обусловленных матриц».
9. Эксперимент 4 «Обращение матриц».

Варианты заданий

Вариант заполнения матрицы R	Отражения Хаусхолдера		Вращения Гивенса		Ортогонализация Грама–Шмидта		
	a	b	a	b	c	d	e
 $\triangleq R_{\mathrm{ne}}$	1	2	3	4	5	6	7
 $\triangleq R_{\mathrm{nw}}$	8	9	10	11	12	13	14
 $\triangleq R_{\mathrm{se}}$	15	16	17	18	19	20	21
 $\triangleq R_{\mathrm{sw}}$	22	23	24	25	26	27	28

a – столбцово-ориентированный алгоритм;

b – строчно-ориентированный алгоритм;

c – классическая схема;

d – модифицированная схема;

e – модифицированная схема с выбором ведущего вектора.

Различные варианты заполнения матрицы R получаются изменением порядка действий в рассмотренных алгоритмах.

5.3 Методические рекомендации и примеры

1. Система меню для взаимодействия пользователя с программой

См. лабораторный проект № 1 п. 1.

Предусмотреть предупреждение о невозможности решения указанных задач из-за присутствия (почти) линейно зависимых векторов среди столбцов матрицы A (в пределах ошибок округления ЭВМ или другого, заранее определенного критерия).

2. Функция факторизации матрицы

Подробное описание алгоритмов факторизации можно найти в базовом учебнике.

Рассмотрим возможную реализацию функции факторизации на примере столбцово-ориентированного алгоритма отражения Хаусхолдера для варианта заполнения матрицы R_{ne} (подробное описание алгоритма можно найти на с. 116 базового учебника).

```
A=randi([-10 10], m, n);%генерируем матрицу A
function [ A, s]=cHouseholder( A )
[m,n]=size(A);
s=zeros(1,min(m-1,n));%генерируем вектор s
for k=1:(min(m-1,n))
  %вычисляем s_k
  s(k)=-sign(A(k,k))*sqrt(A(k:m,k)'*A(k:m,k));
%вычисляем u (u замещает столбцы из нижней треугольной части
матрицы A)
  A(k,k)=A(k,k)-s(k);
  %вычисляем α
  d=1/(s(k)*A(k,k));
  for j=(k+1):n
    %считаем сумму
    sum_uA = 0;
    for i=k:m
      sum_uA=sum_uA+A(i,k)*A(i,j);
    end
    %считаем λ
    l=d*sum_uA;
    for i=k:m
      A(i,j)=A(i,j)+l*A(i,k);
    end
  end
end
end
```

Особенность программной реализации заключается в том, что в результате работы алгоритма в строго верхней треугольной части матрицы A получаем элементы матрицы R, в строке s получаем диагональные элементы матрицы R, а в нижней треугольной части матрицы A сохраняются вектора $u(k)$.

3. Функция решения системы линейных алгебраических уравнений

Рассмотрим детали реализации алгоритма нахождения решения СЛАУ по разложению матрицы $A=QR$.

Если известно разложение матрицы $A=QR$, тогда решение СЛАУ $Ax=b$ сводится к решению эквивалентной системы $Rx=Q^Tb$.

Сначала найдем произведение $Q^Tb=b_1$. Для этого применим к вектору b цепочку преобразований, которая была проделана над столбцами матрицы A (см. программный код функции факторизации).

```
b1=b;
% A – преобразованная после факторизации матрица
for k=1:(min(m-1,n))
  %вычисляем α
  d=1/(s(k)*A(k,k));
    %считаем сумму
    sum_uA = 0;
    for i=k:m
      sum_uA=sum_uA+A(i,k)*b1(i);
    end
    %считаем λ
    l=d*sum_uA;
    for i=k:m
      b1(i)=b1(i)+l*A(i,k);
    end
end
```

Далее найдем искомый вектор x из равенства $Rx=b_1$:

```
for i=n:-1:1
sum=0;
      if (i<n)
            for j=i+1:n
            sum=sum+R(i,j)*x(j);
            end
      end
            x(i)=(b1(i)-sum)/R(i,i);
      end
```

4. Функция вычисления определителя матрицы

Если нам известно разложение матрицы $A=QR$, тогда определитель квадратной матрицы A посчитаем по формуле

$$\det A = \pm \det R = \pm \prod_{i=1}^{n} R_{ii} .$$

5. Функция обращения матрицы через решение системы AX=E на основе метода ортогонального преобразования (в соответствии со своим вариантом)

Для матрицы $A=A(n, n)$ имеем $A=QR$. Отсюда $A^{-1}=R^{-1}Q^{-1}=R^{-1}Q^{T}$. Следовательно, A^{-1} есть решение матричного уравнения $RX=Q^{T}$. Чтобы найти i-й столбец матрицы A^{-1}, надо в качестве правой части взять i-й столбец единичной матрицы, применить к нему заданную цепочку преобразований и решить систему с треугольной матрицей R (см. решение СЛАУ в задании 3).

6. Эксперимент 1 «Подсчет количества арифметических операций»

Написать реализацию первого эксперимента для исследования количества операций. Описание эксперимента можно найти в базовом учебнике на с. 138.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Теоретическое число операций				Реальное число операций			
	a	b	c	d	a	b	c	d

где a, b, c, d означают количество операций: a – сложения, b – умножения, c – деления, d – извлечения квадратного корня.

Каждая строка в таблице будет зависеть от n – количества уравнений в системе. Число n должно изменяться от 10 до 100 через 10 порядков, т. е. в таблице должно быть 10 строк.

2) Присвоить n очередное значение.
3) Сгенерировать матрицу A случайным образом (см. лабораторный проект № 1).
4) Выполнить факторизацию матрицы A в соответствии с **заданием 2**.
5) Найти решение СЛАУ x в соответствии с **заданием 3**.
6) Подсчитать теоретическое (оценочное) число операций сложения, число операций умножения, число операций деления и число операций извлечения квадратного корня.

7) Подсчитать реальное число операций сложения, число операций умножения, число операций деления и число операций извлечения квадратного корня с помощью специальных счетчиков, которые добавляются в функции факторизации и решения.
8) Добавить в таблицу полученные экспериментальные данные: порядок n, число операций сложения, число операций умножения, число операций деления и число операций извлечения квадратного корня.

7. Эксперимент 2 «Решение СЛАУ»

Написать реализацию первого эксперимента для исследования скорости и погрешности решения СЛАУ. Подробное описание эксперимента можно найти в базовом учебнике на с. 138.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Время	Погрешность

Каждая строка в таблице будет зависеть от n – количества уравнений в системе. Число n должно изменяться от 10 до 100 через 10 порядков, т. е. в таблице будет 10 строк.

2) Присвоить n очередное значение.
3) Заполнить матрицу A случайными числами и сгенерировать вектор b.
4) Выполнить факторизацию матрицы A в соответствии с **заданием 2**.
5) Найти решение СЛАУ x в соответствии с **заданием 3**.
6) Выполнить факторизацию матрицы A в соответствии с **заданием 2** из лабораторной работы № 1.
7) Найти решение СЛАУ x в соответствии с **заданием 3** из лабораторной работы № 1.
8) Подсчитать скорость решения задачи как сумму времени факторизации матрицы и времени решения СЛАУ (для двух методов: метода исключения в лабораторном проекте № 1 (п. 4 и 5) и метода ортогонального разложения в лабораторном проекте № 3 (п. 6 и 7).
9) Оценить погрешность решения СЛАУ (для п. 5 и 7):

- Задать точное решение x^*. В качестве точного решения нужно взять вектор $x^*=(1,2,\ldots,n)^T$, где n – порядок матрицы.
- Образовать правые части $b=Ax^*$, матрица A известна из п. 3.
- Найти решение СЛАУ $Ax=b$.
- Вычислить погрешность решения СЛАУ как максимальный модуль разности компонента вектора точного решения и вектора найденного решения.

  Добавить в таблицу и файл для хранения экспериментальных данных полученные экспериментальные данные: порядок n, время выполнения, погрешность решения СЛАУ, теоретическое и реальное число операций.

10) Пункты 2-10 повторить для всех значений n.
11) Построить следующие графики решения СЛАУ (для двух методов):
    - зависимость времени решения от порядка матриц;
    - зависимость погрешности решения от порядка матриц.

    При построении графиков необходимо использовать данные из файла с экспериментальными данными.
12) Проанализировать полученные данные, сравнить погрешность решения и затраты машинного времени для методов в лабораторном проекте № 1 и лабораторном проекте № 6. Сделать содержательные выводы об эффективности и качестве реализованных численных методов.

8. Эксперимент 3 «Решение СЛАУ для плохо обусловленных матриц»

Написать реализацию второго эксперимента для исследования погрешности решения СЛАУ в случае, когда матрица A является плохо обусловленной. Подробное описание эксперимента можно найти в базовом учебнике на с. 139. Перед проведением эксперимента написать функцию, вычисляющую погрешность решения СЛАУ (см. эксперимент 1).

Методика проведения эксперимента:

Для каждой из 10 плохо обусловленных матриц на с. 47, 48 базового учебника выполнить следующее:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Погрешность

Каждая строка в таблице будет зависеть от n – количества уравнений в системе. Если матрица A имеет фиксированный размер (матрицы с номерами 2, 3, 6, 10), тогда n = размеру матрицы, и в таблице будет только одна строка. Если порядок матрицы не фиксирован (матрицы с номерами 1, 4, 5, 7, 8, 9), тогда число n должно изменяться от 4 до 40 через 4 порядка, т. е. в таблице будет 10 строк.

2) Присвоить n очередное значение.
3) Заполнить матрицу A в соответствии с ее номером и сгенерировать вектор b в соответствии с **заданием 1 (для заданной матрицы!)**.
4) Оценить погрешность решения СЛАУ аналогично как в эксперименте 1.
5) Добавить в таблицу и файл для хранения экспериментальных данных полученные экспериментальные данные: порядок n, погрешность решения СЛАУ.
6) Пункты 2-6 повторить для всех значений n.
7) Построить графики зависимости погрешности решения СЛАУ от порядка матрицы.
8) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

9. Эксперимент 4 «Обращение матриц»

Написать реализацию третьего эксперимента для исследования скорости и погрешности процедур обращения матриц. Подробное описание эксперимента можно найти в базовом учебнике на с. 117-120.

Перед проведением эксперимента написать процедуру, вычисляющую погрешность найденной обратной матрицы: формулы (2.19), (2.20) на с. 49, 50 базового учебника позволяют вычислить погрешность обращения матрицы – для вычисления погрешности обращения матрицы A необходимо в соответствии с формулами сначала умножить исходную матрицу на найденную обратную и вычесть результат из единичной матрицы, затем найти норму найденной разности и поделить ее на норму исходной матрицы. Норма матрицы вычисляется по формуле (2.20) – это есть максимальная сумма модулей элементов строк матрицы.

Методика проведения эксперимента:

1) Распечатать на экране шапку таблицы с экспериментальными данными:

Порядок матрицы	Время первым способом	Время вторым способом	Погрешность первым способом	Погрешность вторым способом

Каждая строка в таблице будет зависеть от n – размера матрицы. Число n должно изменяться от 10 до 100 через 10 порядков, т. е. в таблице будет 10 строк.

2) Присвоить n очередное значение.
3) Заполнить матрицу A случайными числами.
4) Найти обратную матрицу первым способом через решение системы $AX=E$ на основе метода исключения Гаусса в лабораторном проекте № 1 (в соответствии со своим вариантом).
5) Подсчитать скорость обращения матрицы (см. эксперимент 1) первым способом.
6) Найти погрешность обращения матрицы.
7) Найти обратную матрицу вторым способом в соответствии с **заданием 6**.
8) Подсчитать скорость обращения матрицы (см. эксперимент 1) вторым способом.
9) Найти погрешность обращения матрицы.
10) Добавить в таблицу полученные экспериментальные данные: порядок n, время и погрешность обращения первым и вторым способами.
11) Пункты 2-11 повторить для всех значений n.
12) Проанализировать полученные данные и сделать содержательные выводы об эффективности и качестве реализованных численных методов.

Литература

Литература по вычислительным методам алгебры

Базовый учебник:

1. Семушин И. В. Вычислительные методы алгебры и оценивания. – Ульяновск: УлГТУ, 2011. – 366 с. – ISBN 978-5-9795-0902-0.

Основная:

1. Семушин И. В. Вычислительные методы алгебры и оценивания. – Ульяновск: УлГТУ, 2011. – 366 с. – ISBN 978-5-9795-0902-0.
2. Воеводин В. В. Численные методы алгебры. Теория и алгоритмы. – М.: Наука, 1966.
3. Самарский А. А., Гулин А. В. Численные методы. – М.: Наука, 1989.

Дополнительная:

Алгоритмы метода Гаусса:

1. Калиткин Н. Н. Численные методы. – М.: Наука, 1978.
2. Воеводин В. В., Кузнецов Ю. А. Матрицы и вычисления. – М.: Наука, 1984.
3. Годунов С. К. Решение систем линейных уравнений. – Новосибирск: Наука, 1980.
4. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем: пер. с англ. – М.: Мир, 1991.
5. Писсанецки С. Технология разреженных матриц: пер. с англ. – М.: Мир, 1988.
6. Тьюарсон Р. Разреженные матрицы: пер. с англ. – М.: Мир, 1977.
7. Фаддеев Л. К., Фаддеева В. Н. Вычислительные методы линейной алгебры. – М.: Физматгиз, 1963.
8. Stoer J., Bulirsch R. «Introduction to Numerical Analysis,» Translated from German. 2nd ed. New York: Springer-Verlag, 1993. [Оригинальное издание: J. Stoer, R. Bulirsch. «Einführung in die Numerische Mathematik,» I, II. Berlin, Heidelberg: Springer-Verlag, 1972, 1976.]

Выбор ведущего элемента:

1. Воеводин В. В. Вычислительные основы линейной алгебры. – М.: Наука, 1977.
2. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений. – М.: Наука, 1986.
3. Райс Дж. Матричные вычисления и математическое обеспечение: пер. с англ. – М.: Мир, 1984.
4. Херцбергер Ю., Алефельд Г. Введение в интервальные вычисления: пер. с англ. – М.: Мир, 1987.
5. Fröberg Carl-Erik. «Introduction to Numerical Analysis». 2nd ed. Reading Massachusetts Menlo Park (California) London Don Mills (Ontario): Addison-Wesley Publishing Company, 1969.
6. Golub G. H., Ortega J. M. «Scientific Computing and Differential Equations. An Introduction to Numerical Methods». San Diego New York Boston London Sydney Tokyo Toronto: Academic Press, 1992.
7. Noble B., Daniel J. W. «Applied Linear Algebra». 2nd ed. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.
8. Stoer J., Bulirsch R. «Introduction to Numerical Analysis: Translated from German. 2nd ed. N. Y.: Springer-Verlag, 1993.

Компактные схемы:

1. Воеводин В. В., Кузнецов Ю. А. Матрицы и вычисления. – М.: Наука, 1984.
2. Калиткин Н. Н. Численные методы. – М.: Наука, 1978.
3. Райс Дж. Матричные вычисления и математическое обеспечение: пер. с англ. – М.: Мир, 1984.
4. Размыслов Ю. П., Ищенко С. Я. Практикум по вычислительным методам алгебры. – М.: Изд-во МГУ, 1989.
5. Семушин И. В., Куликов Г. Ю. Сборник лабораторных работ и контрольных, тестовых заданий по курсу «Вычислительная линейная алгебра». – Ульяновск: УлГТУ, 2000.
4. Stoer J., Bulirsch R. «Introduction to Numerical Analysis: Translated from German. 2nd ed. N. Y.: Springer-Verlag, 1993.

Алгоритмы метода Жордана:

1. Самарский А. А., Гулин А. В. Численные методы. – М.: Наука, 1989.

Вычисление обратной матрицы:

1. Беклемишев Д. В. Дополнительные главы линейной алгебры. – М.: Наука, 1983.
2. Bierman G. J. «Factorization Methods for Discrete Sequential Estimation». N. Y.: Academic Press, 1977.
3. Stoer J., Bulirsch R. «Introduction to Numerical Analysis: Translated from German. 2nd ed. N. Y.: Springer-Verlag, 1993.

Литература по MATLAB

1. Кетков Ю. Л., Кетков А. Ю., Шульц М. М. MATLAB 7. Программирование, численные методы. – СПб.: БХВ-Петербург, 2005.
2. Алексеев Е. Р., Чеснокова О. В. Решение задач вычислительной математики в пакетах MathCad 12, MATLAB 7, Maple 9. Самоучитель. – М.: НТ Пресс, 2005. – 496 с.
3. http://www.mathworks.com/help/matlab/

Приложение

Пример отчета по лабораторному проекту № 1

Государственное бюджетное образовательное учреждение
высшего профессионального образования
«Ульяновский государственный университет»

Факультет Математики и информационных технологий
Кафедра Информационных технологий

Отчет по лабораторному проекту № 1
«Стандартные алгоритмы *LU*-разложения»

Вариант № 1:

$L\overline{U}$ -разложение на основе гауссова исключения по столбцам с выбором главного элемента по столбцу

Выполнил: Сидоров И. П.
гр. МОАИС-О-10/1

_____

Проверила: доц. Цыганова Ю. В.

_____

Ульяновск – 2014 г.

Цель проекта: приобретение практических навыков реализации алгоритмов *LU*-разложения в виде программного проекта на языке MATLAB.

Задания:

1. Система меню для взаимодействия пользователя с программой и генерация исходных данных задачи.
2. Функция факторизации матрицы.
3. Функция решения системы линейных алгебраических уравнений.
4. Функция вычисления определителя матрицы.
5. Функция обращения матрицы через решение системы *AX=E*.
6. Функция обращения матрицы через элементарные преобразования разложения.
7. Эксперимент 1 «Решение СЛАУ для случайных матриц».
8. Эксперимент 2 «Решение СЛАУ с плохо обусловленными матрицами».
9. Эксперимент 3 «Обращение случайных матриц».

Задание № 1 *«Система меню для взаимодействия пользователя с программой и генерация исходных данных задачи»*

Взаимодействие с пользователем осуществляется через командную строку МАТЛАВ. Пользователю на выбор предлагается выбрать одно из действий: ввести данные, обработать данные (выполнить факторизацию матрицы, решить СЛАУ, вычислить определитель, вычислить обратную матрицу) или выйти из программы. Если пользователь выбирает обработку данных, но данные не введены, то программа выводит сообщение «Введите данные для обработки!» и предложит выбрать действие заново.

Программный код меню:

```
clc;
disp('Лабораторный проект № 1 "Стандартные алгоритмы
      LU-разложения"');
disp('Вариант № 1');
disp('Выполнил студент гр. МОАИС-О-10/1 Сидоров И.П.');
while(1)
  param = input('Нажмите "1" для ввода данных;\nНажмите "2" для обработки данных;\nНажмите "3" для выхода из программы\n', 's');
  switch param
    case '1',
```

```
    A = input('Введите матрицу: \n A = \n');
    n_ar =size(A);
    n = n_ar(1);
    b = input('Введите вектор: \n b = \n');
    n_ar_b =size(A);
    n_b = n_ar_b(1);
    if (n_b ~= n)
      disp('Размерности A и b не совпадают!');
      return;
    end
  case '2',
    if (exist('A', 'var') == 0)
      disp('Введите данные для обработки!');
    else
      A_copy=A;
%_____________________________________________________________________
      disp('Факторизация:');
      [A, p, znak] = factorization(A, n);
       disp('Факторизация матрицы A с учетом перестановок');
      % PA - матрица с учетом перестановок
      PA=printPA(A,n,p)
      disp('Нижняя треугольная часть L');
      L=tril(PA)
      disp('Верхняя треугольная часть U');
      U=triu(PA)-eye(n).*PA+eye(n)
%_____________________________________________________________________
      x=zeros(n,1);
      x1=zeros(n,1);
      disp('Решение СЛАУ');
      x=SLAE(A, b, n, p)
      disp('Проверка решения');
      x1=A_copy\b
%_____________________________________________________________________
      disp('Детерминант матрицы A');
      d=determinant(A, n, p, znak)
      disp('Проверка детерминанта');
```

```
            d1=det(A_copy)
    %_____________________________________________________________
            A11=zeros(n,n);
            A12=zeros(n,n);
             disp('Обращение матрицы через решение системы AX=E');
             A11=inversion_1(A, n, p)
             disp('Обращение матрицы через элементарные преобразования
                                  разложения');
            A12=inversion_2(A, n, p)
            disp('Проверка обращения');
            V=inv(A_copy)
          end
       case '3', return;
       otherwise, disp('Введен некорректный символ. Введите один из пред-
                    ложенных.');
    end
end
```

```
Command Window
Лабораторный проект №1 "Стандартные алгоритмы LU-разложения"
Вариант №1
Нажмите "1" для ввода данных;
Нажмите "2" для обработки данных;
Нажмите "3" для выхода из программы
1
Введите матрицу:
 A =
[2 1 1;6 2 1;-2 -2 -1]
Введите вектор:
 b =
fx [0;3;1]
```

Рис. 1. Пример работы меню

Задание № 2 *«Функция факторизации матрицы»*

Алгоритм 1. $L\overline{U}$ -разложение на основе гауссова исключения по столбцам:

Для k=1 до n

Выбираем главный элемент в $A^{(k-1)}$.

Нормируем первую строку матрицы $A^{(k-1)}$.

Для i=k+1 до n

Вычитаем первую строку матрицы $A^{(k-1)}$, умноженную на $a_{ik}^{(k-1)}$, из i-й строки.

Стратегия выбора главного элемента по столбцу. В качестве главного на k-м шаге метода Гаусса выбирается максимальный по модулю элемент первого столбца активной подматрицы. Затем этот элемент меняется местами с диагональным элементом, что соответствует перестановке строк матрицы $A^{(k-1)}$. Строки матрицы $A^{(k-1)}$ остаются на своих местах, переставляются только элементы вектора перестановок p, в котором хранятся номера строк исходной матрицы, т. е. элементы вектора перестановок. Все обращения к элементам нижней треугольной части L и верхней треугольной части $\overline{U}$ матрицы A осуществляются через вектор перестановок.

Программный код функции факторизации ($L\overline{U}$ -разложение):

```
function [A, p, znak, op_numb] = factorization(A, n)

  p = zeros(n,1);
  max = 0;
  imax = 0;
  buf = 0;
  znak=1;
  op_numb = 0;
  % Заполняем массивы перестановок начальными значениями перед началом факторизации
  for i=1:n
    p(i)=i;
  end
  for k=1:n
    %---------Выбор главного элемента----------------
    max=abs(A(p(k),k));
    imax=k;
    for i=(k+1):n
        if(abs(A(p(i),k))>max)
```

```
            max=abs(A(p(i),k));
            imax=i;
        end
    end
        % Обмен
        if(imax ~= k)
            znak=-znak;
          buf=p(k);
           p(k)=p(imax);
          p(imax)=buf;
        end
    %---------------------------------------------
    div = A(p(k),k);
    for q=(k+1):n
       A(p(k),q)= A(p(k),q)/div;%Нормировка
       op_numb = op_numb + 1;
    end
    for i=(k+1):n
       mult = A(p(i),k);
       for j=(k+1):n
          A(p(i),j)= A(p(i),j)- mult *A(p(k),j);
          op_numb = op_numb + 1;
       end
    end
  end
end
```

Вспомогательная функция для учета перестановок строк в факторизованной матрице:

```
function [ PA ] = printPA( A,n,p )
% Вспомогательная функция
  for i=1:n
    for j=1:n
      PA(i,j)=A(p(i),j);
    end
  end
end
```

```
Факторизация:
Факторизация матрицы A с учетом перестановок

PA =

   6.000000000000000   0.333333333333333   0.166666666666667
  -2.000000000000000  -1.333333333333334   0.500000000000000
   2.000000000000000   0.333333333333333   0.500000000000000

Нижняя треугольная часть L

L =

   6.000000000000000                   0                   0
  -2.000000000000000  -1.333333333333334                   0
   2.000000000000000   0.333333333333333   0.500000000000000

Верхняя треугольная часть U

U =

   1.000000000000000   0.333333333333333   0.166666666666667
                   0   1.000000000000000   0.500000000000000
                   0                   0   1.000000000000000
```

Рис. 2. Вывод результатов факторизации

Задание № 3 *«Функция решения системы линейных алгебраических уравнений»*

Поиск решения СЛАУ осуществляется по $L\overline{U}$ разложению. Решение находим в два этапа: сначала находится решение для системы с нижней треугольной матрицей, затем находится решение с верхней треугольной матрицей. Математическое обоснование:

$$Ax = b \Rightarrow (L\overline{U}x = b) \Rightarrow (\overline{U}x = y) \Rightarrow (Ly = b) \Rightarrow y \Rightarrow (\overline{U}x = y) \Rightarrow x.$$

Программный код функции решения СЛАУ:

```
function [x, op_numb] = SLAE(A, b, n, p)
   y = zeros(n,1);
   x = zeros(n,1);
   op_numb = 0;
   for i=1:n
      sum=0;
      for j=1:i
         sum =sum+A(p(i),j)*y(j);
```

```
        op_numb = op_numb + 1;
      end
      y(i)=(b(p(i))-sum)/A(p(i),i);
      op_numb = op_numb + 1;
   end
   for i=n:-1:1
      sum=0;
      for j=i+1:n
         sum =sum+A(p(i),j)*x(j);
         op_numb = op_numb + 1;
      end
      x(i)=y(i)-sum;
   end
end
```

```
Решение СЛАУ

x =

     1
    -1
    -1
```

Рис. 3. Вывод результатов решения СЛАУ

Задание № 4 *«Функция вычисления определителя матрицы»*

Определитель находится как произведение диагональных элементов факторизованной матрицы, поскольку

$$\det A = \det L\overline{U} = \det L \cdot \det \overline{U} = \prod_{i=1}^{n} l_{ii} \cdot 1 = \prod_{i=1}^{n} l_{ii} = \prod_{i=1}^{n} a_{ii}$$

(определитель треугольной матрицы с единичной диагональю равен единице, а определитель треугольной матрицы равен произведению диагональных элементов).

Так как используется первая стратегия выбора главного элемента, то при работе алгоритма факторизации происходит обмен строк. При этом каждый раз при обмене знак определителя меняется на противоположный. Для учета знака в программе используется переменная-флаг `znak`, в котором сохраняется значение –1 или 1, в зависимости от того, четное или нечетное количество перестановок было сделано на данный момент.

Программный код функции вычисления определителя:

```
function [det] = determinant(A, n, p, znak)
   det=1;
   for i=1:n
      det=det*A(p(i),i);
   end
   det= det*znak;
end
```

```
det =

    -4
```

Рис. 4. Вывод результатов вычисления определителя

Задание № 5 *«Функция обращения матрицы через решение системы AX=E»*

Решение осуществляется исходя из равенства $AX=E$, где X – обратная матрица, E – единичная матрица. Разобьем матрицы X и E на столбцы и запишем систему СЛАУ:

$$\begin{cases} Ax_1 = e_1 \\ Ax_2 = e_2 \\ \dots \\ Ax_n = e_n \end{cases}$$, где x_k – k-ый столбец матрицы X, e_k – k-ый столбец матрицы E.

Учитывая, что известно $L\overline{U}$-разложение матрицы A, каждая из СЛАУ решается алгоритмом, описанном в задании № 3. Столбцы обратной матрицы находятся по очереди и затем собираются вместе.

Программный код функции обращения матрицы через решение системы AX=E:

```
function [V, count_ops_1] = inversion_1(A, n, p)
 V = zeros(n,n);
 I = eye(n);
```

```
count_ops_1 = 0;
count_ops_cur = 0;
for count = 1:n
  b = I(:,count);
  y = zeros(n,1);
  x = zeros(n,1);
  [x,count_ops_cur]=SLAE(A, b, n, p);
  V(:,count) = x;
  count_ops_1 = count_ops_1 +count_ops_cur;
end
end
```

```
Обращение матрицы через решение системы AX=E
Обратная матрица первым способом

A11 =

                   0   0.250000000000000   0.250000000000000
  -1.000000000000000   0.000000000000000  -1.000000000000000
   2.000000000000000  -0.500000000000000   0.500000000000000
```

Рис. 5. Результат обращения матрицы через решение системы AX=E

Задание № 6 *«Функция обращения матрицы через элементарные преобразования»*

Пусть $A = L\overline{U}$. Тогда $A^{-1} = \overline{U}^{-1}L^{-1}$. Матрицы L и $\overline{U}$ уже известны. Следовательно, сначала найдем обратные к ним, а затем найдем их произведение и получим матрицу, обратную к A.

Программный код функции обращения матрицы через элементарные преобразования разложения:

```
function [A_ob, count_ops_2] = inversion_2(A, n, p)
count_ops_2 = 0;
% Первый этап - подготовка (обращение элементарных матриц)
```

```
for i=1:n
   for j=(i+1):n
     A(p(i),j)=-A(p(i),j);
   end
end
for j=1:n
   A(p(j),j)=1/A(p(j),j);
   count_ops_2=count_ops_2+1;
   for i=(j+1):n
     A(p(i),j)=-A(p(i),j)*A(p(j),j);
     count_ops_2=count_ops_2+1;
   end
end
disp('Результат выполнения первого этапа');
PA=printPA(A,n,p)
% Считаем матрицу U^{-1}
for k=n:(-1):2
         for i=1:(k-2)
            for j=k:n
             A(p(i),j)= A(p(i),j)+A(p(i),k-1)*A(p(k-1),j);
             count_ops_2=count_ops_2+1;
            end
         end
end
% Считаем матрицу L^{-1}
for k=1:(n-1)
   for i=(k+2):n
      for j=1:k
         A(p(i),j)=A(p(i),j) + A(p(i),k+1)*A(p(k+1),j);
         count_ops_2=count_ops_2+1;
      end
   end
   for j=1:k
      A(p(k+1),j)=A(p(k+1),j) * A(p(k+1),k+1);
      count_ops_2=count_ops_2+1;
```

```
   end
end
disp('Результат выполнения второго этапа');
PA=printPA(A,n,p)
% Перемножаем в одной матрице U^{-1} на L^{-1}
for i=1:n
   for j=1:n
      if(i<j)
         sum=0;
         for k=j:n
            sum = sum +A(p(i),k)*A(p(k),j);
            count_ops_2=count_ops_2+1;
         end
      end
      if(i>=j)
         sum =A(p(i),j);
         for k=i+1:n
            sum = sum +A(p(i),k)*A(p(k),j);
            count_ops_2=count_ops_2+1;
         end
      end
      A(p(i),j)=sum;
   end
end
disp('Результат выполнения третьего этапа');
% Учитываем перестановки срок
A_ob=printPA1(A,n,p);
end
```

Вспомогательная функция для учета обратной перестановки столбцов:

```
function [ PA1 ] = printPA1( A,n,p )
% Учитываем перестановки строк при обращении
% Вектор обратных перестановок строк
p1=zeros(n,1);
for i=1:n
 p1(p(i))=i;
```

```
end
for i=1:n
    for j=1:n
       PA1(i,j)=A(p(i),p1(j));
    end
end
end
```

```
Обращение матрицы через элементарные преобразования разложения
Результат выполнения первого этапа

PA =

   0.166666666666667  -0.333333333333333  -0.166666666666667
   0.333333333333333  -0.750000000000000  -0.500000000000000
  -0.333333333333333   0.250000000000000   2.000000000000000

Результат выполнения второго этапа

PA =

   0.166666666666667  -0.333333333333333                   0
  -0.250000000000000  -0.750000000000000  -0.500000000000000
  -0.500000000000000   0.500000000000000   2.000000000000000

Результат выполнения третьего этапа

A12 =

                   0   0.250000000000000   0.250000000000000
  -1.000000000000000   0.000000000000000  -1.000000000000000
   2.000000000000000  -0.500000000000000   0.500000000000000
```

Рис. 6. Результат обращения матрицы через элементарные преобразования

Задание № 7 *«Эксперимент 1 «Решение СЛАУ»*

Цель данного эксперимента – исследовать скорость и погрешность решения СЛАУ. В ходе эксперимента определяется число операций умножения и деления, погрешность и скорость решения СЛАУ со случайно сгенерированными матрицами порядка от 5 до 100 (через 5 порядков). Значения элементов матриц генерируются в диапазоне от –100 до 100. Результаты эксперимента записываются в файл, а затем выводятся на экран в виде таблицы и графиков.

Программный код эксперимента №1:

```
function [] = exp1()
clc();
frmt   = get(0,'Format');
frmtV = get(0,'FormatSpacing');
format('shortG')
format('compact')
exp_Data = zeros(1,100);
count = 1;
for n=5:5:100
   exp_Data(count) = n;
   count = count+1;
   A = randi([-100,100],n,n);
   A_copy=A;
   b = randi([-100,100],n,1);
   % Засекаем время
   tic
   [A, p, znak, op_numb] = factorization(A, n);
   [x, op_numb_2] = SLAE(A, b, n, p);
   exp_Data(count) = toc;
   count = count+1;

   x_ex = zeros(n,1);
   b_ex = zeros(n,1);
   for i=1:n
      x_ex(i) = i;
   end;
   % Вычисляем правую вектор b для точного решения
   for i=1:n
      for j=1:n
         b_ex(i) = b_ex(i) + A_copy(i,j) * x_ex(j);
      end
   end
   [x_calc] = SLAE(A, b_ex, n, p);
   % Вычисляем погрешность найденного решения
   for i=1:n
```

```
        ex = abs(x_ex(i) - x_calc(i));
        if (i == 1)
            max_ex = ex;
        else
            if (ex > max_ex)
                max_ex = ex;
            end
        end
    end
    exp_Data(count) = max_ex;
    count = count +1;
    exp_Data(count) = n^3/3;
    count = count +1;
    exp_Data(count) = op_numb + op_numb_2;
    count = count + 1;
end
dlmwrite('file.dat',exp_Data,';');
clear A;
clear exp_Data;
n_ar = zeros(1,20);
time_ar = zeros(1,20);
ex_ar = zeros(1,20);
th_numb_ar = zeros(1,20);
fact_numb_ar = zeros(1,20);
read_data = dlmread('file.dat', ';');
count = 1;
% Вывод таблицы
disp('        Порядок   Время    Погрешность   Теоретическое ЧО  Реальное
ЧО');
for i=1:5:100
    n_ar(count) = read_data(i);
    time_ar(count) = read_data(i+1);
    ex_ar(count) = read_data(i+2);
    th_numb_ar(count) = read_data(i+3);
    fact_numb_ar(count) = read_data(i+4);
    count = count + 1;
```

```
  str = [round(read_data(i)), read_data(i+1), read_data(i+2),
round(read_data(i+3)), round(read_data(i+4))];
  disp(str);
end
% Вывод графиков
plot(n_ar, time_ar, 'k');
title('Зависимость времени решения от размера матрицы A (мск)');
grid on;
figure;
plot(n_ar, ex_ar, 'k');
title('Зависимость погрешности решения от размера матрицы A');
grid on;
figure;
plot(n_ar, th_numb_ar, n_ar, fact_numb_ar,'--');
title('Реальное и теоретическое количество операций');
grid on;
legend('Теоретич. ЧО','Реальное ЧО',2);
format(frmt)
format(frmtV)
end
```

Порядок	Время	Точность	Теоретическое ЧО	Реальное ЧО
5	0.0039584	0	42	70
10	0.00029873	2.931e-014	333	440
15	0.00044193	8.8818e-015	1125	1360
20	0.00076479	1.7764e-013	2667	3080
25	0.0012632	1.9362e-013	5208	5850
30	0.0019874	6.3949e-014	9000	9920
35	0.0028923	1.8474e-013	14292	15540
40	0.0043064	1.0516e-012	21333	22960
45	0.0054849	6.6791e-013	30375	32430
50	0.0073148	4.6896e-013	41667	44200
55	0.0093525	6.6791e-013	55458	58520
60	0.011702	5.1159e-013	72000	75640
65	0.014792	6.0396e-013	91542	95810
70	0.02053	7.1054e-013	1.1433e+005	1.1928e+005
75	0.021371	1.1136e-010	1.4063e+005	1.463e+005
80	0.025844	2.5295e-012	1.7067e+005	1.7712e+005
85	0.030643	2.5722e-012	2.0471e+005	2.1199e+005
90	0.035586	1.5419e-012	2.43e+005	2.5116e+005
95	0.041318	6.9917e-012	2.8579e+005	2.9488e+005
100	0.048416	3.2401e-012	3.3333e+005	3.434e+005

Рис. 7. Таблица экспериментальных данных для решения СЛАУ

Рис. 8. График зависимости времени решения от порядка матриц для решения СЛАУ

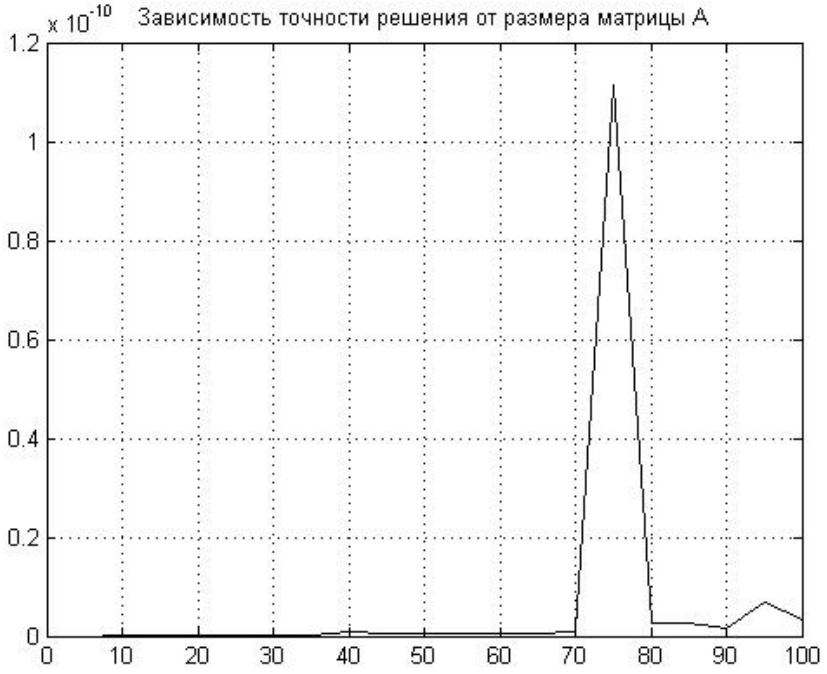

Рис. 9. График зависимости погрешности решения от порядка матриц для решения СЛАУ

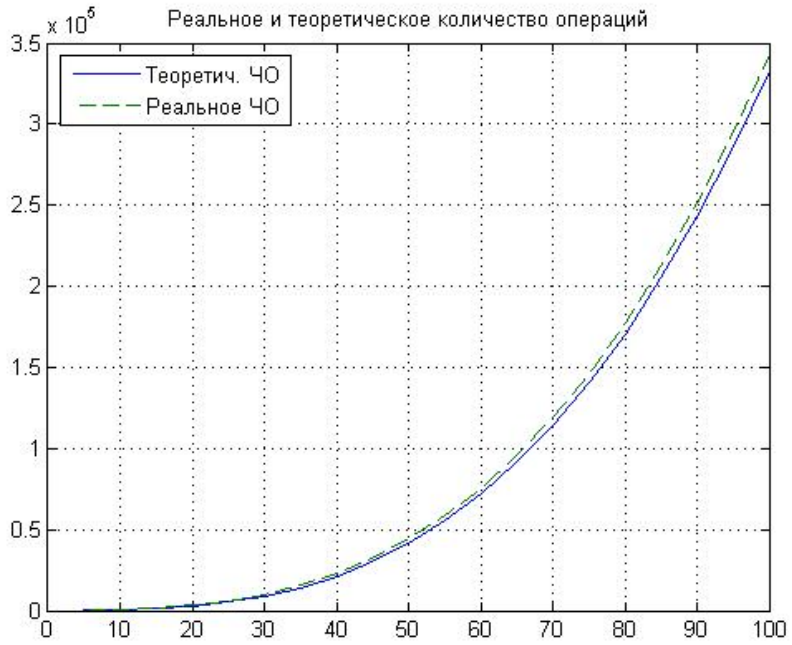

Рис. 10. График зависимости реального и теоретического числа операций от размера матрицы для решения СЛАУ

Далее требуется сформулировать полученные результаты исследования скорости и погрешности решения СЛАУ с помощью заданного алгоритма факторизации!

Задание № 8 *«Эксперимент 2 «Решение СЛАУ с плохо обусловленными матрицами»*

Цель данного эксперимента – исследовать скорость и погрешность решения СЛАУ с плохо обусловленными матрицами.

В ходе эксперимента определяется число операций умножения и деления, погрешность и скорость решения СЛАУ для различных видов плохо обусловленных матриц. Результаты записываются в файл, а затем выводятся на экран в виде таблицы и графиков. Программный код аналогичен программному коду для эксперимента 1, кроме генерации матриц: плохо обусловленные матрицы нефиксированного размера генерируются порядка от 4 до 40 (через 4). Графики зависимости времени решения, погрешности от размера матриц приведены только для матриц нефиксированного размера. Графики зависимости числа операций не строим, так как они являются аналогичными графикам эксперимента 1.

Программный код Эксперимента № 2

```
function [] = exp2(tM)
if tM==1
```

```
    disp('Результаты эксперимента с матрицей Гильберта');
    exp2_bad_nonfixed(1);
elseif tM==2
    disp('Результаты эксперимента с матрицей № 2');
    exp2_bad_fixed(2);
elseif tM==3
    disp('Результаты эксперимента с матрицей № 3');
    exp2_bad_fixed(3);
elseif tM==4
    disp('Результаты эксперимента с матрицей № 4');
    exp2_bad_nonfixed(4);
elseif tM==5
    disp('Результаты эксперимента с матрицей № 5');
    exp2_bad_nonfixed(5);
elseif tM==6
    disp('Результаты эксперимента с матрицей № 6');
    exp2_bad_fixed(6);
elseif tM==7
    disp('Результаты эксперимента с матрицей № 7');
    exp2_bad_nonfixed(7);
elseif tM==8
    disp('Результаты эксперимента с матрицей № 8');
    exp2_bad_nonfixed(8);
elseif tM==9
    disp('Результаты эксперимента с матрицей № 9');
    exp2_bad_nonfixed(9);
elseif tM==10
    disp('Результаты эксперимента с матрицей № 10');
    exp2_bad_fixed(10);
else disp('Неверный номер матрицы!')
end
```

Вспомогательная функция для вычисления погрешности обращения:

```
function [norm] = exact_func(A_orig, A_ob, n)
% вычисляем погрешность обращения
  Mult = A_orig*A_ob;
  Mult=eye(n,n)-Mult;
```

```
  for  i=1:n
     sum_a = 0;
     sum_mult = 0;
     for j=1:n
        sum_a = sum_a + abs(A_orig(i,j));
        sum_mult = sum_mult + abs(Mult(i,j));
     end
     if (i==1)
        max_a = sum_a;
        max_mult = sum_mult;
     else
        if (sum_a > max_a)
           max_a = sum_a;
        end
        if (sum_mult > max_mult)
           max_mult = sum_mult;
        end
     end
  end
  norm = max_mult/max_a;
end
```

Функция для эксперимента с матрицами нефиксированного размера:

```
function [] = exp2_bad_nonfixed(tM)
clc();
frmt   = get(0,'Format');
frmtV = get(0,'FormatSpacing');
format('shortG')
format('compact')
exp_Data = zeros(1,50);
count=1;
for n=4:4:40
   exp_Data(count) = n;
   count = count+1;
   A=zeros(n,n);
```

```
 % Матрица Гильберта
 if (tM==1)
    for i=1:n
     for j=1:n
      A(i,j)=1.0/(i+j-1);
     end
    end
 end
 % Матрица № 4
if (tM==4)
 for i=1:n
   for j=1:n
     if (i==j)
        A(i,i) = 0.01/(n-i+1)/(i+1);
      elseif (i<j)
        A(i,j) = 0;
     else
        A(i,j) = i*(n-j);
      end
    end
 end
end
 % Матрица № 5
if (tM==5)
 for i=1:n
   for j=1:n
     if (i==j)
        A(i,i) = 0.01/(n-i+1)/(i+1);
      elseif (i<j)
        A(i,j) = j*(n-j);
     else
        A(i,j) = i*(n-j);
      end
    end
 end
end
```

```
 % Матрица № 7
if (tM==7)
 w = 5;
 for i=1:n
    for j=1:n
       if (i==j)
          A(i,j) = w^(abs(n-2*j)/2);
       elseif ((i==1) || (j==1))
          A(i,j) = A(1,1)/w^j;
       elseif((i==n) || (j==n))
          A(i,j) = A(n,n)/w^j;
       end
    end
end
end
 % Матрица № 8
if (tM==8)
h = 0.000001;
for i=1:n
   for j=1:n
      A(i,j) = exp(i*j*h);
   end
end
end
 % Матрица № 9
if (tM==9)
c = 99999;
for i=1:n
   for j=1:n
      A(i,j) = c + log2(i*j);
   end
end
end

 x_ex = zeros(n,1);
 b_ex = zeros(n,1);
```

```
    for i=1:n
        x_ex(i) = i;
    end;
    % Вычисляем правую вектор b для точного решения
    for i=1:n
        for j=1:n
            b_ex(i) = b_ex(i) + A(i,j) * x_ex(j);
        end
    end
    % Засекаем время
    tic
    [A, p, znak, op_numb] = factorization(A, n);
    [x_calc,op_numb_2] = SLAE(A, b_ex, n, p);
    exp_Data(count) = toc;
    count = count+1;
    % Вычисляем погрешность найденного решения
    for i=1:n
        ex = abs(x_ex(i) - x_calc(i));
        if (i == 1)
            max_ex = ex;
        else
            if (ex > max_ex)
                max_ex = ex;
            end
        end
    end
    exp_Data(count) = max_ex;
    count = count +1;
    exp_Data(count) = n^3/3;
    count = count +1;
    exp_Data(count) = op_numb + op_numb_2;
    count = count + 1;
end
dlmwrite('file.dat',exp_Data,';');
clear A;
clear exp_Data;
```

```
n_ar = zeros(1,10);
time_ar = zeros(1,10);
ex_ar = zeros(1,10);
th_numb_ar = zeros(1,10);
fact_numb_ar = zeros(1,10);
read_data = dlmread('file.dat', ';');
count = 1;
   % Вывод таблицы
disp('          Порядок   Время    Погрешность   Теоретическое ЧО  Реальное
ЧО');
for i=1:5:50
  n_ar(count) = read_data(i);
  time_ar(count) = read_data(i+1);
  ex_ar(count) = read_data(i+2);
  th_numb_ar(count) = read_data(i+3);
  fact_numb_ar(count) = read_data(i+4);
  count = count + 1;
  str = [round(read_data(i)), read_data(i+1), read_data(i+2),
   round(read_data(i+3)), round(read_data(i+4))];
  disp(str);
end
   % Вывод графиков
plot(n_ar, time_ar, 'k');
title('Зависимость времени решения от размера матрицы (мск)');
grid on;
figure;
plot(n_ar, ex_ar, 'k');
title('Зависимость погрешности решения от размера матрицы');
grid on;
figure;
plot(n_ar, th_numb_ar, n_ar, fact_numb_ar,'--');
title('Теоретическое и реальное количество операций');
grid on;
legend('Теоретич. ЧО','Реальное ЧО',2);
format(frmt)
format(frmtV)
```

```
end
```

Функция для эксперимента с матрицами фиксированного размера:

```
function [] = exp2_bad_fixed(tM)
frmt   = get(0,'Format');
frmtV = get(0,'FormatSpacing');
format('shortG')
format('compact')
exp_Data = zeros(1,5);
count=1;
   % Матрица № 2
if (tM==2)
   n=20;
   exp_Data(count) = n;
   count = count+1;
    A = zeros(n,n);
   for i=1:(n-1)
     A(i,i) = 1;
     A(i,i+1) = 1;
   end
   A(n,n)=1;
end
   % Матрица № 3
if (tM==3)
 n=7;
 exp_Data(count) = n;
 count = count+1;
 A =[5 4 7 5 6 7 5;
    4 12 8 7 8 8 6;
    7 8 10 9 8 7 7;
    5 7 9 11 9 7 5;
    6 8 8 9 10 8 9;
    7 8 7 7 8 10 10;
    5 6 7 5 9 10 10];
end
   % Матрица № 6
```

```
if (tM==6)
 n=8;
 exp_Data(count) = n;
 count = count+1;
 t = pi*0.999;
 A = [cot(t) csc(t) (1-cot(t)) csc(t) 1 1 1 1;
     -csc(t) cot(t) -csc(t) (1+cot(t)) 1 1 1 1;
     (1-cot(t)) csc(t) cot(t) csc(t) (1-cot(t)) csc(t) 1 1
     -csc(t) (1+cot(t))  -csc(t) cot(t) -csc(t) (1+cot(t)) 1 1;
     1 1 (1-cot(t)) csc(t) cot(t) csc(t) (1-cot(t)) csc(t);
     1 1 -csc(t) (1+cot(t)) -csc(t) cot(t) -csc(t) (1+cot(t));
     1 1 1 1 (1-cot(t)) csc(t) cot(t) csc(t);
     1 1 1 1 -csc(t) (1+cot(t)) -csc(t) cot(t)];
 end
  % Матрица № 10
if (tM==10)
 n=4;
 exp_Data(count) = n;
 count = count+1;
 A = [ 0.9143*10^(-4)   0   0   0;
     0.8762   0.7156*10^(-4)   0   0;
     0.7943   0.8143   0.9504*10^(-4)   0;
     0.8017   0.6123   0.7165   0.7123*10^(-4) ];
end
  x_ex = zeros(n,1);
  b_ex = zeros(n,1);
  for i=1:n
     x_ex(i) = i;
  end;
  % Вычисляем вектор b, соответствующий точному решению
  for i=1:n
     for j=1:n
        b_ex(i) = b_ex(i) + A(i,j) * x_ex(j);
     end
  end
  tic
```

```
    [A, p, znak, op_numb] = factorization(A, n);
    [x_calc,op_numb_2] = SLAE(A, b_ex, n, p);
    exp_Data(count) = toc;
    count = count+1;
    % Вычисляем погрешность найденного решения
    for i=1:n
        ex = abs(x_ex(i) - x_calc(i));
        if (i == 1)
            max_ex = ex;
        else
            if (ex > max_ex)
                max_ex = ex;
            end
        end
    end
    exp_Data(count) = max_ex;
    count = count +1;
    exp_Data(count) = n^3/3;
    count = count +1;
    exp_Data(count) = op_numb + op_numb_2;
    count = count + 1;
dlmwrite('file.dat',exp_Data,';');
clear A;
clear exp_Data;
read_data = dlmread('file.dat', ';');
    % Вывод таблицы
disp('          Порядок   Время    Погрешность   Теоретическое ЧО  Реальное
ЧО');
   n_ar = read_data(1);
   time_ar = read_data(2);
   ex_ar = read_data(3);
   th_numb_ar = read_data(4);
   fact_numb_ar = read_data(5);
   str = [round(read_data(1)), read_data(2), read_data(3), round(read_data(4)),
round(read_data(5))];
   disp(str);
```

```
format(frmt)
format(frmtV)
end
```

1. Матрица Гильберта

Результаты эксперимента с матрицей Гильберта

Порядок	Время	Точность	Теоретическое ЧО	Реальное ЧО
4	0.0016928	7.6117e-013	21	40
8	0.00028076	1.7042e-006	171	240
12	0.00021763	1.0521	576	728
16	0.00036854	110.61	1365	1632
20	0.00069498	344.48	2667	3080
24	0.00091005	687.21	4608	5200
28	0.0013345	3629.4	7317	8120
32	0.0020624	2036.4	10923	11968
36	0.0025459	3227.6	15552	16872
40	0.0033517	31343	21333	22960

Рис. 11. Таблица экспериментальных данных для матрицы Гильберта

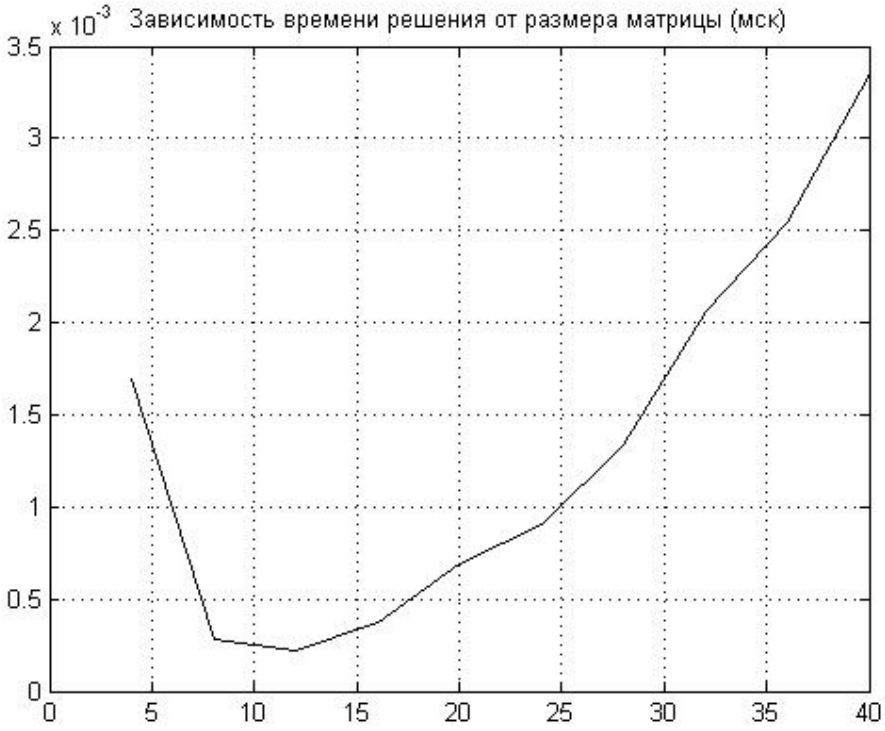

Рис. 12. График зависимости времени решения от размера матрицы Гильберта

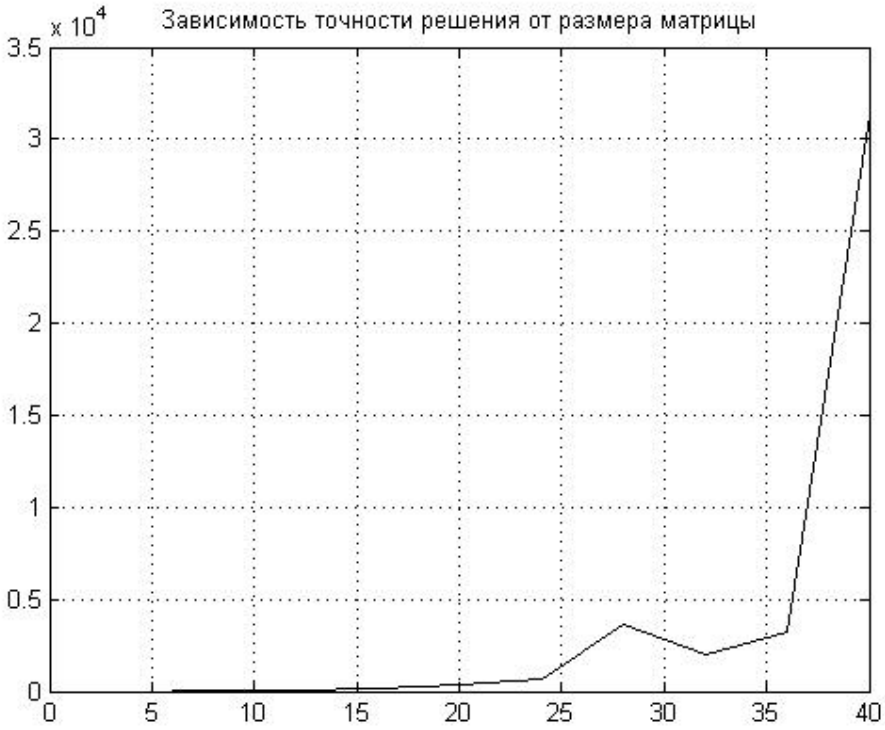

Рис. 13. График зависимости погрешности решения от размера матрицы Гильберта

Вычислительный эксперимент с матрицами № 4, 5, 7, 8, 9 выполняется аналогично!

2. Матрица № 2

Результаты эксперимента с матрицей № 2

Порядок	Время	Точность	Теоретическое ЧО	Реальное ЧО
20	0.0034226	0	2667	3080

Рис. 14. Таблица экспериментальных данных для матрицы № 2

Вычислительный эксперимент с матрицами № 3, 6, 10 выполняется аналогично!

Далее требуется сформулировать полученные результаты исследования скорости и погрешности решения СЛАУ с помощью заданного алгоритма факторизации для каждой плохо обусловленной матрицы (при исследовании необходимо варьировать параметры матрицы)!

Задание № 9 *«Эксперимент 3 «Обращение случайных матриц»*

Цель эксперимента – исследовать скорость и погрешность обращения матриц разными способами: через решение системы $AX=E$, где E – единичная матрица, и через разложение матрицы A в произведение элементарных матриц.

В ходе эксперимента определяется число операций умножения и деления, погрешность и скорость обращения случайно сгенерированных матриц порядка от 5 до 100 (через 5 порядков). Значения элементов матриц генерируются в диапазоне от -100 до 100.

Для вычисления погрешности обращения матрицы A умножаем исходную матрицу на найденную обратную и вычитаем результат из единичной матрицы, затем находим норму найденной разности и делим ее на норму исходной матрицы. Норму матрицы вычисляем как максимальная сумма модулей элементов строк.

Результаты эксперимента записываются в файл, а затем выводятся на экран в виде таблицы и графиков.

Программный код функции вычисления погрешности обращения матрицы:

```
function [norm] = exact_func(A_orig, A_ob, n)
% вычисляем погрешность обращения
   Mult = A_orig*A_ob;
   Mult=eye(n,n)-Mult;

   for  i=1:n
      sum_a = 0;
      sum_mult = 0;
      for j=1:n
         sum_a = sum_a + abs(A_orig(i,j));
         sum_mult = sum_mult + abs(Mult(i,j));
      end
      if (i==1)
         max_a = sum_a;
         max_mult = sum_mult;
      else
         if (sum_a > max_a)
```

```
            max_a = sum_a;
        end
        if (sum_mult > max_mult)
            max_mult = sum_mult;
        end
    end
  end
  norm = max_mult/max_a;
end
```

Программный код эксперимента № 3:

```
function [] = exp3()
clc;
frmt   = get(0,'Format');
frmtV = get(0,'FormatSpacing');
format('shortG')
format('compact')
exp_Data = zeros(1,160);
count = 1;
for n=5:5:100
   exp_Data(count) = n;
   count = count+1;
   A = randi([-100,100],n,n);
   A_orig = A;
   [A, p, znak, op_numb] = factorization(A, n);
   tic
   [V, count_ops_1] = inversion_1(A, n, p);
   exp_Data(count) = toc;
   count = count+1;
   tic
   [A_ob, count_ops_2] = inversion_2(A, n, p);
   exp_Data(count) = toc;
   count = count+1;
   [norm] = exact_func(A_orig, V, n);
   exp_Data(count) = norm;
   count = count+1;
```

```
    [norm] = exact_func(A_orig, A_ob, n);
    exp_Data(count) = norm;
    count = count+1;
    exp_Data(count) = n^3;
    count = count +1;
    exp_Data(count) = count_ops_1+op_numb;
    count = count+1;
    exp_Data(count) = count_ops_2+op_numb;
    count = count+1;
end
dlmwrite('file_obr.dat',exp_Data,';');
clear A;
clear exp_Data;
n_ar = zeros(1,20);
time_ar_1 = zeros(1,20);
time_ar_2 = zeros(1,20);
ex_ar_1 = zeros(1,20);
ex_ar_2 = zeros(1,20);
th_numb_ar = zeros(1,20);
fact_numb_ar_1 = zeros(1,20);
fact_numb_ar_2 = zeros(1,20);
read_data = dlmread('file_obr.dat', ';');
count = 1;
disp('          Порядок  Время1     Время2     Погрешность1   Погрешность2
Теор. ЧО   Реал. ЧО_1   Реал. ЧО_2');
for i=1:8:160
  n_ar(count) = read_data(i);
  time_ar_1(count) = read_data(i+1);
  time_ar_2(count) = read_data(i+2);
  ex_ar_1(count) = read_data(i+3);
  ex_ar_2(count) = read_data(i+4);
  fact_numb_ar_1(count) = read_data(i+5);
  fact_numb_ar_2(count) = read_data(i+6);
  th_numb_ar(count) = read_data(i+7);
  count = count + 1;
```

```
  str = [round(read_data(i)), read_data(i+1), read_data(i+2), read_data(i+3),
read_data(i+4), round(read_data(i+5)),
round(read_data(i+6)),round(read_data(i+7))];
  disp(str);
end
% Вывод графиков
plot(n_ar, time_ar_1, n_ar, time_ar_2, '--');
title('Зависимость времени обращения от размера матрицы A (мск)');
legend('Первый способ','Второй способ',2);
grid on;
figure;
plot(n_ar, ex_ar_1, n_ar, ex_ar_2, '--');
title('Зависимость погрешности обращения от размера матрицы A');
legend('Первый способ','Второй способ',2);
grid on;
figure;
plot(n_ar, th_numb_ar, '-o', n_ar, fact_numb_ar_1, '-x', n_ar, fact_numb_ar_2,
'-*');
title('Количество операций от размера матрицы A');
legend('Теоретическое ЧО','Реальное ЧО_1','Реальное ЧО_2',2);
grid on;
format(frmt)
format(frmtV)
end
```

Порядок	Время1	Время2	Точность1	Точность2	Теор. ЧО	Реал. ЧО_1	Реал. ЧО_2
5	0.0021476	0.0037126	2.2415e-018	3.9567e-018	125	190	125
10	0.00053484	0.0021445	4.8213e-018	7.4494e-018	1000	1430	1000
15	0.00083049	0.004195	4.3785e-018	5.5245e-018	3375	4720	3375
20	0.0014808	0.00634	1.6037e-017	2.2293e-017	8000	11060	8000
25	0.0024206	0.0096209	5.1472e-017	6.3268e-017	15625	21450	15625
30	0.0044845	0.015898	1.3092e-017	1.3433e-017	27000	36890	27000
35	0.0049645	0.018918	5.381e-017	6.1379e-017	42875	58380	42875
40	0.0068405	0.040425	1.0653e-016	1.3057e-016	64000	86920	64000
45	0.0092498	0.033725	2.391e-016	3.4767e-016	91125	1.2351e+005	91125
50	0.010741	0.036903	1.3329e-017	2.0218e-017	1.25e+005	1.6915e+005	1.25e+005
55	0.016907	0.047304	1.3636e-016	1.7652e-016	1.6638e+005	2.2484e+005	1.6638e+005
60	0.019835	0.073862	1.2767e-017	2.7887e-017	2.16e+005	2.9158e+005	2.16e+005
65	0.020892	0.090163	3.1243e-017	4.6314e-017	2.7463e+005	3.7037e+005	2.7463e+005
70	0.025872	0.085474	1.9876e-017	6.0511e-017	3.43e+005	4.6221e+005	3.43e+005
75	0.04121	0.095778	1.001e-016	1.6583e-016	4.2188e+005	5.681e+005	4.2188e+005
80	0.03225	0.097317	2.8568e-017	5.5886e-017	5.12e+005	6.8904e+005	5.12e+005
85	0.039445	0.13542	4.254e-017	7.1307e-017	6.1413e+005	8.2603e+005	6.1413e+005
90	0.046592	0.12142	5.5188e-017	9.507e-017	7.29e+005	9.8007e+005	7.29e+005
95	0.048156	0.13242	4.6127e-017	9.9936e-017	8.5738e+005	1.1522e+006	8.5738e+005
100	0.061671	0.14638	3.7417e-017	9.5256e-017	1e+006	1.3433e+006	1e+006

Рис. 15. Таблица экспериментальных данных для эксперимента № 3

Рис. 16. График зависимости времени обращения первым и вторым способом от размера матриц

Из графика, представленного на рис. 16, можно сделать вывод, что время обращения первым способом меньше времени обращения вторым способом, т. е. первый способ обращения матриц эффективнее по времени.

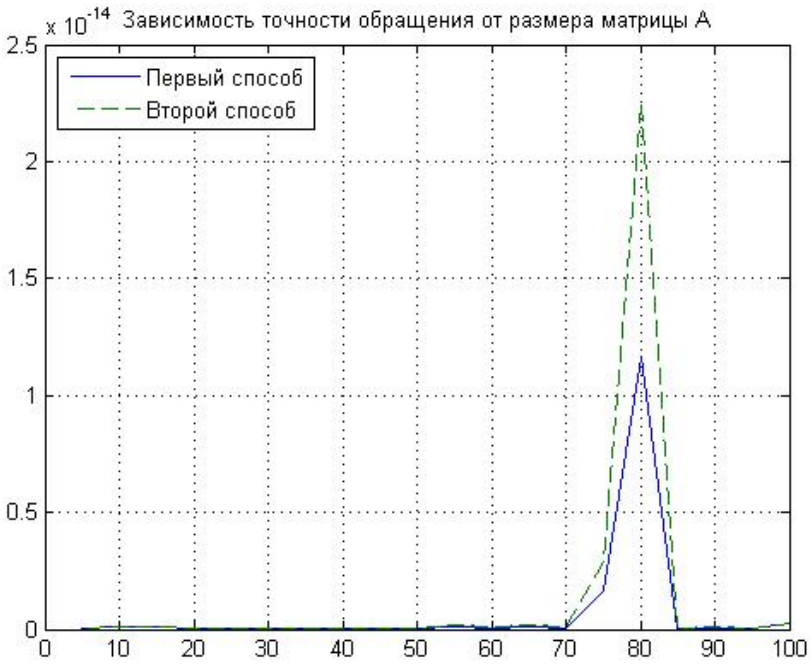

Рис. 17. График зависимости погрешности обращения первым и вторым способом от размера матриц

Из графика, представленного на рис. 17, можно сделать вывод, что погрешность вычисления обратной матрицы первым способом выше, чем вторым.

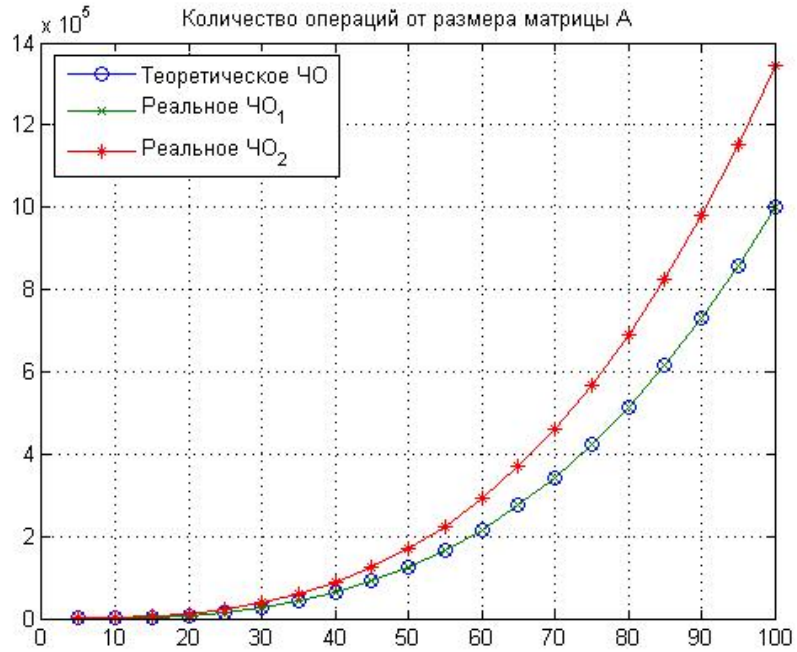

Рис. 18. График зависимости теоретического и реального числа операций при обращении первым и вторым способом от порядка матриц

Общие выводы по проекту: в ходе выполнения лабораторного проекта написан программный проект на языке MATLAB, реализующий $L\overline{U}$ -разложение на основе гауссова исключения по столбцам с выбором главного элемента по столбцу для численного решения системы линейных алгебраических уравнений $Ax=f$, вычисления определителя и нахождения обратной матрицы A^{-1}.

Программа включает в себя следующие функции:

1. Функция факторизации матрицы.
2. Функция решения системы линейных алгебраических уравнений.
3. Функция вычисления определителя матрицы.
4. Функция обращения матрицы через решение системы $AX=E$.
5. Функция обращения матрицы через элементарные преобразования разложения.
6. Также проведены вычислительные эксперименты:
7. Эксперимент 1 «Решение СЛАУ».
8. Эксперимент 2 «Решение СЛАУ с плохо обусловленными матрицами».
9. Эксперимент 3 «Обращение случайных матриц».
10. В ходе выполнения экспериментов оценено время выполнения вычислений, погрешность вычислений и число производимых операций умножения и деления.

Далее необходимо сформулировать содержательные выводы о работе алгоритмов по своему варианту!

ДЛЯ ЗАМЕТОК